İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

RENDEZVOUS POINT SELECTION IN MULTICAST NETWORKS

M.Sc Thesis by

Eng. O. Belgi Özen

Department : Computer Engineering

Programme: Computer Engineering

JANUARY 2005

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

RENDEZVOUS POINT SELECTION IN MULTICAST NETWORKS

M.Sc Thesis by
Osman Belgi ÖZEN, Eng.
(504011423)

Date of submission : 27 December 2004
Date of defence examination: 23 January 2005

Supervisor (Chairman): **Assoc. Prof. Dr. Sema OKTUĞ**

Members of the Examining Cottee **Prof. Dr. Emre HARMANCI (İ.T.Ü)**
Assoc. Prof. Dr. Erdal Çayırcı (Turkish War Colleges)

JANUARY 2005

ACKNOWLEDGEMENTS

I would like to thank to Assoc. Prof. Dr. Sema F. (Akgün) Oktuğ very much who advised me in all stages of this work and shared her experience with me.

January 2005 Osman Belgi Özen

CONTENTS

ABBREVIATIONS

RP	: Rendezvous Point
DR	: Designated Router
Mem-DR	: Member Designated Router
NonMem-DR	: Non-Member Designated Router
MemT(x)	: Member DRs table of Agent x
MRT(x)	: Multicast Routing Table of DR-x
AL(x)	: Agent List of Core x
S	: Source
G	: Multicast Group G
C-RP	: Candidate Rendezvous Point
BSR	: Bootstrap Router
ST	: Steiner Tree
CBT	: Core Base Tree
QoS	: Quality of Service
SPT	: Shortest Path Tree
KBM	: Kou, Markowsky and Berman Tree
MBONE	: IP Multicast Backbone on The Internet
DVRMP	: Active Microwave Instrument
MOSPF	: Multicast Open Shortest Path First
PIM-DM	: Protocol Independent Multicast Dense Mode
PIM-SM	: Protocol Independent Multicast Sparse Mode
BGMP	: Border Gateway Multicast Protocol
MIGP	: Multicast Interior Gateway Protocol
MIN-MEMB	: Minimal Member Protocol
HILLCLIMB	: HillClimbing Protocol
MIGP	: Multicast Interior Gateway Protocol
SCMP	: Scalable Core Migration Protocol
RPF	: Reverse Path Forwarding
IGMP	: Internet Group Management Protocol
DTS	: Directed Steiner Tree
DBST	: Delay Bounded Steiner Tree
TS	: Transit Stub Networks
FR	: Flat Random Networks
RC	: Relay Chat
VC	: Video Conference

FIGURES

Page No

ÇOKLU GÖNDERİM AĞLARINDA MERKEZİ DÜĞÜM SEÇİLMESİ

ÖZET

Bu çalışmada, günümüzde kullanılan seyrek tarzlı çoklu aktarım algoritmalarının bir eksiği olan ve dinamik üyelere sahip çoklu aktarım gruplarında daha belirgin olarak gözlemlenen çoklu gönderim ağacına bağlı servis kalitesinde düşme problemi üzerinde durulmaktadır.

Günümüzde kullanılan seyrek tarzlı çoklu gönderim algoritmalarında, merkez düğüm seçilmesi yönetimsel olarak yapılmaktadır ve durağan bir seçim yöntemidir. Bu nedenle, zamanla çoklu aktarım grubuna yeni alıcılar ve kaynaklar üye olduklarında ya da ayrıldıklarında, yönetimsel olarak seçilen merkez düğümlü çoklu aktarım ağaçlarında servis kalitesi düşer. Beklenen servis kalitesine tekrar ulaşabilmek için, yeni bir merkez düğüm seçilmeli ve çoklu gönderim ağacı yeni bulunan merkez düğüme göre oluşturulmalıdır.

Yeni merkez düğümü seçerken, o anda aktif olan kaynak ve mümkünse alıcıların konumuna bakılarak yeni bir merkez düğümün hesaplanması doğru bir yaklaşımdır. Çoklu aktarım grubunun kaynak ve üyelerinden oluşan ağın ağırlık merkezine yakın yerlerde yeni merkez düğümü seçmenin iyi sonuç vereceği söylenebilir. Fakat internetin karmaşık yapısı ve bu yapının tam olarak modellenememesi sebebiyle, bazı yaklaşımlarda bulunarak merkez düğüm seçimi yapılmak zorundadır.

Bu çalışmada, var olan protokollerden PIM-SM çoklu gönderim protokolü ile merkez düğümün dinamik değişmesine olanak veren SCMP çoklu gönderim protokolü incelenmiş, birbirleriyle karşılaştırılmış ve merkez düğümün yer değiştirilmesinin sağladığı avantajlar ve dezavantajlar farklı tipteki ağlar ve çoklu aktarım senaryoları üzerinde denenerek belirlenmeye çalışılmıştır. Ayrıca, yapılan bu çalışma sırasında esnek bir çoklu gönderim senaryo üreteci geliştirilmiştir.

RENDEZVOUS POINT SELECTION IN MULTICAST NETWORKS

SUMMARY

In this study, the focus is on the problem of the degradation of the multicast trees used in sparse mode multicast protocols, which have dynamic members, due to inefficiency in the location of the core (rendezvous) router as time proceeds.

In sparse mode multicast protocols, the rendezvous point is chosen administratively and it is a static selection method unresponsive to the changes in the network dynamics. Therefore, when new sources or receivers join/leave the multicast group by time, the quality of service(QoS) provided by the multicast tree degrades. A better rendezvous point should be selected to prevent this problem and a new multicast tree must be reconstructed rooted at the new RP.

The location of the sources and the receivers should be considered at the RP selection process in order to increase the efficiency of the multicast tree. Choosing an RP in the topological center of the graph formed by the sources and the receivers will give the optimum result. However, the topological center of a graph minimizing the delay may not be calculated correctly in a polynomial time for the existing network structures on the internet. So, some assumptions are made in order to calculate an efficient RP.

In this study, PIM-SM protocol,with static RP, is compared with SCMP protocol which enables the RP to be changed. The advantages and the disadvantages of dynamic RP relocation process is investigated for different type of networks and multicast scenarios. During this work, a flexible multicast scenario generator is developed and used.

1. INTRODUCTION

Internet is a great communication network that enables computers from different regions to communicate with each other efficiently. As the time and the cost to obtain/share information in the internet decreased significantly, its usage is increased tremendously.

The applications like e-mail programs, chat applications, internet browsers etc. accelerated the use of internet among people. As time lapses, people need different kinds of software to communicate with each other and to share information among them. Nowadays, the softwares commonly used by people to communicate rely on unicast communication (one-to-one) in which data is sent between a sender and a receiver. In other words, when a group communication is required, the data is separately sent to each computer and network resources are wasted. Applications like video-conferencing, tv and radio broadcast over the Internet require multiple receivers/sources receive the same data concurrently, thus, causing a separate copy of the data sent to each receiver which results in a high traffic load on the Internet. To make it clear, let's think about a company that wants to broadcast video-streams (a tv program, music concert etc.) to its customers. If the company has a few thousand customers, then it may be convenient to use unicast communication even though it will require high bandwidth and fast hardware resources. But when the number of customers is over a few hundred thousand, it will be nearly impossible to overcome this problem.

Due to the different types of applications requiring concurrent communication with more than two endpoints, a new paradigm called *group communication* is introduced. By using specific routing software for group communication, the traffic load and delay on the internet are decreased.

Multicasting is a new communication type which helps groups communicate efficiently by using less network resources and cause less delay than unicast communication. Multicast communication is currently not supported widely on the

internet and there are still open issues that should be searched like scalability of the communication, the support for different QoS requirements, the security of communication, congestion avoidance etc. On the other hand, there exists some routing protocols already defined for multicast communication [1-5] and these protocols are used on the routers in MBONE (IP Multicast Backbone On The Internet) which enables people from different regions to communicate with each other by using softwares that supports multicast communication. Since most of the routers on the internet does not support multicasting, the data packets are encapsulated and sent as unicast until a multicast enabled network is reached which is called *tunneling*. In addition to MBONE support for multicast communication, companies may enable multicast communication in their networks by configuring some routers as multicast routers internally.

Considering multicast applications with different requirements, multicast communication can be classified into two types: source-specific and group-shared. In source specific multicast communication, one node in the multicast group sends data while the other nodes receive it. In group-shared multicast communication, each node in the multicast group can not only send data to the multicast group but also receive data from other nodes in the multicast group. Group-shared multicast protocols use a core router, in other words rendezvous point(RP), to send data to receivers in the multicast group. Therefore, the location of the core point is a key point that should be considered in group-shared multicast protocols in order to achieve the desired QoS for the multicast communication.

In this thesis, we focus on the algorithms [10-17] that tries to relocate the core router, by choosing a better one in the network when the network dynamics (the changes in the location of receivers/sources or the changes in the quality of the multicast tree constructed in the algorithm) changes, in order to provide a better communication quality in terms of delay and bandwidth to the members of the multicast group.

It should be noted that the simplicity of the algorithm is important. In addition to this, the overhead in terms of processing and memory cost on routers, the complexity in terms of the extra messages that flows between the nodes in the network, the performance gain in comparison to the other algorithms, and the packet loss rate during relocation are some other important problems which should be considered by a good multicast RP relocation algorithm.

In this work, we investigate and compare algorithms according to the points listed above. We implemented the algorithm proposed in [11] and PIM-SM [7]. We compare the tree cost, delay variance, average delay and the message traffic due to communication for different network types and different multicast scenarios. We create a software which generates different realistic multicast scenarios to be used in the simulations.

The rest of this work is organized as follow. Section 2 gives detailed information about the multicast algorithms and protocols. Section 3 describes the issues of RP relocation and the algorithms proposed for RP relocation. The details of my work and the simulation results are given in Section 4. Conclusions are finally stated in Section 0.

2. MULTICASTING FUNDAMENTALS

Multicasting is a communication mechanism in order to send data to a group of receivers in an efficient way. In multicast transmission, the source sends each datagram once no matter how many receivers there are. There is only one copy of the datagram on the physical link at a time. The datagram is copied and routed to different links by the multicast routers if it is necessary. Therefore, it reduces the amount of network traffic in contrast to unicast communication mechanism in which point-to-point connections are established between the source and each of the receivers.

In Multicast routing, different nodes at the same/different networks forms a group. The main idea is based on group model. Groups have some characteristics. Each group has a group address which represents a session between one or more senders and one or more receivers. A sender transmits a datagram addressed to the multicast group address. It does not have to know anything about the receivers or where they are located. The only information that needs to be known is the group address of the receivers. During the flow of the data along the multicast tree, the data may be replicated on routers if there is more than one outgoing interface that connects the receivers to the multicast tree.

According to the flow of information from sources to receivers, the multicast communication can be classified into two types: Source-Specific multicast communication and Group-Shared multicast communication. In source-specific multicast, there is a one-to-many relation; while the source, which is a single node, sends the data, all the other nodes receive it. However, in group-shared multicast, any node can not only send data to the group but also receive data from them.

The networks, in general, can be modeled as graphs. In multicast routing, we are interested in some part of the graph in which it contains the nodes in the multicast group. Therefore, the problem of multicast routing can be defined as finding a spanning tree in the related graph which includes all the nodes in the multicast group.

And the construction of the tree may differ whether the multicast communication is source-specific or group-shared [1] which will be described in the following sections.

2.1 Types of Multicast Communication

In Multicast routing algorithms, the way of constructing the multicast tree is important. The main factor which affects the way of tree construction is the type of the multicast communication. For source-specific multicast communication, *source-specific trees* are used. Similarly, for group-shared multicast communication, *group-shared trees* are used.

Group-shared trees give better results than source-specific trees if we take the average of average source-specific delay for each node in the multicast group. On the other hand, source-specific trees give better results if we calculate the average of the delays between the source and each node in the multicast group. For example let's look at the example given in figure 2.1 [1].

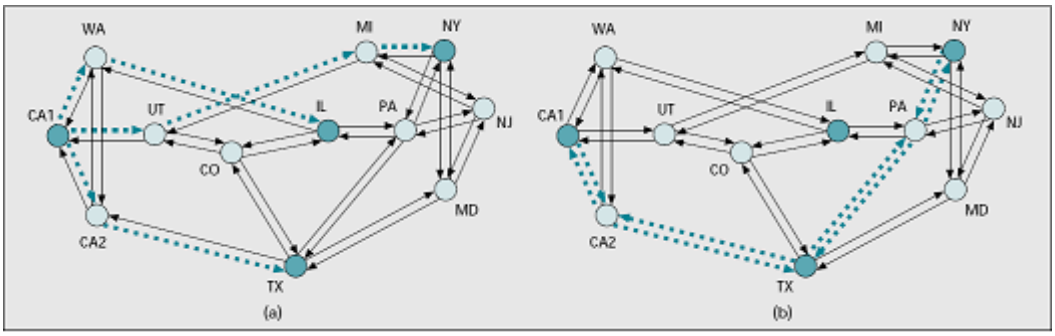


Fig 2.1 an example of a source-specific (a) and a group shared tree (b) tree

The multicast tree in figure 2.1a is a source-specific tree rooted at source CA1 with a source specific delay of (2+2+3) / 3 which is equal to 2.33. But its average group shared delay, which is the average of source specific delay calculated for each node (rooted at that node) in figure 2.1a, is (7/3 + 11/3 + 11/3 + 13/3) / 4 which is 3.5 as a result. On the other hand, if we make the same calculation for the group shared tree, the source specific delay of the tree rooted at node CA1 is 3.33 and the average source specific delay of the tree is 2.67. It is clearly seen from the example that the source specific trees give better result than the group shared tree for source specific delay, whereas, the group shared tree is better than the source specific tree in terms of the average of source specific delays as it is expected.

2.2 Multicast Routing Algorithms

Multicast routing algorithms differ from each other in the way they construct the multicast tree. These algorithms are used in the multicast protocols. Some of the algorithms have high complexity but ensure less delay than the others and some others ensure low complexity and low bandwidth usage while causing more delay. The type of the tree, that is used, depends on the protocol running on routers which can also be changed according to the requirements of the multicast application used. In the following sections, various multicast algorithms and protocols are described briefly.

2.2.1 Shortest Path Tree

Shortest path trees are used for source specific multicast communication. First of all, shortest paths between the source and each of the multicast group members except the source are found. Then a graph is constructed by taking the union of the shortest paths. Lastly, the loops in the final graph are removed to obtain the shortest path multicast tree for the source. The shortest path trees are used to minimize the source specific multicast delays, while, it requires more network resources due to the separate branch which is used to reach every group member. It is not an optimal solution to use shortest path trees with a large number of groups and with each group having a large number of sources since it requires large storage capacity and more calculations on routers and uses high network resources. It is appropriate in dense multicast groups and used with protocols such as DVMRP(Distance Vector Multicast Routing Protocol) [1-3], PIM(Protocol Independent Multicast)-Dense Mode [2,5], MOSPF(Multicast Open Shortest Path First) [1-3] which are used for dense-groups and are not scalable for large networks but has good QoS [1]. An example of the shortest path tree is given

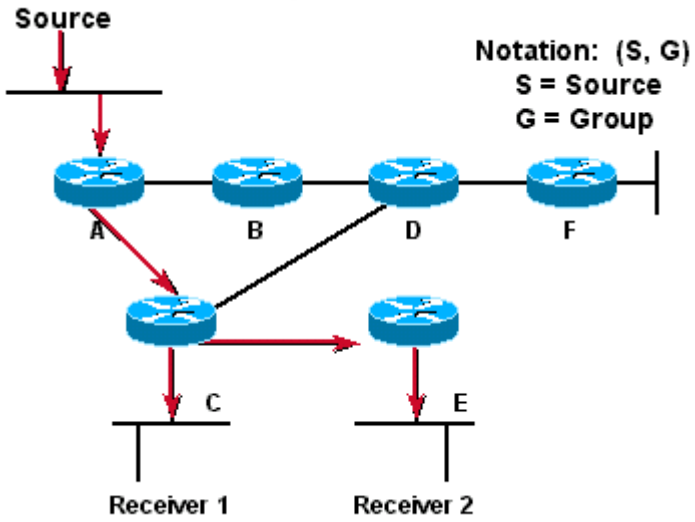


Fig 2.2 An example of a source specific tree[1]

2.2.2 Reverse Shortest Path Tree

To construct a Reverse Shortest Path Tree, the steps listed below are followed [1-5]:

i) After the source broadcasts packets to the network, the first-hop router receive the packet and send it on all outgoing interfaces.

ii) Each router receiving a packet performs a Reverse Path Forwarding (RPF) check. That is, each router checks to see if the incoming interface on which a multicast packet is received is the interface that the router will use as an outgoing interface to reach the source. In order to understand this, the router looks up the source address in the unicast routing table .So, the router will only receive packets on the interface that it believes is the most efficient path back to the source. All packets received on the proper interface are forwarded on all outgoing interfaces. All others are discarded. This step is given with an example in Figure 2.3.

[1] The picture is taken from Cisco IP Multicast Training Materials

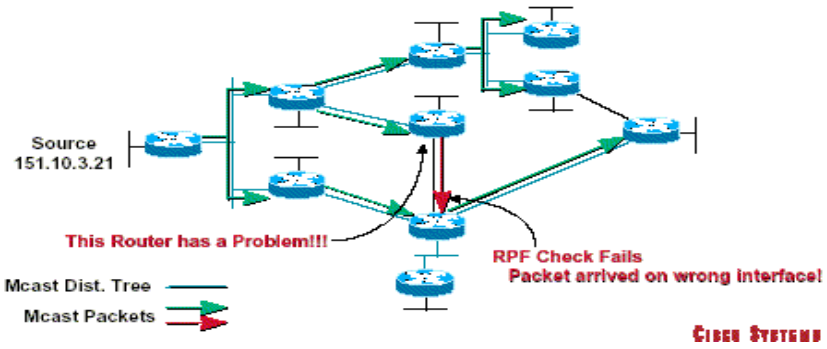


Figure 2.3 RPF Check[2]

iii) If a router has outgoing interfaces that are local networks, these routers are called *leaf routers*. A leaf router will check to see if it knows of any group members on its local interfaces. A router discovers the existence of group members by periodically issuing Internet Group Management Protocol (IGMP) queries. If there are members, the leaf router forwards the multicast packet on the subnet. Otherwise, the leaf router will send a *prune message* toward the source on RPF interface.

iiii) Prune packets are forwarded back toward the source, and routers along the way create prune state for the interface on which the prune message is received. If prune messages are received on all interfaces except the RPF interface, the router will send a prune message of its own toward the source.

As a result of the steps described above, the reverse shortest path trees are formed.

2.2.3 Steiner Tree (ST)

Unlike shortest path tree algorithms which guarantees packet delivery with minimum delay but by using high network resources, the Steiner trees [1-5] tries to minimize minimum usage of network resources but with a higher delay than shortest path tree algorithms. In ST algorithms, we try to find a tree that spans all the multicast group members such that the total cost, which is the sum of the costs of all edges forming the tree, is minimum.

But Steiner Tree Problems are NP-Complete. There is not a polynomial function that gives us the amount of time to calculate it. Due to the difficulties in its computational time, ST has little importance [1, 4]. Moreover, as the structure of a Steiner tree changes with a node join/leave, they are unstable [4].

[2] The picture is taken from Cisco IP Multicast Training Materials

Let M be the set of Multicast nodes in the graph G. In the following cases, the Steiner Tree problem can be solved in polynomial time [1];

1) |M| = 2 (Unicast Case) : If there are only 2 nodes in the multicast group, Steiner Tree Problem becomes the Shortest Tree Problem and can be solved in polynomial time.

2) |M| = |V| (Broadcast Case): As all the nodes in the graph are in the multicast group, we can say that STP for |M| = |V| is a broadcasting problem. And by using Minimum Spanning Tree Algorithms such as Prim's Algorithm and Kruskal's Algorithm, the STP can be solved in polynomial time.

There are also some other cases which states some restrictions about the structure of the graph to solve STP in polynomial time. But these restrictions can not be applied when the number of nodes in the multicast group is very very fewer than the total number of nodes in the tree or the graph is sparse. However, some approximation algorithms gives a performance guarantee. Performance guarantee is a numeric number, ß, which means that the algorithm is ß times costlier than the optimal solution. Some of these algorithms are explained in the following sections of the paper.

2.2.4 KMB Tree

It is a cost optimization algorithm for Steiner trees which was proposed by Kou, Markowsky and Berman [1]. This algorithm assumes that the network has symmetric links. It has five steps which are explained below [1] :

i) A closure graph is constructed that includes only the nodes in the multicast group where the cost of the edges between each group member is the shortest paths.

ii) The minimum spanning tree of the graph constructed in step1 is found; T1.

iii) A sub graph of the original graph T2 is constructed by replacing each edge in T1 by its corresponding shortest path in the original graph.

iv) The minimum spanning tree of T2 found in step3 is found.

v) Finally, the tree is constructed by deleting edges in T2 (if necessary) so that a tree that does not have any closed loops and contains only the nodes in the multicast group can be formed.

The worst case time complexity of KMB tree is O(|S| * |V| * |V|). Its cost is no more than 2(1 - 1/l) * optimal cost where l is the number of leaves in the Steiner tree. An example of the construction of KMB trees is given in figure 2.4.

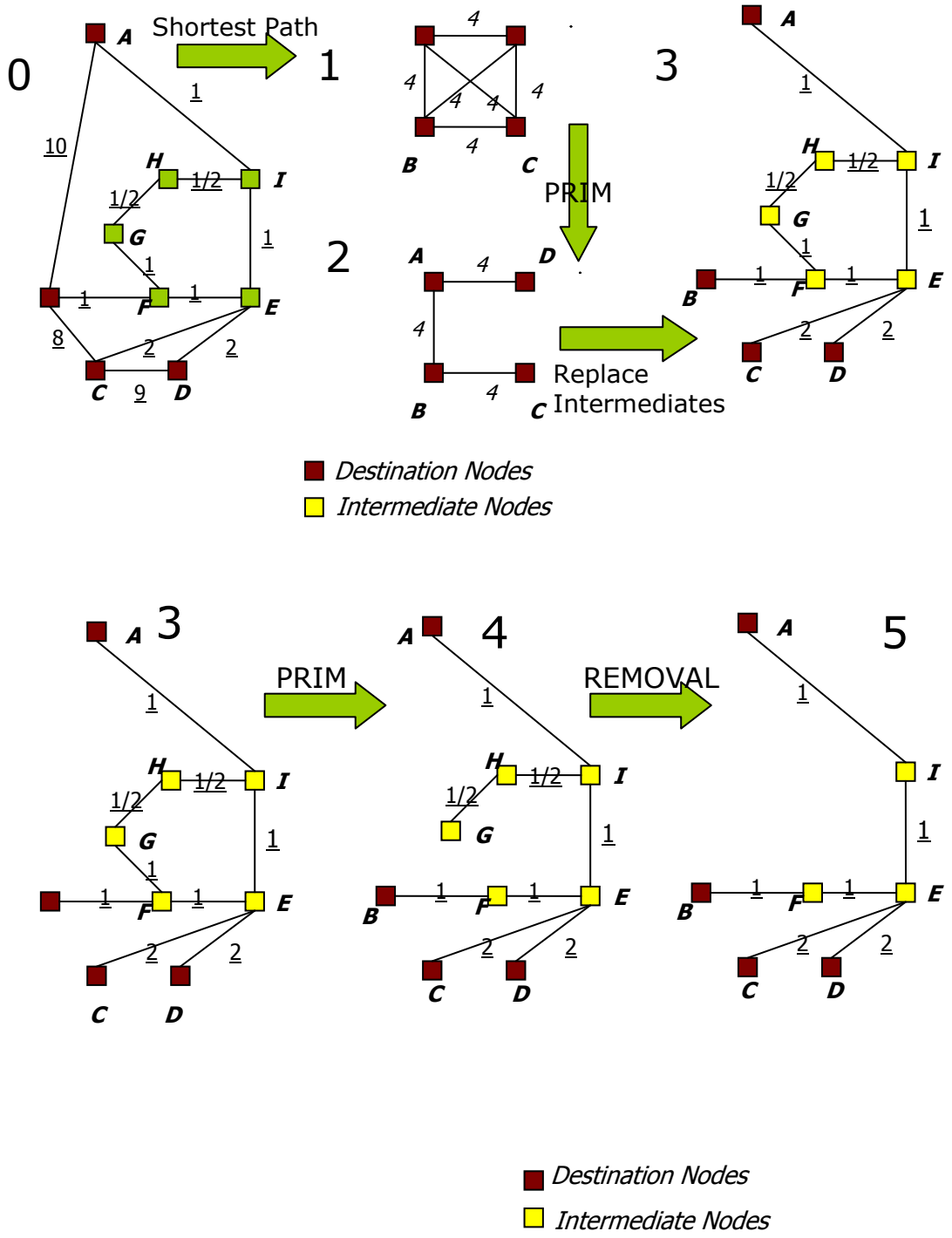


Figure 2.4 Construction of the KMB Tree

2.2.5 Source-Rooted Directed Steiner Tree (DST)

The problem of finding a source-specific multicast tree which uses unidirectional links is modeled by using a source-rooted directed Steiner Tree[1]. DST is minimum-cost directed tree, rooted at source (does not have any input link, in-degree of source is 1) and it contains the multicast group nodes (nodes have only one input and minimum one output) except itself with all links directed from the source. Due to the asymmetry in the in the directed graph, we can not find an approximate solution for this algorithm.

2.2.6 Delay-Bounded Steiner Tree (DBST)

A tree which has minimal network cost under a delay constraint is called a Delay-Bounded Steiner Tree [1]. It is especially useful for multimedia communications. Kumar and Kompella [1] algorithms are the two algorithms that can be given as an example.

In Kumar algorithm, two different routing trees are constructed: a shortest path tree T and a Steiner tree T'. It identifies 'k' destinations for which the difference between the delay in T and the delay in T' is largest. And then replaces the paths to the 'k' destinations in T' with their shortest paths in T.

Kompella algorithm is based on Minimum Spanning Tree heuristic which generates a routing tree starting from source. It selects the nodes to be added to the tree according to two heuristics. First one is cost-delay heuristic which uses a function to convert the cost and the delay of a link into the weight. By this way it tries to minimize the delay while adding some cost. The other heuristic is cost heuristic which is minimum spanning tree algorithm.

2.2.7 Reduced Tree

Reduced trees [1] are proposed in as a solution for scalability of multicasting. The set of vertices in a tree can be partitioned into a set of members (of degree 1), relay nodes (of degree 2) and duplicating nodes (of degree at least 3). A reduced tree is a tree that is modified so that there are no relay nodes. This reduces approximately 80% in the amount of state information that is maintained per group.

2.3 Multicast Routing Protocols

Multicast routing protocols concerns with the efficient transmission of datagrams in multicast communication. The general structure to achieve this is multicast trees. The routing protocols vary according to the way they construct the multicast trees and their support on different IP unicast routing algorithms. We can classify the multicast routing protocols into two types; The *Dense Mode* Multicast Protocols and *Sparse Mode* Multicast Routing Protocols [2, 4].

In a dense environment, members of the multicast group are distributed in such a way that most subnets contain receivers and there is a lot of bandwidth available. In dense mode protocols, in order to construct the distribution tree, flood/prune mechanism is used. Initially multicast data is flooded to the entire network, and then the links that does not have interested receivers are removed as a result of pruning messages to constructed final distribution tree. In a sparse environment, members of the multicast group are distributed across many regions of the Internet and there may not be much bandwidth available. The amount of available bandwidth sets limitations to the routing algorithm compared to dense mode algorithms. Receivers are assumed to be uninterested unless they explicitly ask for joining to the group. In the following sub sections Dense Mode Protocols like DVRMP [1-5], MOSPF [1-5], PIM-DM [1-5], Sparse Mode Protocols like PIM-SM [7] and CBT [1-5], and the border gateway multicast protocol MBGP[1-3] are described briefly.

2.3.1 Dense Mode Routing Protocols

These protocols provide minimum delay but use high network bandwith and require high memory space on multicast routers. They are generally used in inter-domains and referred as “inter-domain protocols”. The most commonly used protocols in this category are explained below in a detailed manner.

2.3.1.1 Distance Vector Multicast Routing Protocol (DVMRP)

This protocol constructs a minimum spanning tree choosing the root as the source and the other receivers in the multicast group as the leaves of the tree. The path between the source and each group member in the tree is the shortest path based on the number of hops named as DVMRP metric between them. DVMRP assumes that

all hosts on the network are multicast receivers initially. The designated router floods the message to all its neighbour routers and they forward the message to their downstream routers by applying RPF (reverse path forwarding) explained in section 2.2.2. In addition to RPF, there is a check to see whether the local router is on the shortest path between its neighbour and the source before forwarding a packet to that neighbour. If this check returns false, the packet is not forwarded to that neighbour router as it is certain that he packet will be dropped then. This enhancement reduces the number of flooding messages in the network considerably.

The prune messages in the network eliminate branches of the tree that do not have any multicast group members. The IGMP [1-3,5], running between hosts and their immediately neighbouring multicast routers, is used to maintain group-membership data in the routers. When a router determines that no host in its subnet belongs to the multicast group, it sends a prune message to its upstream router. And routers update (source, destination group) state information in their tables to reflect which branches have been pruned from the tree. This process continues until all unnecessary branches are deleted from the tree which lastly constructs the minimum spanning tree (As all prune messages have specific timeout value, the flood and prune messages are sent periodically in the network). After all, the minimum spanning tree is constructed and the messages are sent over it. An example of the construction of the minimum spanning tree in DVMRP is given in Figure 2.5 below.

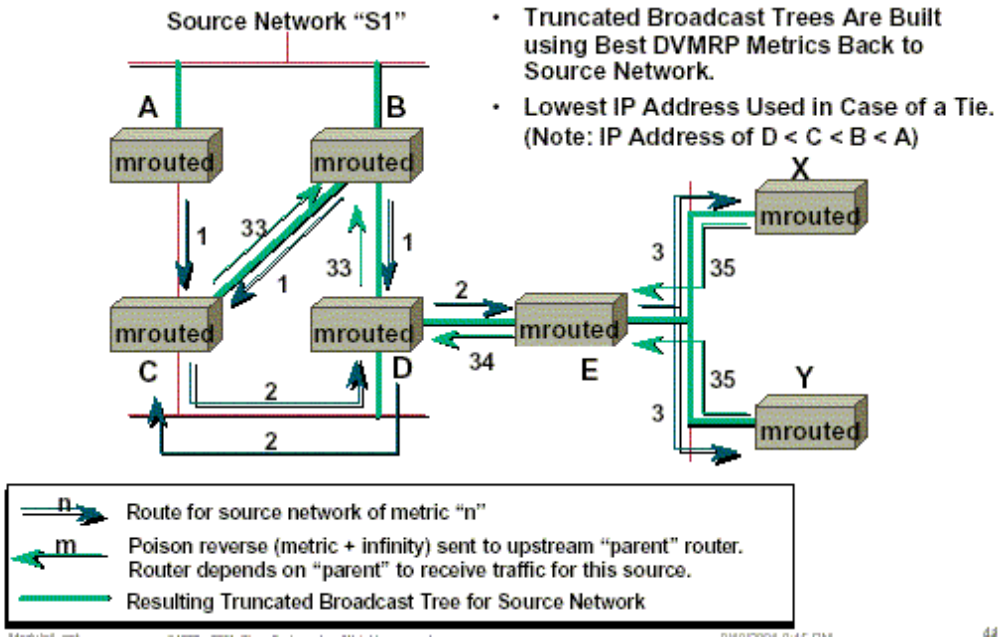


Figure 2.5 DVMRP3

In this example, the DVMRP updates are exchanged for the source S1.Routers A and B advertise its metric (hop count) to its neighbouring routers. When a router determines that the received advertisement is the best to reach the source, it informs the upstream sender by sending back a RP which is the sum the hop count and infinity (32). As a result of these exchanges, the tree is constructed.

As new group members will leave or join the group, the construction of the tree is done by DVMRP periodically. This protocol is useful for multicast groups whose members are distributed densely on the network. As for the groups whose members are sparsely distributed over the network, the periodical broadcast part of the protocol makes it inefficient. In addition to this, the number of each source group and prune-state information require a considerable amount of memory for multicast groups distributed sparsely on different networks.

2.3.1.2 Protocol Independent Multicast - Dense Mode (PIM-DM)

PIM-DM [1, 2, 5] is similar to DVMRP except the fact that PIM-DM works with any unicast protocol, whereas DVMRP relies on specific unicast protocol. Another difference between them is that although both of the protocols use RPF to construct the multicast tree, DVMRP selects the next hop routers that the datagram will be forwarded by looking at its specific tables instead of forwarding it to its all interfaces

[3] The figure is taken from Cisco IP Multicast Training Materials

after RPF check is successful. Therefore, in PIM-DM, packet duplication is likely to occur more than the packet duplication in DVRMP. In order to prevent the transmission of the duplicated packets onto the same multi-access network, an assertion mechanism is used in PIM-DM. The routers send assertion messages to each other in order to determine the winner. The assertion messages contain the administrative distance and the metric to the source. These values forms a single assert value and it is compared by routers to decide who has the best path to the source. The winner sends the datagram to the multi-access network, whereas the loser prunes its interface. It is clearly shown in Figure 2.6 and 2.7 below.

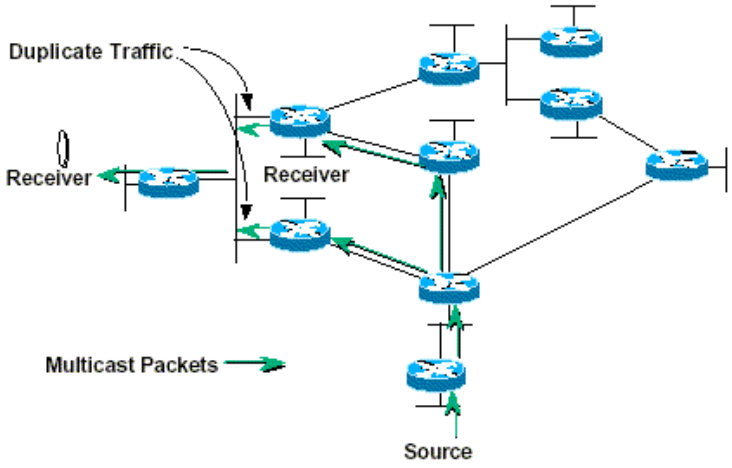


Figure 2.6 PIM-DM tree construction4

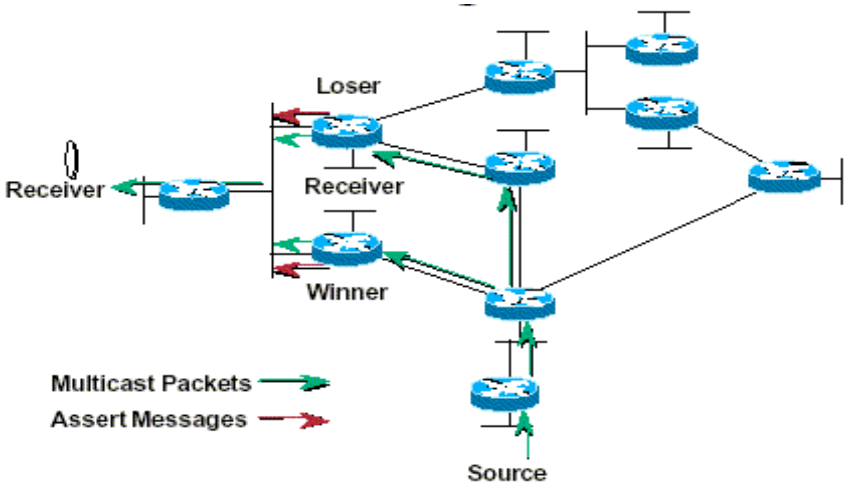


Figure 2.7 PIM-DM assert messages[4]

[4] The figure 2.6 and 2.7 are taken from Cisco IP Multicast Training Materials

2.3.1.3 MOSPF (Multicast Open Shortest Path First)

MOSPF [1-5] is generally used within a single domain in an organization. It uses OSPF as its unicast routing algorithm which routes messages along the least cost(it is calculated by a formula based on hop count, traffic on that link and other network parameters) path. In MOSPF network, each router has a up-to-date knowledge of the entire network topology. They exchange link-state information between each other and as a result, they construct the multicast tree to be used. Each multicast router uses IGMP periodically for multicast group information. The link-state information and the multicast group information is flooded to other routers in the network so that they update their link-state information. As each router knows the entire network topology, it can calculate the least-cost spanning tree with the multicast source as the root of the tree and other multicast group members as the leaves of the tree. As each router knows the same information, the multicast tree they form is the same indeed. MOSPF uses Dijkstra [2] Algorithm to compute the shortest path between source and each multicast group member when it first receives the datagram. This protocol is not scalable as flooding mechanism is used periodically.

After explaining the protocol shortly, multicast transmission of the datagrams in inter-area and inter-domain are explained in a detailed manner below.

In Inter-Area Multicast transmission of the datagram, the information (advertisement) about the existence of group members, which is exchanged between the routers on the network in MOSPF, is called group-membership LSA (Link State Advertisement). These LSAs are flooded on the network periodically. LSAs are flooded within that area such that this information is not flooded to other areas. So, different least-cost spanning trees are formed in different areas independent of each other.

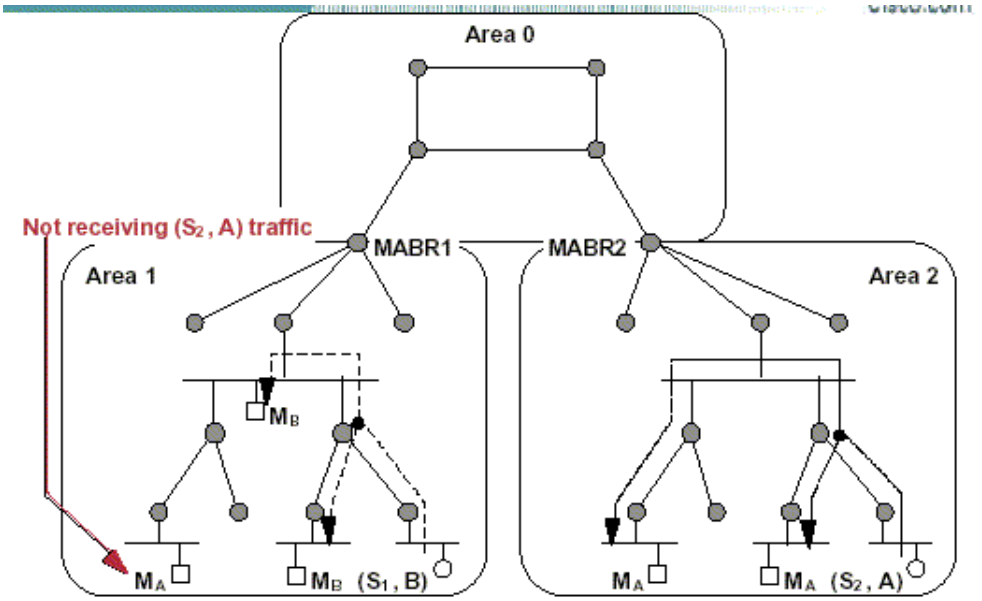


Figure 2.8 Inter-Area Traffic in MOSPF[5]

Let M_A: Member of Multicast Group A, M_b: Member of Multicast Group B, (S_1, B): Source of Multicast Group B, (S2, A): Source of Multicast Group A.

For the given example in figure 2.8; S1, in Area 1, begins to transmit multicast traffic to group B. When the datagram reaches the routers in the area, each router uses Dijkstra to calculate the shortest path tree rooted at the source and transmits the datagram according to the formed tree. This is the same for the routers in Area 2 . But, neither the routers in Area1 nor the ones in Area 2 are aware of the group membership information in the other area. So this traffic between the areas are handled by routers called MABR (MOSPF Area Border Routers). MABRs uses a 'Wildcard Receivers' which set the Wildcard Receiver Flag (*,*) in the router LSAs which is equivalent to wildcard Group Membership LSA. Wildcard Group Membership LSA means that the router has members for every group which are connected to it. By this way, MABR always becomes the part of the multicast tree in the area which means it always gets the multicast datagrams.

[5] The figure 2.8 is taken from Cisco IP Multicast Training Materials

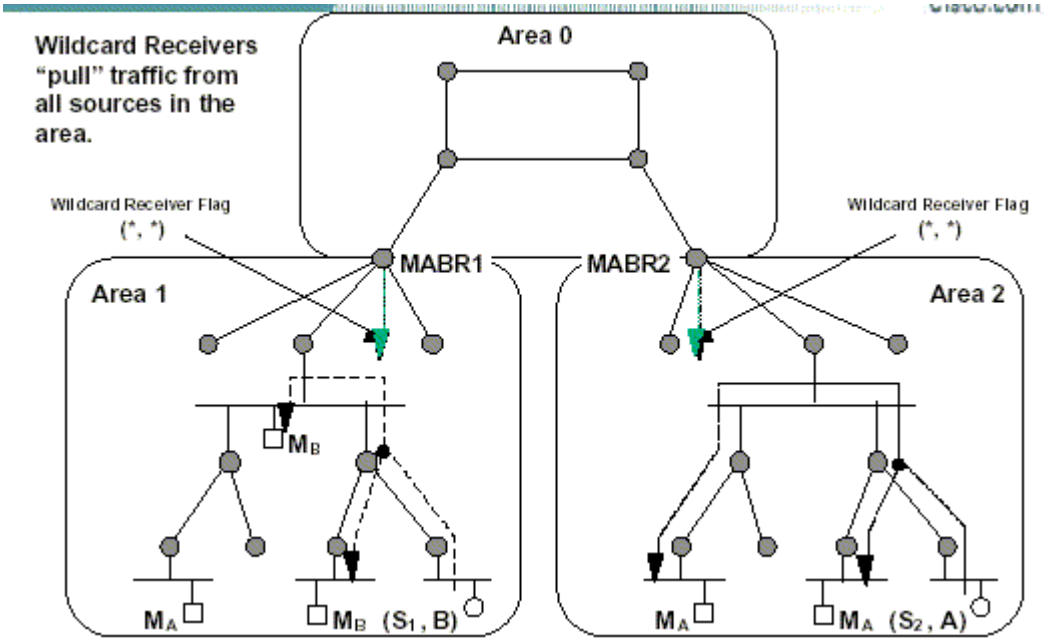


Figure 2.9 Area-Area Connections by MABRs[6]

These MABRs connect the areas to the backbone area allowing the transmission of the datagrams between the areas. The backbone area connecting Area1 and Area2, must know the multicast groups in each area in order to route the traffic between them. For this reason, the MABRs send Summary Membership LSA s to the routers in the backbone area. Finally, the routers in the backbone area use this information to find the least-cost multicast spanning tree between the areas and route the traffic between them.

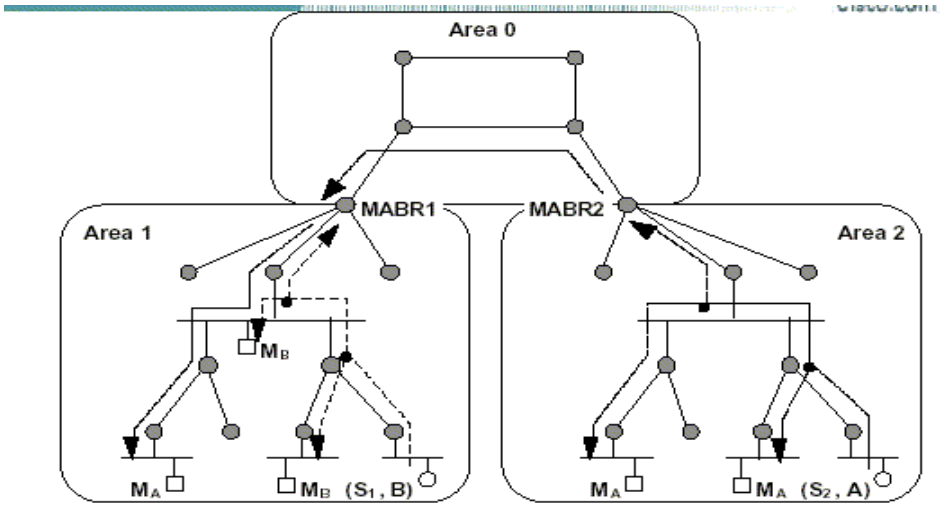


Figure 2.10 Backbone Tree Construction

[6] The figures 2.9 and 2.10 are taken from Cisco IP Multicast Training Materials

Inter-domain multicast traffic in MOSPF is the same as inter-area multicast traffic. When traffic arrives from outside the domain via the Multicast AS (Autonomous System) Border Router (MASBR), this traffic is forwarded through the backbone to the MABRs if necessary (decision is made based on the Summary Membership LSAs that they have sent).

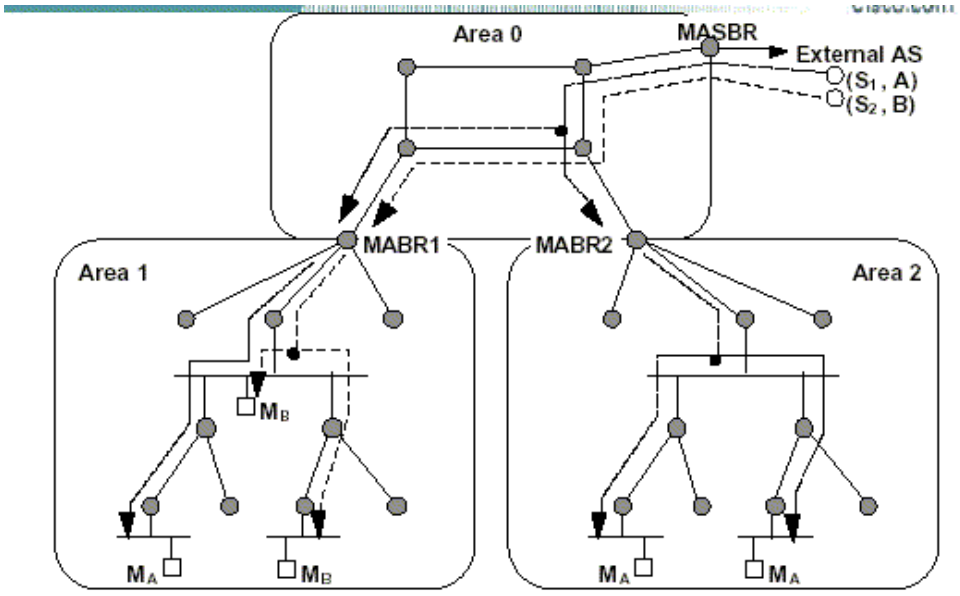


Figure 2.11 Inter-Domain Multicast in MOSPF[7]

2.3.2 Shared Tree Based Routing Protocols

The multicast protocols that use the shared tree as multicast delivery tree are sparse mode protocols in which multicast group members sparsely exist on the network. For the protocols that use shared tree structure, there is a rendezvous (core) point which is the root of the tree. The multicast traffic for the entire group is send and received on the same shared tree over the rendezvous router. There is no periodic flooding as it is in dense-mode protocols. Explicit join mechanism is used to minimize the network traffic such that no host receives the group traffic until specific join. These protocols minimizes the network traffic, on the other hand the path between the source and the receivers may not be the optimal path with minimum delay as it is in dense mode protocols explained above. These protocols reduce the bandwidth usage in the network.

[7] The figure 2.11 is taken from Cisco IP Multicast Training Materials

2.3.2.1 PIM-SM (PIM Sparse Mode)

PIM-SM [7] supports both shared and source based trees. It supports any unicast protocol different from CBT [1-5]. It assumes that no host wants multicast traffic unless the hosts ask for joining to the multicast group from the Rendezvous Point(RP) which is the core router managing the multicast traffic. When a sender wants to join the group, it is registered to RP by its first-hop router and sends multicast messages through the RP to the multicast receivers in that group. Receivers are joined to the shared tree rooted at RP by their designated local routers called DR. The type of the traffic flowing from senders to the RP is unicast traffic. After RP receives the datagram from the source, it uses shared tree to multicast the message to the receivers. The DRs directly connected to receivers may choose to switch to shortest path trees if the delay from source to hosts connected to it exceeds a pre-defined threshold value.

The PIM-SM protocol can be studied under five important topics; the bootstrap router (BSR) and RP selection methods [8], sources' joining to multicast group [7], receivers' joining to multicast group [7], leaving the multicast group [7] and switching to shortest path trees when needed [7]. The details about these topics are described briefly in the following parts of the document.

The Bootstrap router plays an important role in PIM-SM. It informs the multicast PIM routers in the network about which RPs they will use for each multicast group. The candidate RPs, which are configured initially, send special bootstrap messages to bootstrap router periodically in order to show their willingness to be the RP for the multicast groups they want. After receiving the messages from the candidate RPs, the bootstrap router make a decision about which ones are available for being a RP and makes a list of RP for each group to broadcast to the PIM routers in the domain. When the PIM routers in the domain receive the message sent from the bootstrap router, they update their routing tables according to it. The bootstrap router keeps a timer for each of the candidate router and expects a message to be sent to it in that period. Otherwise, it assumes that the candidate router is down and updates its bootstrap message according to this fact. Whenever it receives a message from a candidate RP, it restarts its time for that candidate RP. By this way, all the PIM routers in the multicast domain learn the available RPs for each group and they all

apply the same hash function defined in PIM-SM protocol to choose the available RP.

Another issue of PIM-SM is the join mechanism of the receiver and the source nodes to a multicast group. For a source, there is indeed no separate registration message to join the multicast group. When a source initially wants to join a group, it just sends the multicast message to its first hop router. The designated router (DR) checks whether there is an entry related with the source and the group in his routing table. If an entry is not found, it encapsulates this message in a special format to form a registration message with the data and sends it to the RP by using its unicast protocol. Indeed, it requests from the RP to build a tree back to the source by this way. When RP receives this message, it decapsulates the message and sends it through the shared tree to the receivers and sends a (Source, Group) join message towards the source in order to form a shortest path tree back to the source. So (S, G) state is created on the routers along the shortest path tree. Once the RP receive the data down the shortest path tree from the source, it sends “register stop” message to the source to inform that it can stop sending unicast messages and send it natively.

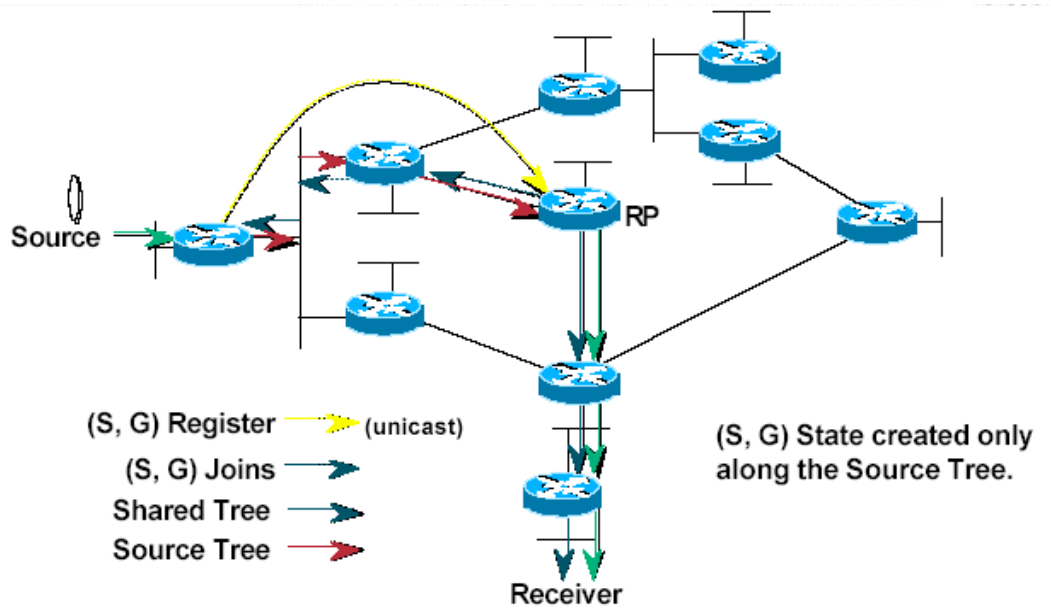


Figure 2.12 the source wants to join to the multicast group8

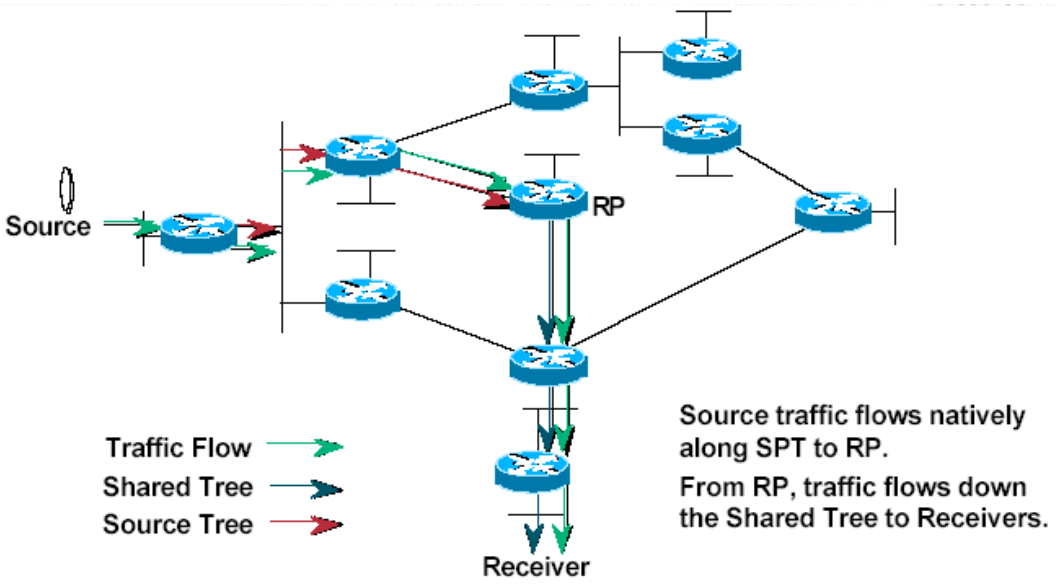


Figure 2.13 After the source joins to the multicast group[8]

After explaining the join mechanism of a source to a multicast group, now let's turn to the problem of the join mechanism of a receiver to a multicast group. When a host wants to be a receiver for a multicast group, it informs its DR (this can be done by IGMP). If its designated router is already on the multicast tree, it does nothing. Otherwise, it sends a join message towards the RP for the multicast group that is wanted to be joined. The message passes from different routers until it is received by the RP or any router having (*,G) entry for the group that is wanted to be joined. As a result, a new branch of a multicast tree is formed.

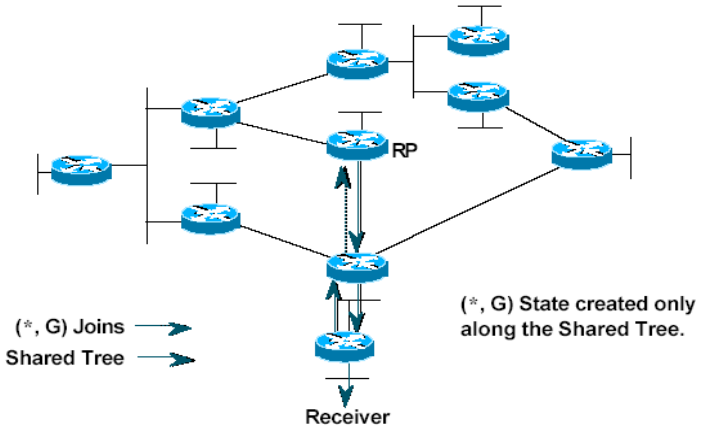


Figure 2.14 After the source joins to the multicast group[8]

[8] The figures 2.12, 2.13,2.14 and 2.15 are taken from Cisco IP Multicast Training Materials

On the other hand, when a host wants to leave a multicast group, it informs its DR. If there is no other host connected to the DR except that host, it sends a prune message to its upstream router. The upstream router does this check again and it continues recursively if necessary until the prune message arrives the RP.

Another issue of PIM-SM is switching to shortest path tree when needed. The routers with directly connected members have the ability to switch to the shortest path tree when the traffic rate is above the threshold.

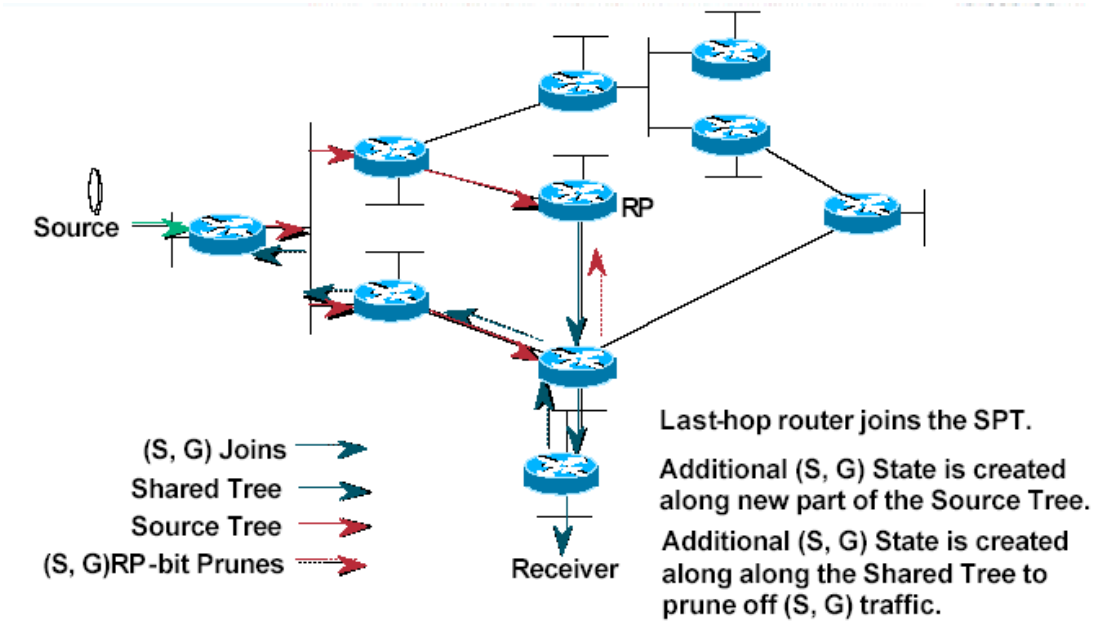


Figure 2.15 Switching to shortest path tree [8]

First of all, the last-hop router which is directly connected to the receiver sends a join message towards the source to join the shortest path tree. The message passes through different routers on the path to the source and forms another branch of shortest path tree. And also the (S, G) state is created on all the routers on the path formed. And lastly, Prune messages are sent to RP along the shared tree to cancel the traffic received through the shared tree. And RP sends prune messages back to the source to cancel the traffic from the source to RP.

2.3.2.2 Core Based Tree (CBT)

CBT is similar to PIM-SM but has some differences. CBT uses bi-directional shared trees. The traffic flows over Core Router. CBT does not switch to shortest path trees like PIM-SM when the traffic rate on the shared tree is above the threshold. If the

first-hop router to the source is on the tree, the packet is forwarded to all branches of the tree. Otherwise, the packet is sent to the core router and then to the receivers along the shared tree. Reduced amount of multicast state is stored in routers in comparison with the amount of state information stored in source based trees.

2.3.2.3 Border Gateway Multicast Protocol (BGMP)

Border Routers use BGMP protocol to allow multicast traffic between different ASes (Autonomous System). It consists of two parts; MIGP and BGMP [1-5]. MIGP (Multicast Interior Gateway Protocol) is used within the AS and BGMP is used to construct a bidirectional center-based tree with other routers in different ASes. The nodes in the centered can be thought as ASes and the core node is an AS as well. BGMP uses TCP as its transport protocol. Border routers setup TCP connection between themselves and exchange BGMP messages. When group membership changes in ASes, the border routers send incremental join/prune messages to another one. BGMP also allows shortest path tree construction when needed. The multicasting across ASes is similar to the PIM-SM. The difference is that, in BGMP ASes are used instead of routers when modeling the multicast traffic.

3. RP SELECTION AND RELOCATION ALGORITHMS IN MULTICAST NETWORKS

Rendezvous Point is chosen administratively in center-rooted protocols like CBT and PIM-SM before multicast session starts. The RP of the group is dynamically changed only when the RP stops operating. Sources or receivers may join or leave the multicast group frequently for some multicast applications like network gaming. After a time, the quality of the tree may decrease a lot and may not provide minimal QoS requirements of the group participants due to the location of the rendezvous point. This is a normal situation as the administrators may not guess where the participants of the multicast groups will locate correctly in a very dynamic multicast application. In order to increase the quality of the service, we need a dynamic rendezvous point relocation mechanism adjusted to work with existing center-based multicast protocols or embedded in a new multicast protocol.

The RP Relocation process can be mainly divided into two sub processes; the process of finding a better RP and the process of migration to the new RP.

The process of finding a better RP requires the estimation of the costs of the candidate RPs in terms of delay and bandwidth when the tree is rooted at that RP. A triggering mechanism, either central or distributed, is needed to inform the set of nodes to make their own cost calculations. The informed set of nodes may change as to the algorithm implementing RP selection. This set of nodes calculates their cost by using the pre-defined cost function in the algorithm. There are various cost functions that can be used to make a calculation. A cost function may require a distributed communication between the nodes to get a result or it may use the values in existing routing tables from the unicast protocol that is already used. To find a better RP as a result of RP selection process, the cost function should consider all the input data (the hash routing tables in its unicast protocol, the topological knowledge that it can get from the multicast protocol used etc.) that it knows from its unicast and multicast protocol. After the evaluation of the costs, communication among the nodes and the current RP must occur to compare their cost values in order to choose a better RP.

As the migrating from the current RP to the new RP, if a node with a less cost value is found, may be an expensive operation, the migration process should not occur frequently. A threshold value may be used to compare the improvement in the cost of the new RP with the current RP. If the percentage of the improvement is above the threshold, migration may occur. Such a threshold mechanism decreases the number of migration number, thus, prevents excessive amount of bandwidth usage due to RP migration.

After an appropriate node is found, the migration process takes places. The receivers and senders of the group should be informed about the new RP. If needed, some other special nodes may be informed as well like BSR in PIM-SM. It is important not to use a lot of resource in terms of network bandwidth during migration process. Another important point is that the packets flowing from senders to receivers of the multicast group may get lost due to migration process. Therefore, the migration process should minimize the packet loss or prevents it if possible.

In the following sub sections, detailed information about the weight functions, RP/Candidate RP selection algorithms and lastly migration algorithms will be explained.

3.1 Weight Functions

Weight functions play an important role in the RP selection process. The nodes calculate their cost by using weight functions. A RP selection algorithm may use more than one weight function at the same time and may combine the results with a mathematical function to get a better single cost value. But this operation may cause extra complexity in terms of message and processing time.

For the existing center-based routing protocols, the RP knows only the sources that it is working with. In other words, it does not have any information about the location of the receivers. But, in general, we can define a set S which is either the source nodes or the receiver nodes or both of them. In addition to this, the abbreviation “C-R” will be used for the candidate router

3.1.1 Actual Cost of the Tree

The actual cost of the tree is calculated by taking the sum of the links in the tree rooted at C-R and extending all the nodes in set S [9,10]. For simplicity, unicast routing tables may be used assuming that there is no shared link between C-R and each of the node in set S. Therefore, the sum of the costs between the C-R and each of the nodes in S may be added directly.

3.1.2 Maximum Distance

It is calculated by taking the maximum of the distances between C-R and each of the nodes in S [9, 10]. The term distance between node u and node r refers to the cost of the shortest path between the node u and node r.

3.1.3 Average Distance

It is calculated by taking the average of the distances between C-R and each of the nodes in S [9, 10]. The term distance used here has the same meaning with the term distance used in Section 3.1.2.

3.1.4 Maximum Diameter

It is calculated by taking the sum of the two highest distance value calculated between C-R and each of the nodes in the set S [9-10].

3.1.5 Delay Variance

The delays (distance) between C-R and each of the nodes in S are calculated. Then the difference between the maximum delay among the calculated delays and the minimum delay among the calculated delays is taken which is called as Delay Variance [9-10].

3.1.6 Estimated Cost

This cost function is firstly proposed in [10] by D.G Thaler and C.V Ravishankar. It takes the average of the estimated minimum cost and the estimated maximum cost which are also proposed in [10]. First of all, it calculates the distances between C-RP and each of the nodes in S.

For any tree, if each distance between the root and the nodes in the tree is known and all the distance values are different from each other, for the best case scenario the cost of the tree is the maximum distance value between the calculated values. The best case scenario occurs when the tree is linear with node degree 1 and the nodes are placed one by one with links connecting them having different link values. For the worst case scenario, the root may have a node degree greater or equal to n, which is the number of nodes in the tree. Thus, the cost of the tree for the worst case scenario will be the sum of the each distance values calculated at the beginning. In other words, the tree will branch at level 0 to form a maximum number of branches and will never branch again.

The best and the worst-case tree costs for the tree with distance values, each of which is different from the other, are explained above. If we have duplicate distance values, a linear tree with node degree 1 can not be formed. It means there will be at least one shared link and a branch in the tree. As there may be duplicate distance values, the generalized form of the estimated minimum cost and estimated maximum cost is given below.

$$Est\ Cost_{min} = \max_{u \hat{\imath} S} d(root,u) + \text{ number of duplicate distance nodes in S} \quad (3.1)$$

$$Est\ Cost_{max} = \begin{cases} \sum_{u \in S} d(root,u), & if\ |S| < \deg(root) \\ \sum_{u \in S} d(root,u) - (|S| - \deg(root)), & otherwise \end{cases} \quad (3.2)$$

$$Est\ Cost = \frac{Est\ Cost_{min} + Est\ Cost_{max}}{2} \quad (3.2)$$

In [10], the estimated cost function which takes the average of estimated cost minimum and estimated cost maximum is claimed give better results than the other 5 weight functions described in sections 3.1.1, 3.1.2, 3.1.3, 3.1.4 and 3.1.5.

3.2 RP / C-RP Selection Algorithms

The selection algorithms can be divided into 2 sub categories; the RP selection algorithms and candidate RP (C-RP) selection algorithms which will be presented briefly in the following two sections.

3.2.1 C-RP Selection Algorithms

C-RP selection algorithms are static algorithms. The selection of the candidate rendezvous points take place initially before multicast sessions are created. After the candidate-RPs are selected, and the routers in the network are configured according to the C-RP selection results, the multicast network becomes ready to initiate any multicast session.

3.2.1.1 K-Maximum Path Count

In K-Maximum Path Count algorithm [16], the shortest paths for all pairs of nodes in the network are found. After that, the repeated nodes on the shortest paths are counted. As a result, K nodes (desired number of C-RPs) with the most common usage are selected among the nodes in the network. It is claimed in [16] that the decrease in delay is about %17 in comparison with random C-RP selection scheme.

3.2.1.2 K-Maximum Degree Method

In K-Maximum Degree Method [16], the nodes are sorted in descending order according to their node degree. And top K nodes (desired number of C-RPs) are selected as C-RP. It is claimed in [16] that the decrease in delay is about %15 in comparison with random C-RP selection scheme.

3.2.1.3 K-Minimum Average Distance Method

In K-Minimum Average Distance Method [16], the average distances from each node to all others are calculated. Then the nodes are sorted in descending order according to their average distance values. And top K nodes (desired number of C-RPs) are selected as C-RP. It is claimed in [16] that the decrease in delay is about %11 in comparison with random C-RP selection scheme.

There is a different proposal in order to select C-RPs in [9]. The internet is modeled as transit-stub graphs [9].It does not select a lot of candidates initially in order to decrease the complexity due to RP selection. It suggests choosing the ones that is likely to perform better than the current one. It selects a candidate node from each transit domain and one from the stub domain with the most representation of sources and one from the stub domain with the most representation of receivers. If the number of transit domains is high, it selects randomly among transit domains instead

of selecting one node from each one in order to decrease the number of candidate rendezvous points. As a result of many simulations made in [9], the proposed selection method is claimed to give better results than the random C-RP selection scheme.

3.2.2 RP Selection Algorithms

RP selection algorithms can be classified into two main categories; the static RP selection algorithms and dynamic RP selection algorithms. Dynamic algorithms give better results than the static ones and give response to the changes in the network which can not be estimated in Static algorithms.

In [9], the worst-case RP selection method is studied and the performance of the worst-case scenario is compared with the performance of the shortest path tree constructed from a given graph. The method is easy to implement, requires no knowledge of the network topology. The selection is done among the members of the multicast group. It is observed that the delay of the tree is 2.4 times worse than that of the shortest path. When the location of the core is not so important, this method can be used.

Moreover, in [9], random RP selection method is simulated. It is observed to give better results than the worst-case scenario. But the delay variance is very high as random selection is made. Therefore, it can not be used in multicast applications which are not tolerable to high variance in delay.

Another method studied in [9] is called "topology-based center selection". The topologic center of a graph is the node that minimizes the depth of the tree when it is chosen as root. In [9], a threshold value, T, is used to select nodes which enables the depth of tree be between the topological center depth and T worse than the topological center depth. When T is increased, the delay also increases. But it is observed to give better results than the worst-case and random-selection scenario. Its disadvantage is that we need to know the network topology in order to make any calculation.

In [9], topological center location method is improved, and group-based center location is proposed. Not only the network topology but also information about group members is needed to be known in this method. It gives better results when the receivers of the group are highly localized. It applies 3 selection types; first one is

random selection among receivers of the multicast group. It is observed to obtain a better performance than choosing the center of the whole graph. The second one is choosing the RP among the topological center of the graph formed by the receivers. This is appropriate when the receivers are highly localized and gives better performance than the first one. The last and the third one is selection of the RP as the receiver having best delay/bandwidth value. It is certain that the last one gives better performance than the preceding two of them. In [9], different selection methods for different multicast scenarios are simulated and the results are compared with that of shortest path tree. The work done in [9] gives information about how to select the RP with less/more/none knowledge of network topology or group information.

3.2.2.1 Minimal-Member Protocol (MIN-MEMB)

It is proposed in [10]. The RP is chosen among the members of the group. All group members are taken into account when making calculations. The pseudo code of the algorithm is given in Figure 3.1.

```
S1:
    case "center" :
              Start a timer T1 (frequency of the algorithm)
               Wait until it expires.
               RPCost= Calculate_Cost();
               Multicast(RPCost , Group_Member_List,Group_Source_List);
               Start a timer T2;
               While(! T3 elapsed)
                  ReceiveMessages= Receive_Replies();
               SetCenter(Best(ReceiveMessages));
               Goto S1:

    case "! RP"
            if(Receive_Message())
                if (! IsGroupMember_Or_Source(current))
                    Cost= Calculate_Cost();
                    Start a timer T3;
                    While(! T3 elapsed)
                        Received_Messages= Receive_Replies();
                        if(Cost < nth Lowest Value in Received_Messages)
                            Multicast(Cost)
```

Figure 3.1 The pseudo code of the MIN-MEMB

Every group member calculates its own weight and waits for a random amount of time to receive other nodes' costs. After its timer expires, it compares its cost with the nth lowest cost it has received. If its cost is better than the nth lowest cost, it multicasts its cost to the group. After the timer of the current RP expires, it selects the best node, if exists, as the RP according to its cost and the costs that it has heard.

For *n* best nodes, *m* number of group members; let's select the best node, *n=1* ;

The best case: *1* message is sent, no other received. For the worst case, everyone can multicast message to others, *m* messages. So the number of messages flowing in the network is between 1 and m. The average number of messages will be

(1+2+...+m) /m=(m (m+1)/2) /m = (m+1)/2. So, the complexity of the algorithm is O(m).

3.2.2.2 Hill-Climbing Protocol (HILLCLIMB)

It is an algorithm proposed in [10]. It uses probing. It holds a path list to hold the list of nodes with better weights while traversing towards to neighbours. The current center starts probing and queries its neighbours. It sends them the list of the source/group members. Each neighbour node calculates its own weight and responds to the probing node. The probing node compares its cost value with the nth lowest cost value in the path list. If the probing node's weight is lower than the nth lowest cost value in the list or the list is full, the last probing node is chosen as the new RP and it informs the current RP about it. Otherwise, it will add itself to the list of visited nodes and the search will continue with the next unvisited neighbour node.

The hill climbing algorithm has a better complexity than the MIN-MEMB protocol defined in section 3.2.2.1 [10]. It uses all the network nodes to find a better RP instead of using just the group members as it is in MIN-MEMB. Therefore, the location of the new RP is likely to be located in the same vicinity of the current RP. It is observed from the simulations that when the set of S is chosen as the set of all receivers and sources, it gives better results than the one with the set of S chosen as the set of sources. This is an expected result. If more nodes are taken into consideration while calculating the weight function, better results will be obtained.

3.2.2.3 Scalable Core Migration Protocol (SCMP)

The scalable core migration protocol is proposed by Ting-Yuan Wang, Lih-Chyau Wuu and Shing-Tsaan Huang in [11]. In general, it divides the tree in to sub trees whose depth is bounded with a value. Each root of the sub tree is called as agent. Agents know about the nodes in their sub trees and the RP knows about the agents in the tree. When the depth of the tree (i.e. delay) exceeds the boundary, the part of the tree is merged into more than one sub trees. When the sum of its depth and its neighbour sub tree's depth is smaller than the boundary value, then these sub trees are merged. Each agent calculates its cost and informs other about it. So, the best one with the lowest cost is selected finally. When the agents calculate their cost values, they use average Delay weight function to calculate the average delay between itself and each of the agents. The algorithm, more generally, tries to balance the multicast tree periodically. New RP is chosen among the agents of the multicast group. So, there is no need to define a migration algorithm for the members of the group. Little work is done in comparison with the previous protocols described in Section 3.2.2.1 and 3.2.2.2.

3.2.3 RP Migration Algorithms

The quality of the migration algorithm can be measured with the number of packets lost during migration process and its extra traffic cost due to migration. Generally, it can be said that more network resources (more control messages) are consumed in order to prevent packet loss better. So there is a trade-off between them. The RP migration algorithms may be chosen as to the type of the multicast application.

3.2.3.1 Simple Approach

The packet loss is not taken into consideration in this approach [12, 17]. The migration message including the group address, the old RP and the new RP is sent by the old RP to the multicast group. The nodes receiving the message just deletes the related state from its routing table, send prune messages towards the old RP and send join message towards the new RP. The approach is simple; it does not do any check or apply a process to reduce the packet loss. This algorithm may be used in multicast applications where packet loss is not so important.

3.2.3.2 Independent Trees

It uses bottom-up approach while pruning. The migration message sent by the old RP flows to the leaves of the tree. The prune and join messages starts to be sent from the bottom to the root of the tree. And also a channel is created between the old RP and the new RP to send the packets to receivers in both trees during migration. This process reduces packet loss but does not prevent it completely [17].

3.2.3.3 No Packet Loss

This approach [17] is similar to Independent Trees approach. Both of them use bottom-up approach. Its difference from the method in Section 3.2.3.2 is that the pruning is not done until all members of the node are attached to the new RP. In order to understand this, extra control messages are used between the members and the node.

4. THE SIMULATION

The simulation aims to compare the performance improvements and message overheads of the Rendezvous Point Relocation algorithms. We compared PIM-SM [7] protocol, in which the Rendezvous Point does not change unless it is down, and SCMP [11], in which the Rendezvous Point changes dynamically when there are changes (joins or leavings) in the multicast group , in terms of different metrics which are explained in the following sections.

The simulation mainly consists of 5 stages. The first stage is the generation of different type of networks, on which the simulations are run, described in Section 4.2. The second stage is the generation of different multicast traffic scenarios, which determines the characteristics of the multicast applications in real life, described in section 4.3. The third and the most difficult stage is the implementation of the multicast protocols compared in the simulation whose detailed explanations can be found in [7], [11]. The assumptions and the details about their implementation are explained in Section 4.4. The fourth and the last stage is the analysis stage in which the data, obtained as a result of the simulation, is used to make conclusions. This stage is explained in a detailed manner in section .

4.1 SIMULATION ENVIRONMENT

We used various operating systems and various software development environments during the simulation.

In stage 1, explained above, We used the network simulator (ns) [18] and gt-itm network generator tool and some other conversion tools [19] to generate different type of networks on Linux Fedora operating system. We wrote a C++ program to automate the creation of different networks by using these tools in KDE on Linux Fedora.

In stage 2, We wrote a windows application with GUI to generate multicast traffic files in Microsoft Windows C#.NET on Windows Server 2003 Enterprise Edition.

In stage 3,We implemented the protocols used in the simulation (PIM-SM [7], SCMP [11]) in Microsoft C#.NET and executed them on Microsoft Windows 2003 Server with Intel Pentium 2.4GHz Microprocessor installed on the computer.

Lastly, some of the analyses are made with a program written in Microsoft C#.NET on Windows Server 2003 Enterprise Edition. The implementation details of the stages are explained below.

4.2 NETWORK TOPOLOGIES

Weused different network topologies in the simulation in order to get more accurate results.We used the Georgia Tech's Internetwork Topology generator [19] tool to generate different kinds of networks.We created mainly 2 types of graphs; the flat random graphs and the transit-stub graphs.

For both of these graph types, We used Waxman Probability function [20] to generate edges in the network. The probability that an edge exists between any two nodes *u* and *v* is given by the following probability function:

$$P(\{u,v\}) = a\,e^{\frac{-d(u,v)}{Lb}} \tag{4.1}$$

where *d(u ,v)* is the distance between any two nodes, L is the maximum possible distance and α, β are the parameters between 0 and 1. Larger values of α values increase the average node degree of the network, while larger values of β increases the ratio of longer edges to shorter edges.

We chose α values as 0.5 and 1 and β values as 0.5, 0.5 for both of the network types. In other words, We created different graphs by increasing α to see the effect of the increase in average node degree to multicast group communication for both flat random and transit-stub graphs. The number of nodes in the network is chosen as 50 and 100 in order to see the effect of the increase in the number of nodes to multicast tree cost and the number of generated messages in the network. As a result 8 different networks are created for the simulation. We created 10 graphs for each of these network types. So, total of 80 graphs are generated to see the effects of different parameters to multicast group communication.

4.2.1 Network Analysis

We calculated the average node degree, the number of links, the network diameter and the maximum and the minimum node degree for each graph generated. We generated Flat Random Graphs and Transit-Stub graphs. The edges are created between any two nodes for each graph by using the probability function explained above.

Transit stub graphs include transit graphs which represent the internet backbone and the stub graphs, whose nodes are connected to the nodes in the transit graphs, which represent the WAN in the internet. Any part of the shortest path between any nodes in the stub graph does not pass outside the stub graph. In other words, the data packets created as a result of the communication between any two nodes in the stub graph are not forwarded to the nodes outside the stub graph. In addition to this rule, the shortest path between two nodes in different stub networks passes through the initial stub network, the transit networks between them and the destination stub network but not any another stub network.

On the other hand, Flat random graphs are created randomly, by using a two-dimensional plane. Any two nodes are selected from the plane and the edges between them are created with the probability explained above. Therefore, the transit stub graphs are known to model the internet more efficiently than the flat random graphs.

In the simulation, We generated transit-stub graphs and flat random graphs with 50 and 100 nodes which has different characteristics explained detailedly in the following sub sections. For transit-stub graphs, the the genration of the graphs differs depending on the number of nodes which is selected as 50 and 100.

For 50-node transit-stub graphs, one transit domain with five nodes and 5 stub domains, with 9 nodes in each of them, which are connected to each node in the transit domain are created(5 + 5*9 = 50).The nodes in transit and stub domains are chosen from a 2-dimensional planar domain with Waxman2 probability function. The size of the dimensions vary from 10 to 15 for 50-node transit-stub graphs.

For 100-node transit stub graphs, two transit domain with 5 nodes in each of them and 10 stub domains with 9 nodes in each of them with a total number of (5 + 5 + 10*9=100) nodes are created. Each node in the transit graphs is connected to one stub domain.The stub domains and the transit domains are created similar to the 50-

node transit-stub graphs except the size of the dimensions of the planar domain. The size of the dimensions are chosen to be 30 for 100-node transit-stub domains. So the graphs in 100-node transit-stub graphs are chosen from an area which is 4-9 times greater than the ones in 50-node transit-stub domains. As a result, the cost of the links between the nodes in the 100-node transit-stub graph are very higher than the the costs for 50-node transit stub graphs.

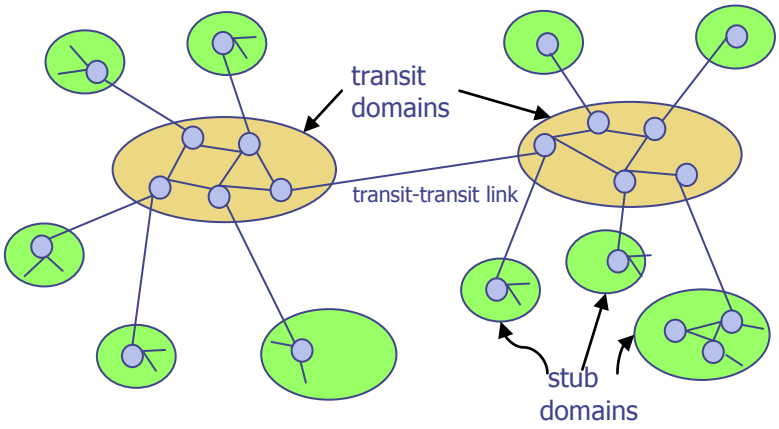


Figure 4.1 100-node transit-stub graph with 2 transit and 10 stub graphs

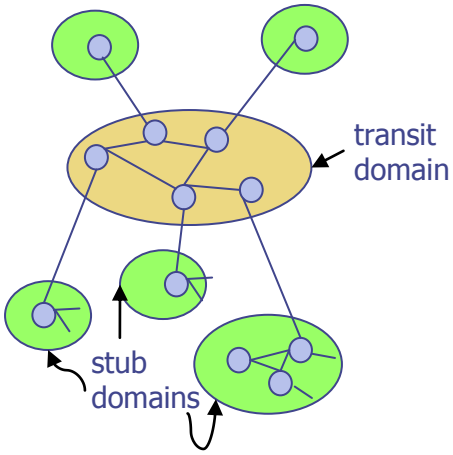


Figure 4.2 50-node transit-stub graph with 1 transit and 5 stub domains

The results and the comments on the generated graphs are explained in detail in the following sub sections by using graphical charts.

4.2.1.1 Transit Stub Graph (α=0.5, β=0.5)

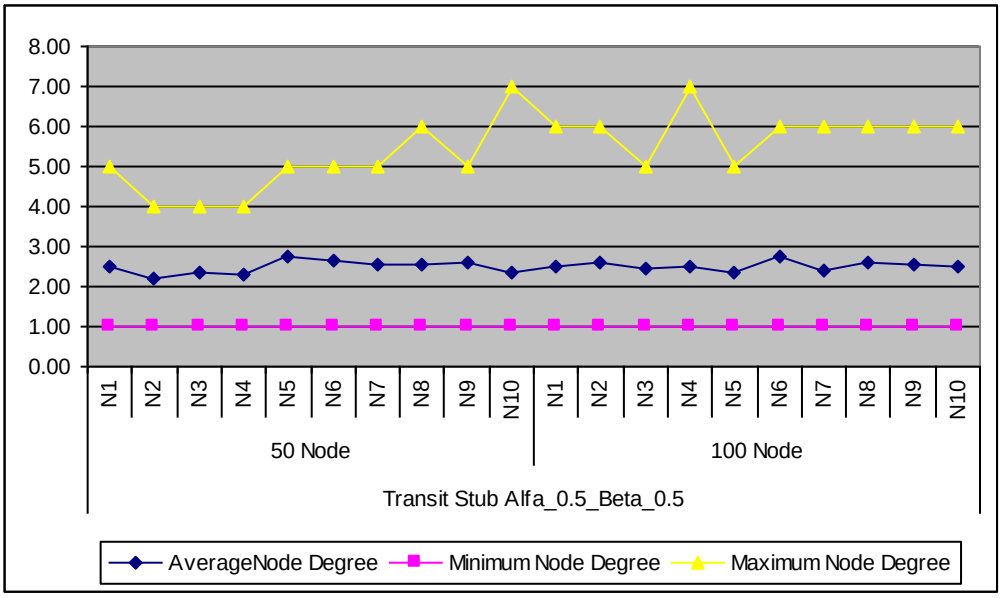


Figure 4.3 the node degree comparison for Transit Stub Graphs (α=0.5, β=0.5)

10 graphs with 50 nodes and 10 graphs with 100 nodes are created by using the parameters α=0.5 and β=0.5. All the networks are compared in terms of average node degree, the minimum node degree and the maximum node degree and the results are shown in Figure 4.3.

From Figure 4.3, it is clearly seen that the average node degree does not change considerably when the size of the network changes. It is an expected result due to the fact that the average node degree changes when the value of α change as explained in [20]. On the other hand, the maximum node degree increases when the size of the network increase. As the number of nodes in the network increase, some nodes may have high number of connections. But on the average, the node degree is nearly constant.

In Figure 4.4, it is observed that the number of links and the network diameter increase when the number of nodes in the network increase which is a normal expected result.

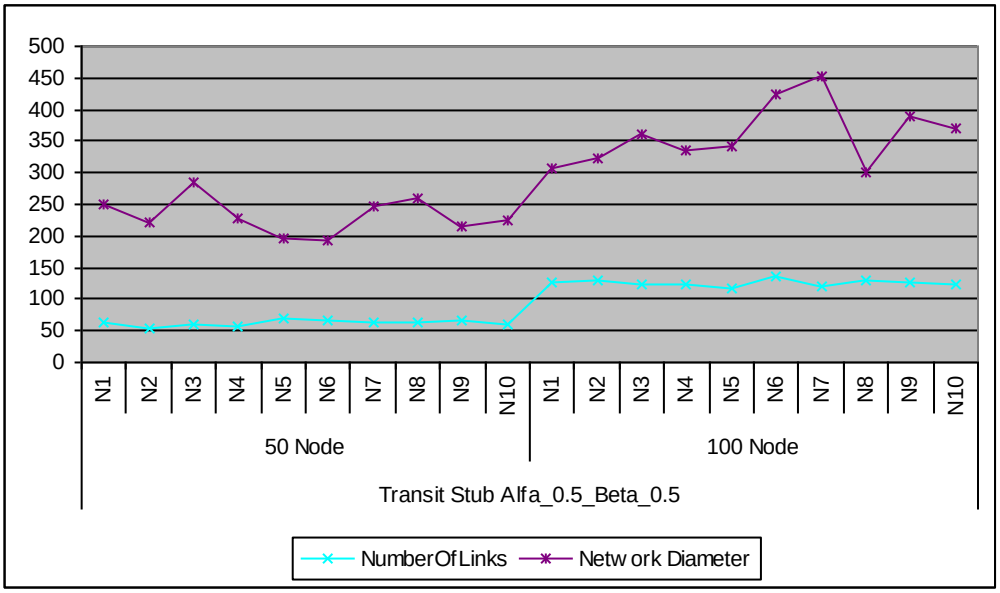


Figure 4.4 the number of link and the network diameter comparison for Transit Stub Graphs (α=0.5, β=0.5)

4.2.1.2 Transit Stub Graph (α=1, β=0.5)

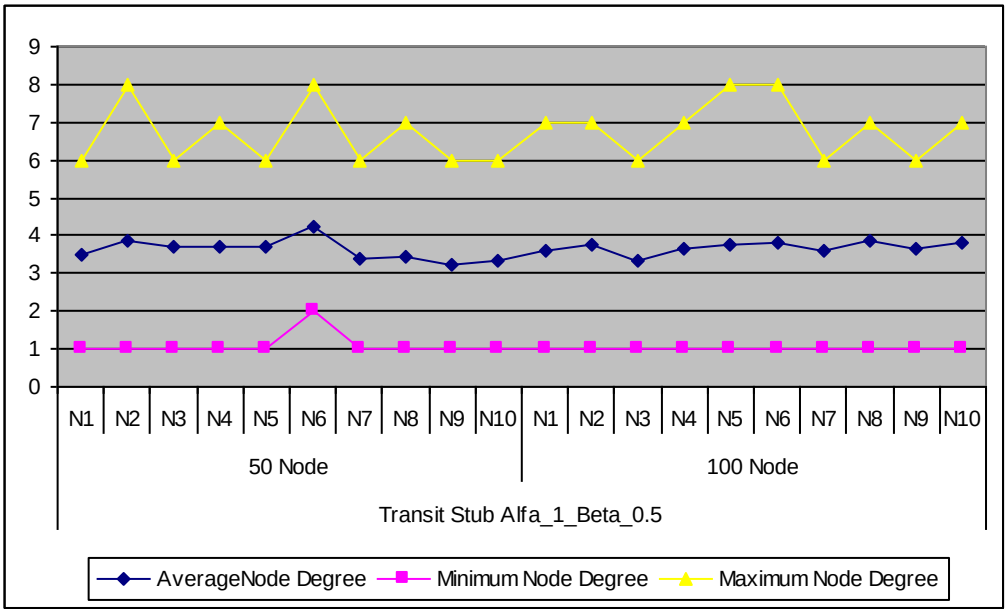


Figure 4.5 the node degree comparison for Transit Stub Graphs (α=1, β=0.5)

10 graphs with 50 nodes and 10 graphs with 100 nodes are created by using the parameters α=1 and β=0.5. All the networks are compared in terms of average node degree, the minimum node degree and the maximum node degree and the results are shown in Figure 4.5.

When the graph in Figure 4.3 and the one in Figure 4.5 is compared, it is seen that the average node degree increases when the value of α increase from 0.5 to 1.0. Moreover, the maximum node degree values in Figure 4.5 are greater than the values in Figure 4.3 which is the result of the increase in the average node degree value. In [20], it is clearly stated that the increase in the value of α increases the average node degree which is also observed when the two figures are compared.

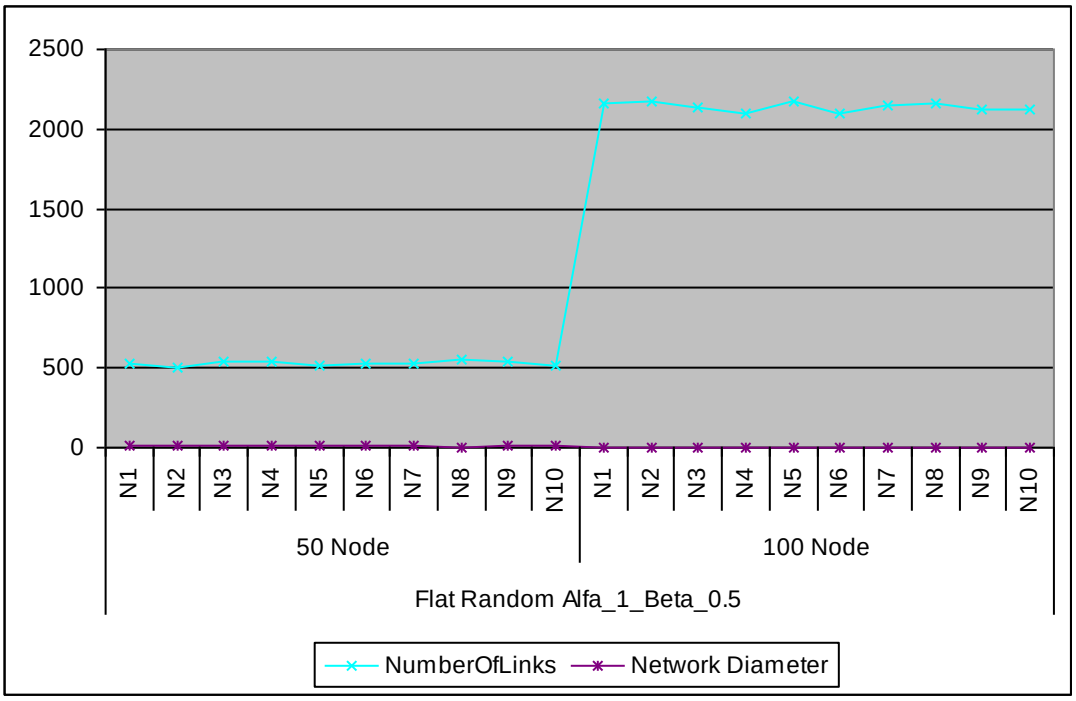


Figure 4.6 the number of link and the network diameter comparison for Transit Stub Graphs (α=1, β=0.5)

In Figure 4.6, we observe that the numbers of links increase when the numbers of nodes in the network increase which is an expected result. If we compare Figure 4.6 with Figure 4.4, as seen in Figure 4.8, we see that the network diameter decreases when the average node degree of the network increase. As the average node degree increase, the graph approaches to a fully connected graph whose network diameter is 1 regardless of the size of the network. Due to this fact, the decrease in the network diameter when the value of α increases, is an expected result.

In Figure 4.7 and 4.8, the change in the network characteristics, when the value of α increase from 0.5 to 1.0, can be observed better.

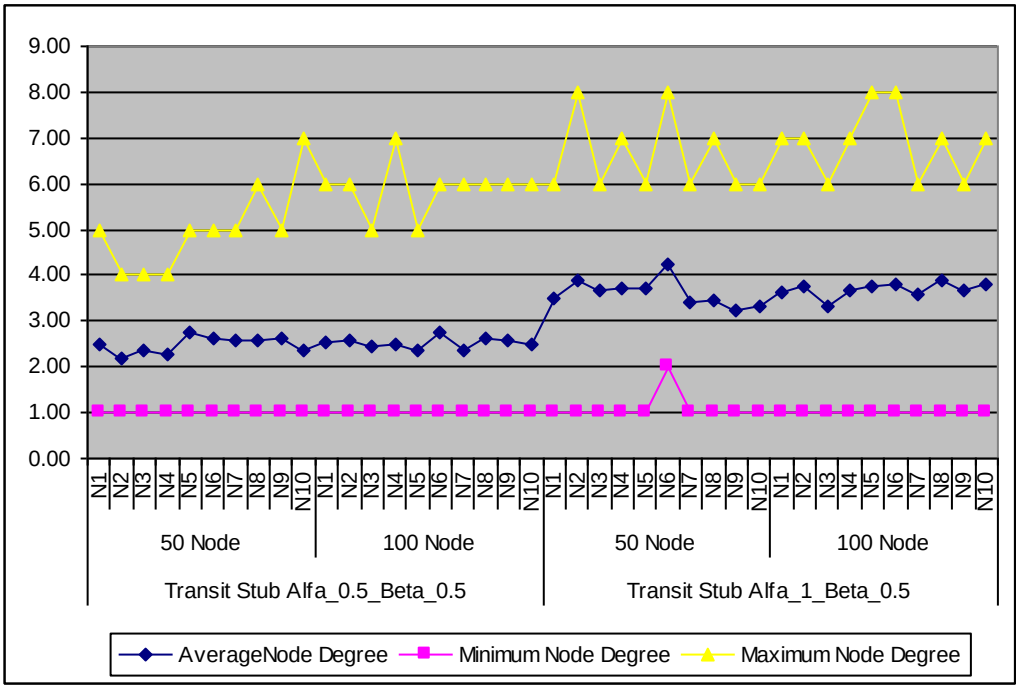


Figure 4.7 the node degree comparison for Transit Stub Graphs (α=1, β=0.5) and (α=0.5, β=0.5)

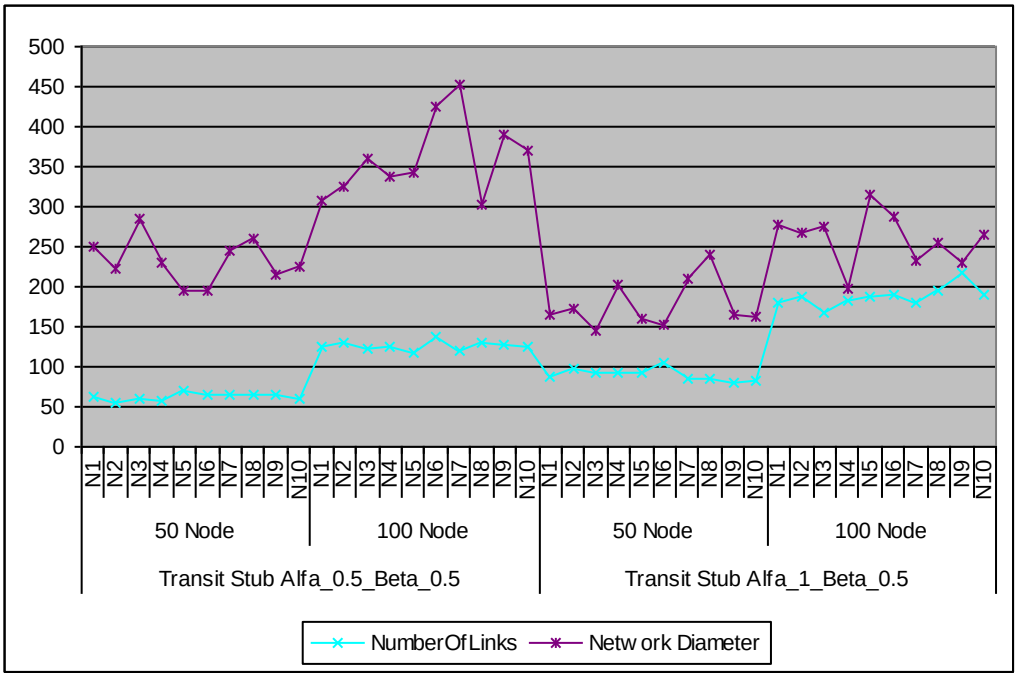


Figure 4.8 the number of link and the network diameter comparison for Transit Stub Graphs (α=1, β=0.5) and (α=0.5, β=0.5)

4.2.1.3 Flat Random Graph (α=0.5, β=0.5)

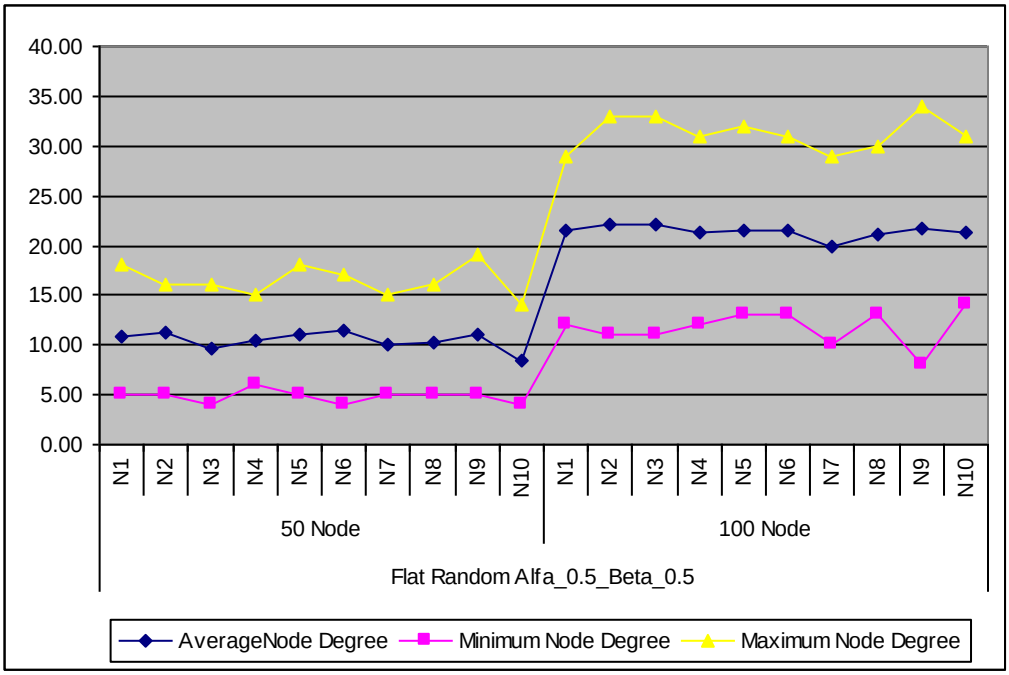


Figure 4.9 the node degree comparison for Flat Random Graphs (α=0.5, β=0.5)

The nodes of the flat random graphs are selected from a single two-dimensional plane, whereas, each graph (transits and stubs) of a single transit stub network is generated from a separate single two-dimensional plane. Therefore, the flat random graphs in the simulation are much denser than the transit stub graphs. This help me see the effect of average node degree on the number of generated multicast messages, delay cost for the multicast protocols.

When the number of nodes in the network increase, the average node degree, minimum node degree and the maximum node degree also increase because the same two-dimensional plane is used to select the nodes for the graphs.

In Figure 4.10, it is clearly seen that the number of links increase when the number of nodes in the graph increase which is also an expected result. In contrast, the network diameter decreases when the network size increases. As the network becomes dense due to the high number of nodes in the network, the network diameter decreased which is an expected result as well.

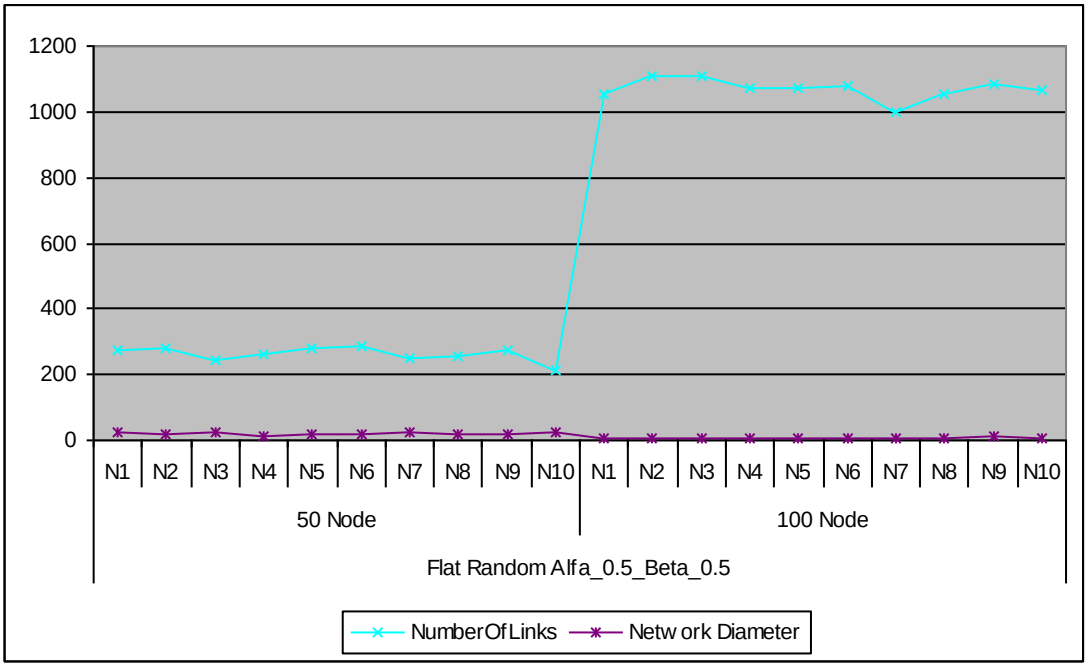


Figure 4.10 the number of link and the network diameter comparison for Flat Random Graphs (α=0.5, β=0.5)

4.2.1.4 Flat Random Graph (α=1, β=0.5)

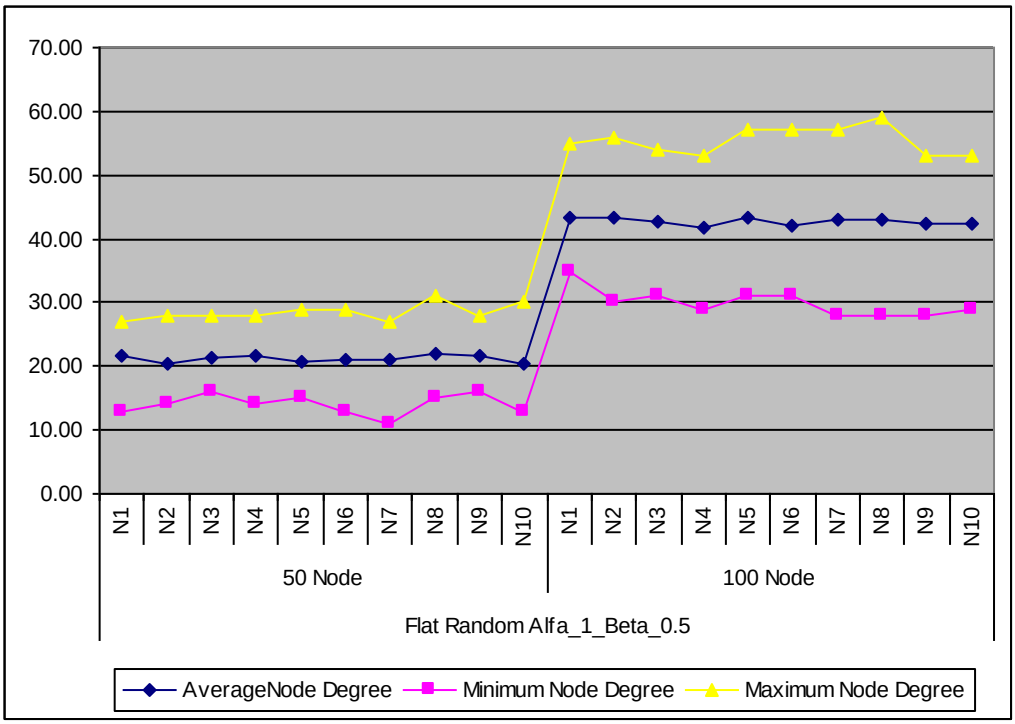


Figure 4.11 the node degree comparison for Flat Random Graphs (α=1, β=0.5)

The results obtained for the Flat Random Graphs with parameter values α=1 and β=0.5 in Figures 4.11 and 4.12 have generally the same tendency as the one shown in Figures 4.9 and 4.10. As the value of α in Figures 4.11 and 4.14 is greater than the value of α in Figures 4.9 and 4.10, the network in Figures 4.11 and 4.12 behave more like a fully connected graph. Therefore, these graphs have higher average node degree values, number of links, but their network diameter is less than the ones in Figures 4.10 and 4.10.

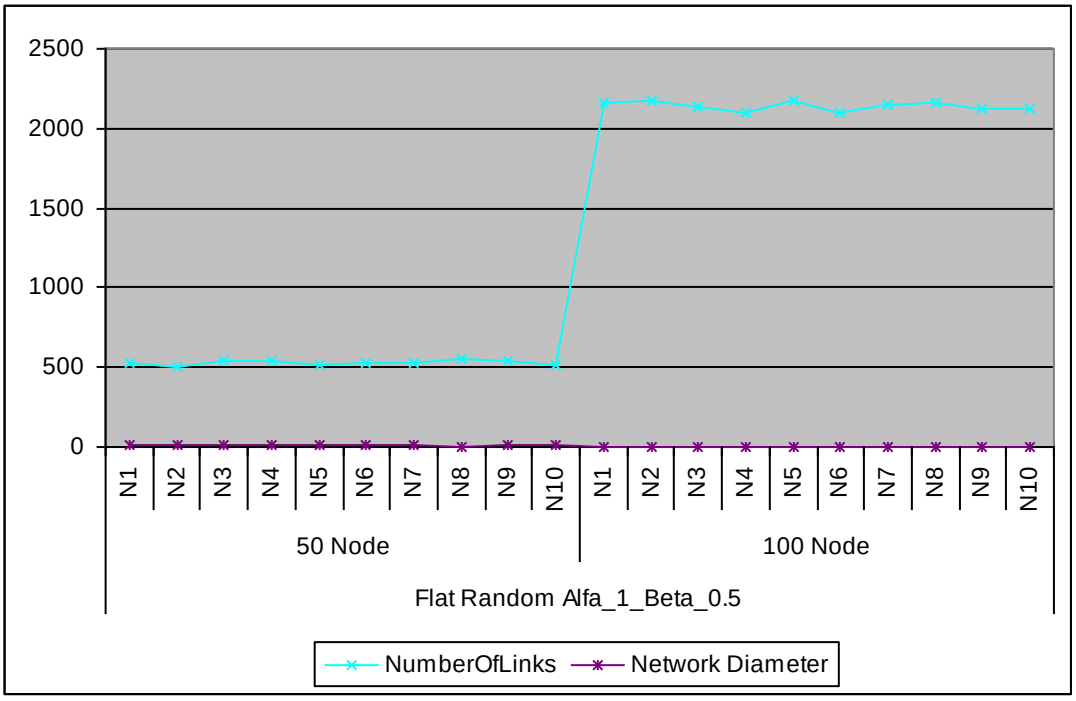


Figure 4.12 the number of link and the network diameter comparison for Flat Random Graphs (α=1, β=0.5)

The Effect of α on the network characteristics for Flat Random Graphs is shown in Figure 4.13 and 4.14 which are discussed above.

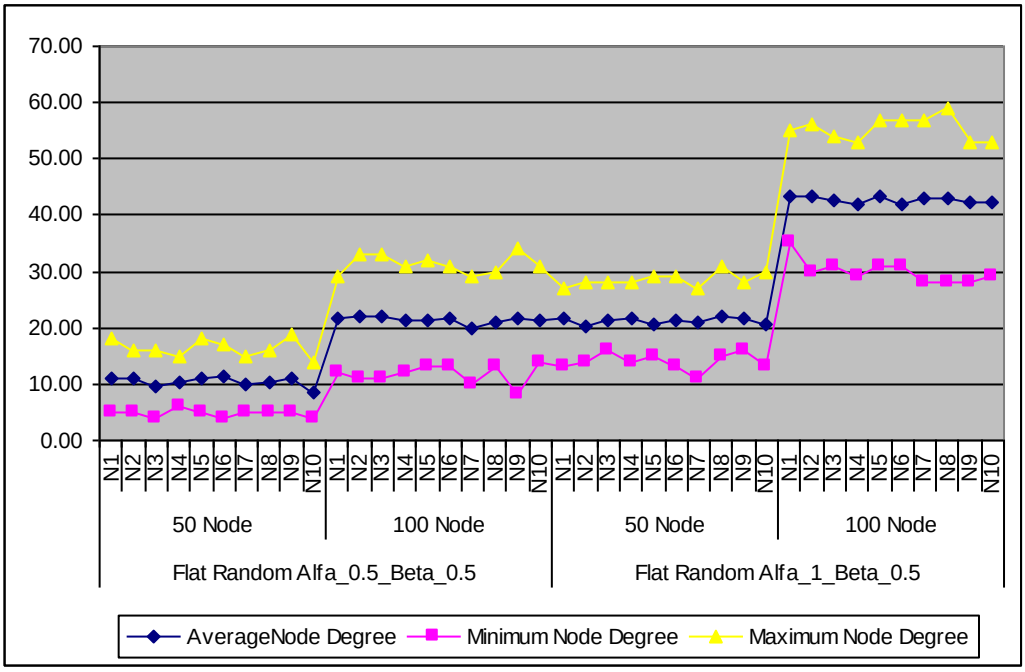


Figure 4.13 the node degree comparison for Flat Random Graphs (α=1, β=0.5) and (α=0.5, β=0.5)

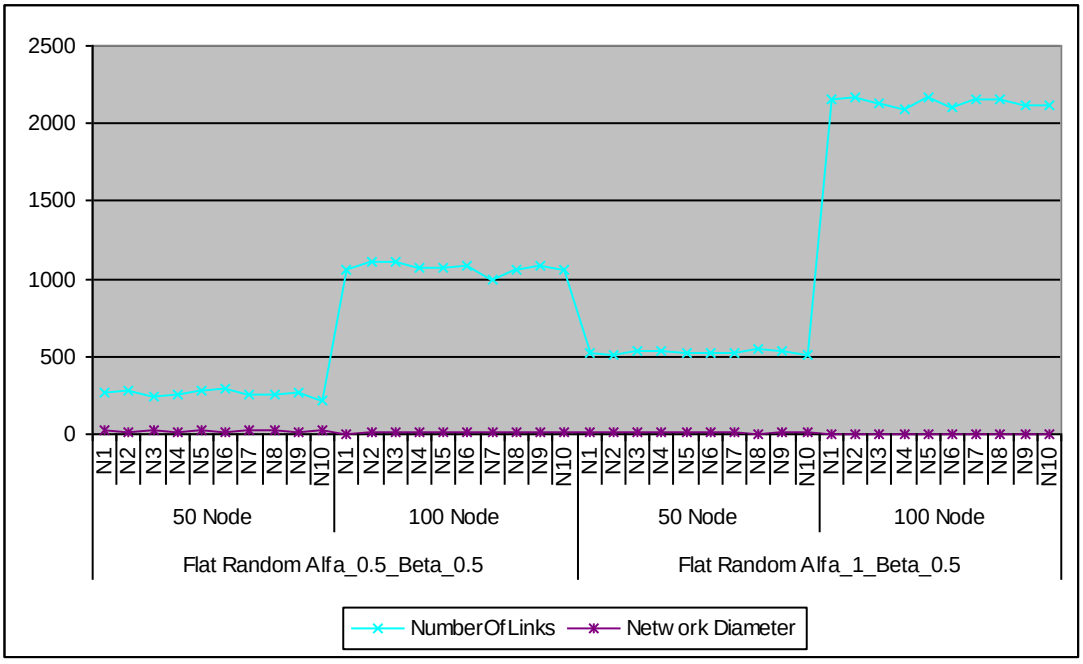


Figure 4.14 the number of link and the network diameter comparison for Flat Random Graphs (α=1, β=0.5) and (α=0.5, β=0.5)

Lastly, the node degree comparison of both Transit Stub and Flat Random graphs with the parameter values (α=1, β=0.5) and (α=0.5, β=0.5) are all shown together in a chart in Figure 4.15.

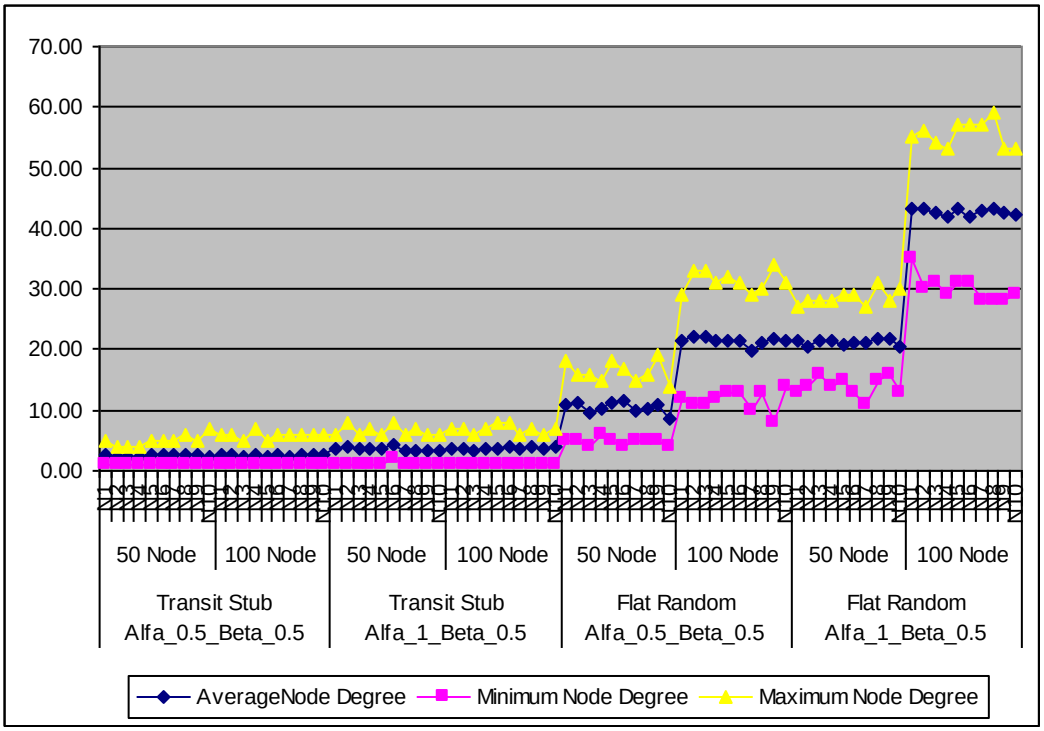


Figure 4.15 the node degree comparison for both Flat Random and Transit Stub Graphs with parameter values (α=1, β=0.5) and (α=0.5, β=0.5)

4.3 Multicast Traffic Types

We developed a program with graphical user interface (GUI) that can generate different types of multicast traffic in Microsoft C#.NET. The software enables users to generate Video-Conferencing (VC) [9], Distributed Interactive Simulation (Scenario IS) [9], Relay Chat (Scenario RC) [9], Seminar (Scenario SM) traffics [9] and Waxman Probabilistic JOIN/LEAVE scenario as well as custom traffic scenarios created by users that are written in C# with a custom interface. In addition to this, the users can analyze the resultant generated traffic files with the graphic generation tool GNUPLOT. The users first select the traffic files that they want to analyze. After that a calculation is made to show the number of sources, receivers and total users participating in the multicast group at constant time slots. The result of the analysis is shown with a line graph by GNUPLOT. So, the user can see approximately how the

multicast application that he/she wants to simulate will behave before making the simulation.

The traffic files, in general, are characterized by the following parameters in the developed software as it is explained in [9].

i) *Migration Probability:* It is the probability that a new participant occurs in a domain which does not exist in the current participant set.

ii) *New Participant Probability:* The result of the probability function is the frequency with which new participants appear in the multicast group. So, by taking the multiplicative inverse of this value, the amount of time to generate a new participant can be obtained. But it does not mean that the new participant is not in the current set of participants. The probability that it is new to the multicast group is evaluated with the probability function given above.

iii) *Life:* It is the percentage of total simulation time that a new participant exists. Therefore, the time for the participant to leave the multicast group can be calculated by using this value and the time that it is generated.

iiii) *Participant Source:* It is the probability that the new generated participant is the source. If all sources are also receivers in the multicast application, this parameter is not taken into consideration.

The general characteristics of a multicast join/leave event are explained above. But every traffic scenario may not be characterized with the parameters given above. It may have its own parameters and a different way of generating the join/leave events. The software that We developed enables users to create their own traffic files by creating their own classes with any number of methods and parameters developed in any .NET programming language. So, the users have the chance of extending the software that We created with very little modification to the source code of the main software.

The software also enables users to change the parameters and the type of the multicast traffic without changing the source code of the program. We created an XML file in which the scenario and its characteristics are defined to enable this flexibility. So, when we want to generated different types of Multicast traffics with different parameters even with different custom written classes and methods, we only change the XML definition file and run the program again by using the Graphical

User Interface. The output of the software, which is the generated traffic file, is also an XML file so that the traffic file can easily be parsed and used by different multicast simulation programs in any operating system with any software development platform which is, in my opinion, is one of the important flexibilities that XML file format provides. Another important flexibility of using XML file is the ease of changing and reading it. In all operating systems, the XML file can be read and manipulated easily by hand which enables us to create different traffic files easily.

As a result, we create a definition file and get a multicast traffic file. In order to get a different multicast traffic file, we simply change the XML definition file. A sample XML definition file is given below to make it clearer.

Portion of XML Definition File:

```
<MulticastScenarios>
  <Scenario name="Scenario_VC" namespace="VSConvert"
class="ScenarioGenerator" method="GenerateScenario" assemblyname=""
numberofunittime="1000" totalsimulationtime="20000"
numberofsimulationfiles="20" outputfilename="" RendezvousPoint="0"
filefiltercaption="Scenario_VC files" filefilter="Scenario_VC*.xml"
definition="Video Conference">
    <Parameters>
      <Parameter name="MigrationProbability" isconstant="1">
          <DistributionFunction name="UniformDistribution" namespace=
"VSConvert.DistributionFunctions">
              <Constant name="coefficient" type="double">0.8</Constant>
          </DistributionFunction>
      </Parameter>
      <Parameter name="NewPartitionProbability" isconstant="0">
          <DistributionFunction name="ExponentialDistribution" namespace=
"VSConvert.DistributionFunctions">
              <Constant name="mean" type="double">50.0</Constant>
              <Constant name="coefficient" type="double">0.9</Constant>
          </DistributionFunction>
      </Parameter>
```

```
        <Parameter name="LifeProbability" isconstant="1">
            <DistributionFunction name="UniformDistribution" namespace=
"VSConvert.DistributionFunctions">
                <Constant name="coefficient" type="double">1</Constant>
            </DistributionFunction>
        </Parameter>
        <Parameter name="PartitionSourceProbability" isconstant="1">
            <DistributionFunction name="UniformDistribution"
namespace="VSConvert.DistributionFunctions">
                <Constant name="coefficient" type="double">0.5</Constant>
            </DistributionFunction>
        </Parameter>
    </Parameters>
  </Scenario>

  <Scenario name="Scenario_Waxman" namespace="VSConvert"
class="ScenarioGenerator" method="GenerateScenario_Waxman"
assemblyname="" numberofunittime="1000" totalsimulationtime="20000"
numberofsimulationfiles="10" outputfilename="" active="1" RendezvousPoint="0"
filefiltercaption="Scenario_waxman"  filefilter="Scenario_waxman*.xml"
definition="Scenario Waxman">

    <Parameters>
      <Parameter name="MigrationProbability" isconstant="0">
        <DistributionFunction name="WaxmanJoinLeaveDistribution"
namespace="VSConvert.DistributionFunctions">
            <Constant name="coefficient" type="Double">1.0</Constant>
            <Constant name="ALFA" type="Double">0.7</Constant>
            <Constant name="NumNodesInNetwork" type="Int32"></Constant>
        </DistributionFunction>
      </Parameter>
      <Parameter name="NewPartitionProbability" isconstant="1">
        <DistributionFunction name="UniformDistribution"
namespace="VSConvert.DistributionFunctions">
            <Constant name="coefficient" type="Double">1.0</Constant>
```

```
        </DistributionFunction>
      </Parameter>
      <Parameter name="PartitionSourceProbability" isconstant="1">
        <DistributionFunction name="UniformDistribution"
namespace="VSConvert.DistributionFunctions">
            <Constant name="coefficient" type="Double">0.2</Constant>
        </DistributionFunction>
      </Parameter>
    </Parameters>
  </Scenario>
</MulticastScenarios>
```

As it is clearly seen, many scenario definitions exists in the XML file (there are two definitions above but may exist more if needed) with the node name "*Scenario"*. But only one scenario is active at a time which has the *"active"* attribute whose value is set to "1". For each scenario, the namespace, class and the method name are specified with the attributes "namespace", "class" and "method" which enables us to change the class and the method to generate the traffic file at run time without making any modification to source code providing us a great flexibility. In addition to this, the total number of traffic files to be generated as output, total simulation time and the unit time in terms of milliseconds are specified for each scenario. After specifying these values, the characteristics of the scenario are specified with probability functions and their parameters. For each scenario, some or all of the parameters among the PartitionSourceProbability, NewPartitionProbability, MigrationProbablity and LifeProbability are defined. These parameters are related with a Predefined Probability Function in the software. For example, the MigrationProbability of the Waxman Scenario is related with a Distribution function (probability function) named WaxmanJoinLeaveDistribution in VSConvert.DistributionFunctions namespace. The distribution functions parameters are also specified in the XML file. So by changing the distribution function name and/or the values of its parameters, we can generate a different multicast traffic scenario which behaves different than the other.

Currently, the supported Distribution Functions by the software that We developed are Normal Distribution Function, Uniform Distribution Function, Exponential Distribution Function, and WaxmanJoinLeave Distribution Function. But the number of supported Distribution Functions can easily be increased by adding new classes written in C#.NET under the existing VSConvert.DistributionFunctions namespace. In order to enable this extension, We created an abstract base class for all the probability functions that can be used to simulate the traffic so that all probability functions inherited from this base class have common interface even if they make the calculations in different ways. So, the main program does not change as it always uses the base abstract class interface and make calls according to the interface definition.

After giving general details about how multicast traffic files are generated, detailed information about the multicast traffic types used in the simulation is given below.

4.3.1 Video-Conferencing (VC)

In a multicast video-conferencing application, the participants join to the multicast group early and remain for the duration of the connection. So, the number of participants in the multicast group does not change frequently. We created Video-Conferencing Traffic with the parameters given in [9] with the software We developed and analyzed the traffic by calculating the change in the number of receivers and sources in the multicast group. We created 10 different Video-Conferencing Multicast traffic all of which have the same parameter values as in [9]. To form the graph, the average of 10 traffic files is taken. The graph of the average multicast video-Conferencing traffic for 100 nodes is given below.

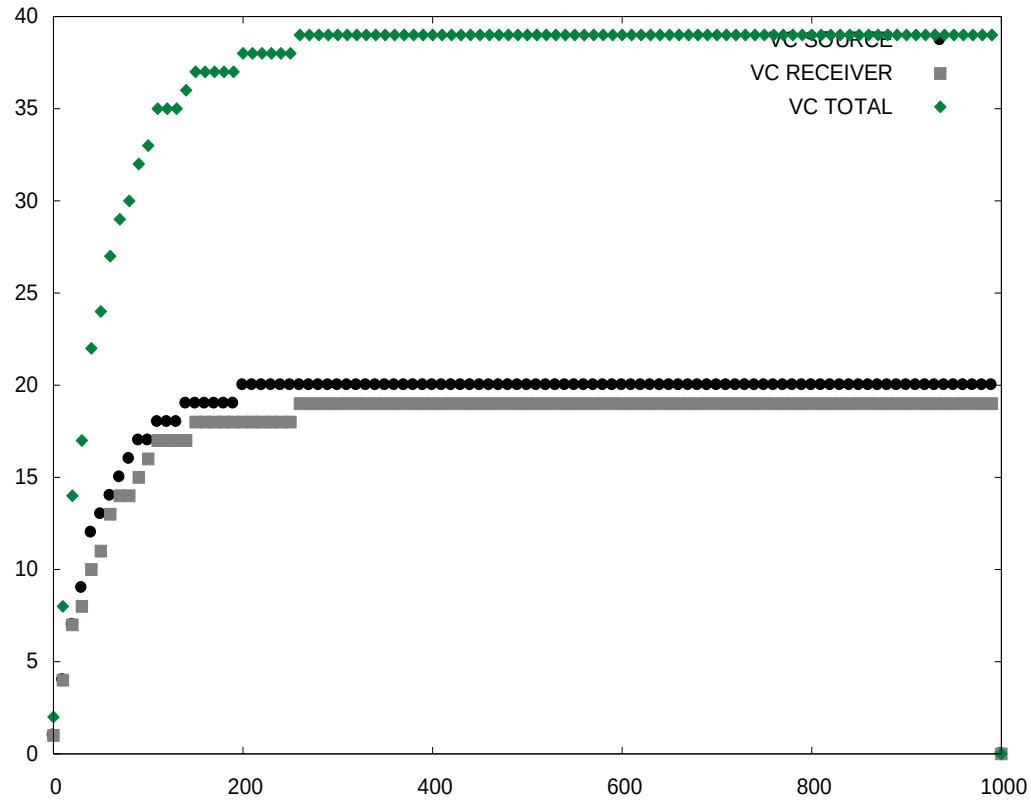


Figure 4.16 Video Conferencing

As seen in Figure , the numbers of participants increase a lot at the beginning of the session and the multicast session gets into the steady state which is the expected result and is almost the same graph in [9]. So, the traffic generator software works perfect for this multicast traffic.

4.3.2 Relay Chat (RC)

The relay chat traffic is very dynamic in contrast to the Video-Conferencing traffic explained above. The participants join to the multicast group frequently but the lifetime of the participant is very short and leaves the group in a short period of time. As a result, the number of nodes in the multicast group changes frequently in very short periods of time. The multicast group never gets into steady state in Relay Chat multicast application as it is in Video-Conferencing multicast traffic. We used the same parameter values as it is in [9] and took the average of the 10 created relay-chat traffic files all of which have 100 nodes.

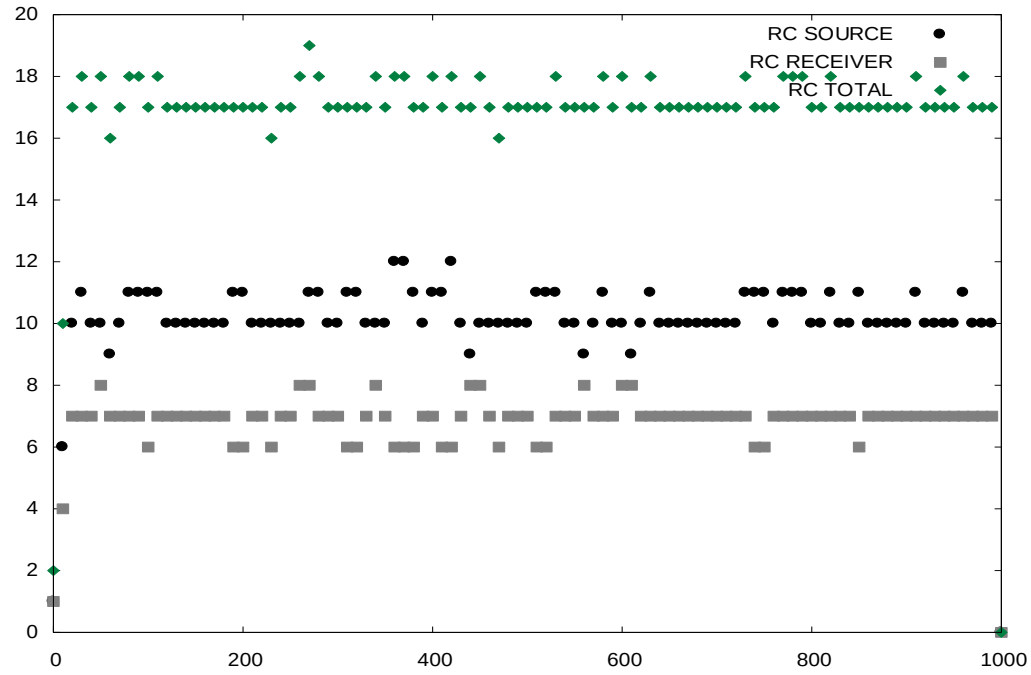


Figure 4.17 Relay Chat

The general shape of the graph is like the one in [9]. The oscillation is a lot as it is expected.

4.4 Routing Protocols

We compared two routing protocols PIM-SM [7] and SCMP [11] to observe the effect of core migration in dynamic multicast communication. PIM-SM, explained in section 2.3.2.1, does not use dynamic Rendezvous Point. The Rendezvous Point is chosen administratively before the communication starts and the selected Rendezvous Point (RP) is used unless it is down. So the performance of the multicast group communication may decrease as the time passes. However, SCMP uses dynamic Rendezvous Point which may change when the multicast tree cost degrades in order to keep multicast tree cost at a satisfactory level so that group communication is not affected from the increase in the number of multicast group participants. But SCMP may not be scalable if its parameters are not chosen well enough which may result in high volumes of message traffic in the network.

We compared these two protocols in terms of tree cost, number of generated messages in the network, the delay variance and the average delay. We run the simulation for the networks explained in section 4.2, used the traffics explained in

section 4.3. The results of the simulations are explained in the following sections in detail. The protocols and implementation details of the multicast protocols are explained in sub sections 4.4.1 and 4.4.2.

4.4.1 Base Classes and Network Implementation for Multicast Protocols

We created three main classes; Node Class, Network Class, Edge Class to implement the multicast protocols. We overloaded the operators like "!=", "= =" for all of these classes so that the users of these classes can easily use the objects created from them without any knowledge of how these classes are implemented. The detailed information about these classes are given in the following three sections.

4.4.1.1 Node Class

Node class is the base class for SCMP and PIM-SM protocols. Both SCMP Node in SCMP Protocol and the PIM Node in PIM-SM Protocol inherits Node Class. Node class stores the Unicast Routing Table and the outgoing and incoming interfaces that connect the node to its neighbors.

The Unicast Routing table is formed by flooding. Event Firing Mechanism in .NET is used to implement flooding. When the edges connecting two nodes are created, both nodes subscribe to each others' RoutingTableChanged Event. RoutingTableChanged event fires when a change in the Routing Table of a node occurs. But to prevent high number of messages flooding in the network, every node in the network waits for a period of time after the last change in its Unicast Routing table occurs. If no other Unicast Table change occurs after the last change for that predefined amount of time, the event is fired and every neighbor of that node catches the fired event and updates their routing tables. Whenever an update in the Routing table of a node occurs, the same procedure runs again.

The event classes that We create in .NET are classes that are derived from System.EventArgs base class. Other than the inherited methods, properties and fields, the custom written events store specific information. So, when a node fires an RoutingTableChanged event, the subscribers of that event, which are the neighbors of that node, catch the fired event and use the information stored in that event class to update their own routing tables. After some time passes(varies depending on the size of the network), the nodes in the network get into steady state which means that all

the nodes in the network know the interface to reach another node in the network by using the shortest path.

4.4.1.2 Edge Class

We created an Edge class to store information about its endpoint nodes and its cost. The network class creates new edges by specifying its end Nodes and its cost. By the way, the nodes are created as well. And the network is constructed finally.

4.4.1.3 Network Class

The nodes are created in a network. Therefore, we implemented Network class to hold the nodes in the network. The network class creates the network dynamically. The class reads the network configuration from a XML file which is created by the software explained in Section 4.2 or it is told explicitly how the network should be formed by calling the appropriate methods. The class has the necessary interfaces to create/drop nodes in the network, create/drop edges between two nodes and print the multicast tree at any time.

4.4.2 PIM-SM Protocol

In the following sections, assumption made in the simulation for PIM-SM protocol and the implementation details are explained in detail. Detailed information about PIM-SM can be found in Section 2.3.2.1.

4.4.2.1 Assumptions

The bootstrap router and the bootstrap messages are ignored in the software and are not implemented. In addition to this, despite we implemented the source join messages to multicast communication, we did not create any source nodes in the simulation. As we are interested in the tree cost and delay variance of the multicast tree, we implemented the joining and leaving process of the nodes for the multicast communication. Moreover, we simulated Relay Chat and Video-Conferencing applications in which all the receivers are also the senders which is more appropriate as we did not implement the sender process. The software implementation of the protocol is done according to the guidelines in PIM-SM draft [7].

4.4.2.2 Class Hierarchy

We implemented the PIM-SM protocol in Microsoft C#.NET in an object oriented manner. We created different classes for the messages and the data structures at the nodes in order to do the message processing and data structure manupilation easily. We created a Node class for the multicast nodes, Edge Class for the links between the ndoes, the Graph class to keep information about the network and the Event Class to demonstrate the events fired during the simulation. We implemented flooding mechanism which enables nodes in the network to create their Unicast Routing tables by using threads in C#. When there is a change in the Unicast routing table, the nodes notify its neighbours by firing an event which keeps information about the changes in its unicast routing table. As the neighbours of that node has already subscribed to the event when the network was being created, they got the message when the event is fired. So, they update their unicast routing tables and notify their neighbours. This process goes on continously as it is in real networks.

The multicast communication is triggered from another process(an executable written by me in Microsoft C#.NET to execute different processes) which is responsible for executing the simulation by providing different parameters(scenario Type, ALPHA, node Number etc...) to it.While the simulation is running, the snapshot of the network is taken and stored in a XML file after each join or leave process.

As there are lots of message comparison done by the multicast nodes during the simulation, we implemented operator overloding functions for different classes in or to make message processing simpler. We used hash tables to store unique informations in order to make processing faster. The source code and other necessary files can be found in the CD.

4.4.2.3 Data Structures at Participants

The UnicastRoutingTable, MInterface, MulticastRoutingTable, MulticastRoutingTableEntry and the McastRoutingState are the main data structures stored at each node in PIM-SM protocol implementation.

UnicastRoutingTable is a hash table which stores the IP of the node in the network that it can reach as the key of the hash table and the MInterface data structure as the value of the hash table.

MInterface stores the edge information(the next hop router to reach to the destination) and the total cost of the full path to reach to the destination.

MulticastRoutingTable is a vector(an array with unlimited size, an item can be added and dropped at any time) that stores the MulticastRoutingTableEntry data structure in its indices.

MulticastRoutingTableEntry is the main data structure which stores information about multicast routing at the node. It stores the Source Adress, multicast group adress, the incoming interface, the outgoing interface list to forward the packets to when it come from that incoming interface and some other function to do processing easily.

Lastly, McastRoutingState data structure is used to keep the multiast routing states like (*,G),(S,G) and (S,G,rpt). The state search is done by using this data structure instead of the whole Multicast Routing table which fastens the search.

4.4.2.4 Join / Leave Messages

Both join and prune messages are sent by a single message structure called PIMJoinPruneMessage. The message has 5 parts that identify the JOIN/Prune message. The first part is the IP address(4 byte) of the multicast group which a join/leave is requested from. The second part is the WildCard bit(1 bit). If it is set to 1, it means that the join/leave is requested from any source. Otherwise, it means that the join/leave message is requested froma specific source. The third part of the message is the IP address of the node to join to. It is either the address of the Rendezvous point(for the shared tree) or the address of the source(for source specific tree). The fourth part of the message is the IP address of the node to leave from. It is either the address of the Rendezvous point(for the shared tree) or the address of the source(for source specific tree). And the last part is the RPTree_Bit which determines whether the packet will be forwarded on the shared tree or not.

4.4.3 SCMP Protocol

SCMP protocol [11] is a multicast protocol which changes its Rendezvous Point by time if the increase in the cost of the tree is above the predefined threshold value. IN SCMP, the nodes are categorized as RP, Agent and DR. RP is the rendezvous Point of the multicast tree. Agent nodes, which is explained in the following paragraphs in a more detailed manner, may be called as the sub-RPs of the subtrees(the multicast tree is divided into subtrees.) which they are responsible for. And lastly, Designated routers(DRs) are the simple nodes that represent a LAN(RP and Agents are also designated routers). Designated Routers are divided into two types; Member Designated Routers and Non-Member Designated Routers. Member Designated Routers are the nodes to which at least one host is attached from the the LAN that it is in. Non-Member designated routers are the nodes to which no host is attached from the LAN that it is in. A DR can also be categorized as Leaf DR or Non-Leaf DR. A leaf DR is a designated router which has no downstream routers attached to it.

In SCMP, the multicast tree is divided into subtrees. Each subtree has an agent which keeps information about the nodes in the subtree.Agents store a table called “Memt” in which the IP address of the nodes in its subtree and the distance between the node and the agent is kept. The DRs store a table called “MRT” which keeps information about its child DRs, the IP address of the multicast group and the IP address of its parent to reach the Agent.In addition to these tables, the RP also stores a table called “Agent List”(AL) in which the IP addresses of the agents are stored.

The agents periodically monitors its subtree and calculate the max delay in its subtree.If the max delay is bigger than the maximum delay bound, then the subtree is merged into other subtrees which means that at leat one new agent is created. So, the protocol keeps the depth of the subtrees below a delay bound value. While the agents monitor the nodes in their subtrees, the RP monitors the agents. Rp periodically requests from its adjacent agents to calculate their weights. If an agent with a lower weight than the current RP is found, then it starts probing.This process continues recursively to find the best agent. If a new agent with a lower cost is found, then core migration process occurs.

The protocol is a based on a distributed system which shares the work to the nodes in the network. In fact, the RP relocation process is the tree balancing process. When

the RP is relocated, the new selected RP is already on the multicast tree. The only thing that happens is the tree balancing process.

SCMP performs five main tasks; Join Process, Leave Process, The process of the shrinking of the agents, the process of the expanding of the agents and core migration process. All these processes are implemented according to the algorithms stated in [11].

The join process begins with a DR's sending a join Message towards the RP.The node starting the Join Process is called Join Pending Router(JPR). First-On-tree DR that gets the messages replies with a acknowledgement message and forwards the request towards the core. But it does not reach the core as the core do not need to know the members of the agents. The first on tree agent that receives the message stores the IP of the DR and the distance between thenm to its MemT table. The distance between the receiver and its prehop node is added to the distance value in the message(is 0 in the beginning) until the message reaches the first ON-Tree agent. So, the agent can determine the distance between itself and the JPR and adds it to its Memt Table. If an Off-Tree node on the path betwen the JPR and the first on tree DR receives the message, it becomes On-Tree DR when it receives the acknowledgement message. Bu the agent does not have nay information about this node because the agent keeps track of the nodes that have at least one host attached to it from its LAN.

The leaving process begins with a DR's sending a Leave message towards the core which is called as the LPR(Leave Pending Router). When all attached members(the members on its LAN) of a DR leaves the group, it sends a Leave message and deletes its MRT table. Its parent node removes it from its MRT table and forwards the message toward the agent.If the parent node has no members attached to it from its LAN and the LPR is its only child, then it also sends a PRUNE message towards the Core.When the agent receives the Leave message, it first checks whether the LPR is its child or not. If the LPR is its child, it removes the related entry from its MRT table. As it also keeps information about all its children in its MemT table, it removes the related entry from its Memt table,too. If LPR is also an agent, the LPR's upstream agent also sends a message towards the core so that the RP removes LPR from its AL(agent list).

The expanding process is triggered by the agents in the multicast tree. It send a CREATE_AGENT_REQUEST message to its children and deletes its MRT table. After receiving the reply it updates its Memt table.When the leaf nodes in the subtree receive the message, they create a DISCOVER messaage and set the distance value to the threshold value minus the distance between itself to its next hop DR towards the RP and send the message towards its agent.When The DRs on the path to the agents receive the DISCOVER message,they also decrease the distance value and check whether it is under zero or not. If it is under zero, it means that they should be the agent for the subtree which includes the nodes on the path from the leaf up to itself(All nodes receiving the message adds their IP addresses to the messages and forwards it. So the receiver knows the nodes in the subtree). In order to be agent, they notify the RP and set a new message and send it towards the initial requesting agent. This process continues recursively until the initial requesting agent is reached.

The fourth process is the shrinking process. If the total of distance of the path between the requesting agent and its sub agent and the maximum delay in the subagent's subtree is below the threshold, then the subtree of the subagent is merged with the requesting agent's subtree. The process is similar with the expanding process and its details are given in [11].

The last process is the Core Migration process in SCMP. The Core request from its agents to calculate their weight with a predefined weight function. They calculate their weights and reply to the Core. When the Core receives the replies, it compares its own weight with the received ones. If there is an agent with a lower weight than itself, it chooses that agent as the next probing agent. This process continues until the agent with the lowest weight is found. The final probing agent becomes the new core finally.

4.4.3.1 Assumptions

The Agent Weight Calculation Process triggered by the Core is not implemented in a distributed fashion to reduce the complexity of the simulation. Instead, the core node directly calls the specific methods of the agents to calculate their weights and got the result as a return value from these methods. But this does not effect the overall process as the same values are obtained in both cases.

4.4.3.2 Class Hierarchy

We used the Node Class as a base class that we implemented for PIM-SM protocol for

SCMP implementation(we inherit the Node Class). Some changes are made to the Graph class so that the multicast network can be trackesd by the console application.

In addition to the PIM-SM classes new WeightFunction namespace which includes three different weight function classes inherited from a abstract base weight class(all classes have the same interface) is implemented. For SCMP protocol, a SCMPNode class with SCMP specific methods, events, thread and properties is implemented. The messages and the data structures for SCMP are implemented in seperate classes.The details about the implementaion and the source code can be found in the attached CD-ROM.

4.4.3.3 Join/Leave Messages, Prune Messages, Agent Messages, Core Messages, Data Structures at Participants

The software implementation is done according to algorithms and data structures in [11]. We did not change any part of the algorithms or the data structure in [11] except the bugs that we found and explained in the following section.

4.4.3.4 Protocol Bugs

We found some algorithm bugs which are not addressed in the paper as a result of the simulations that we made. The bugs and their details are explained in the following paragraphs.

During the leaving process, the first upstream agent, which receives the Leave message, removes the LPR if LPR is one of its children. But the Leave process may be generated as the LPR may not have any hosts attached to it but has DRs as its

children. In this case, when the agent removes it from its MRT table(Designated Router Table), then the messages will not be forwarded to these nodes which will end the communication between the sender and the children of the LPR. So, the agent should know whether it has attached DRs or not. We corrected this error in my implementation.

The second bug that we found was about the shrinking process in SCMP. As the shrinking process is done periodically, during that period an agent may loose all its children but is still an agent according to the algorithm. So, its maximum delay will be 0 for this case. And when the shrinking message is received by that node, it compares the delay of the path from the upstream agent to itself, adds its maximum delay in its subtree to that value and compares the result with the maximum delay bound. As the maximum delay is zero in its subtree, if the distance is greater than the maximum delay bound, it will continue to serve as an agent even though it is a Non-Member DR which must leave the tree.We corrected this error in my implementation, too.

The third and the final bug that we found is about the Changing Core Process.When the core changes, it in fact tries to balance the tree. So it updates only the MRT tables of the nodes on the path from the new RP to the old RP so that the parent and the children information on these nodes may be correct. But the Memt Table entries of the agents on the path are not updated enough. The nodes in the subtrees of the agents(which are on the path from the new RP to the old one) does belong to another agent's subtree in that case. But this is ignored during core migration. Another bug, which is less important, is that SCMP does not offer any algorithm to redirect the Join/Leave messages to the new core during core migration. So the nodes join to the old RP instead of the new one which may cause a longer path to reach the new RP. This is not as important as the other bugs as this can be corrected by the tree balancing process done later on.

4.5 RESULTS AND ANALYSIS

As explained in Section 4.4, We compared two routing protocols PIM-SM [7] and SCMP [11] in terms of tree cost, average delay, delay variance, message traffic. We ran the simulations on differnet multicast networks with different multicast scenarios

as explained in section 4.2 and 4.3. In the following sections the results of the simulations are given and analysis according to the simulation results are made.

We used four parameters(Scenario Type, Number Of Nodes, Network Type and ALPHA(α,determines the node degree of the graph) to calculate tree cost for PIM-SM and SCMP protocol. We ran the simulation 10 times by changing one parameter each time(the other three metrics were constant) with a software that we developed automatically.

We chose the node number of the networks as 50 and 100, the value of ALPHA(α) as 0.5 and 1, the scenarios as Video Conference(VC) and Relay Chat(RC) and the network types as Flat Random(FR) and Transit-Stub(TS). As a result of the simulations, XML files are created by the softwares(for both multicast protocols) which keeps information about the nodes and network characteristics with a total size of 4GB approximately. After the creation of the XML files, we wrote a simple software in Microsoft C#.NET to recursively parse the XML files and create Microsoft Excel Charts which are listed in **Hata! Başvuru kaynağı bulunamadı.**.

In the following sub sections, we compare the multicast routing protocols in terms of Tree Cost, Avrage Delay, Delay Variance, Network Traffic.

4.5.1 TREE COST

SCMP protocol, in comparison with PIM-SM protocol, gave better results for Average Delay,Delay Variance and tree cost. The file names with even numbers between EK-A 2: PIM-SM, Flat Random Network, 50 nodes with α=0.5, Scenario Type : RC(Relay Chat) and EK-A 64: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:VC(Video Conference) (e.g: EK-A 2: PIM-SM, Flat Random Network, 50 nodes with α=0.5, Scenario Type : RC(Relay Chat), EK-A 4: PIM-SM, Flat Random Network, 50 nodes with α=1, Scenario Type : RC(Relay Chat), ..., EK-A 64: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)) are the charts that shows the tree costs versus time for diffrent values of the parameters such as Scenario Type, number of nodes, network type and ALPHA(α).

It is an expected result that SCMP performs better than PIM-SM in terms of tree cost due to the fact that SCMP protocol changes its RP(Rendezvous point) to decrease the tree cost when it is above the threshold value which is predefined by the user in the

protocol. However, in PIM-SM protocol, RP is chosen administratively before the multicast communication and can not be changed later on unless it is down.

Greater values of ALPHA(α) increases the average node degree in the network which is explained in section 4.2.1 in a detailed manner.As a result of simulations, when ALPHA increases from 0.5 to 1 , tree cost decreases which is also an expected result. As the node degree increases(when ALPHA increase), the total number of links in the graph also increase. When there are more links in the network, the probability of finding better paths(low cost links) increase. Therefore, tree cost decrease.

When we increase the number of nodes in the network, we obtain different results according to the type of the network. For Flat Random networks, in which, the nodes are chosen from a single two-dimensional plane, an increase in the number of nodes decreased the tree cost as expected. As the nodes are chosen from the same plane, when the number of nodes increase, the nodes can find better paths(with low cost) to join to the multicast group. So, the tree cost decreases. But for the transit-stub networks, when the number of nodes increase, the tree cost also increases which ,at first, is not an expected result.The 50-node transit-stub networks are created with one transit and 5 stub graphs. The nodes in both transit and stub graphs in 50-node transit-stub graph are chosen from a 10*10 or 15*15 two-dimensional plane.Whereas 100-node transit-stub graphs include 2 transit and 10 stub graphs. The nodes of these sub graphs are chosen from a 30*30 two-dimensional plane(More detailed information van be found in section 4.2.1). When the nodes are chosen from a larger plane, the cot of the links between nodes increase.Moreover, as there are more transit graphs in 100-node transit-stub graph, the shortest path between different nodes passes through the transit-graphs if the stub graph of the nodes are connected to diiferent transit graphs. Therefore, the cost of the shortest path between these nodes increases.As a result, for transit-stub networks which are used in the simulation, the tree cost decrease when the number of nodes increase.The simulation results can be shown in EK-A 22: PIM-SM, Transit-Stub Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat) and EK-A 18: PIM-SM, Transit-Stub Network, 50 nodes with α=0.5, Scenario Type:RC(Relay Chat), EK-A 20: PIM-SM, Transit-Stub Network, 50 nodes with α=1, Scenario Type:RC(Relay Chat) and EK-A 24: PIM-SM, Transit-Stub Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat), EK-A 50: SCMP, Transit Stub Network, 50 nodes with α=1, Scenario Type:RC(Relay Chat)

and EK-A 55: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat), EK-A 52: SCMP, Transit Stub Network, 50 nodes with α=0.5, Scenario Type:RC(Relay Chat) and EK-A 53: SCMP, Transit Stub Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat).

If we change the scenario type from RC(Relay Chat) to Video Conference(VC), we see that SCMP and PIM-SM may give different responses. To see the change in the tree cost when the scenario type changes, the charts with the names with even numbers between EK-A 2: PIM-SM, Flat Random Network, 50 nodes with α=0.5, Scenario Type : RC(Relay Chat) and EK-A 64: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:VC(Video Conference) (e.g: EK-A 2: PIM-SM, Flat Random Network, 50 nodes with α=0.5, Scenario Type : RC(Relay Chat), EK-A 4: PIM-SM, Flat Random Network, 50 nodes with α=1, Scenario Type : RC(Relay Chat), ..., EK-A 64: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)) must be analysed.

For SCMP protocol, in both Flat Random Graphs and Transit-Stub graphs, the tree costs for VC are greater than the tree costs for RC. In VC scenario, the number of joins are very less than the ones in RC. Therefore, the tree may not be balanced enough by SCMP in VC scenario which may result in higher tree costs. When the number of nodes increase in the Flat Random networks, the effect of the change in the multicast scenario type to the cost of the multicast tree decreases which is as expected. It is also seen that SCMP, when compared with PIM-SM for Flat Random Networks, does not perform for VC as good as it performs for RC(but for both RC and VC, SCMP performs better, but especially for RC it performs much better). Another result obtained from the simulations is that If SCMP protocol is used instead of PIM-SM for Transit-Stub Networks, a bigger decrease in the tree cost can be obtained than the one for Flat Random Networks.

As a result of these observations, we see that in intradomain dense networks, if the changes in the multicast group does not happen very often(like VC), we may prefer a multicast routing protocol with constant Rendezvous Point.

4.5.2 AVERAGE DELAY,DELAY VARIANCE

SCMP generally performs much better than PIM-SM for average delay and delay variance.

When node numbers in the network increase from 50 to 100, the delay variance and the average delay decrease for Flat Random Networks for both SCMP and PIM-SM. For transit-stub networks, the increase in the number of nodes from 50 to 100 also increases the average delay and delay variance. Detailed explanation about the increase,which is not an excpected result is given in section 4.5.1.

As the increase in ALPHA also increases the average node degree which enables us to find a lower cost path between any two node, it decreases the average delay and delay variance.

When the scenario type changes the average delay and the delay variance does not change significantly, but the values for VC is a little bit larger than RC.

4.6 CONCLUSION

In this work, the performance of SCMP protocol is observed in terms of tree cost, delay variation and average delay. The results obtained are compared with these of PIM-SM protocol. The delay variance, average delay and tree cost values lower in SCMP than the ones in PIM-SM generally. It is observed that the message traffic size between the nodes in SCMP protocol is larger than that of PIM-SM. Another point to note is that SCMP protocol, which is a dynamic RP relocation protocol, does not provide a significant increase in the performance for intra-doman multicast applications with stable traffic like Video-Conference. Therefore, using less complex protocols like PIM-SM for such applications in intra-domain may give better results in terms of message traffic. In addition to these observations, it is seen that SCMP protocol has some open points to be studied on, especially for core migration process which is generally most important part of the dynamic multicast protocols.

As a future work, we are planning to focus on applying new weight functions for multicast communication which takes more parameters into account than the ones currently being used and to find a core migration algorithm for SCMP in order to make it reliable.

As a result, we developed a new flexible multicast scenario generator to be used further on and analysed the advantages and disadvantages of dynamic multicast protocols from different aspects successfully.

REFERENCES

[1] **Lexman H. Sahasrabuddhe, Biswanath Mukherjee,** 2000. Multicast Routing Algorithms and Protocols: A Tutorial ,IEEE Network, Vol 14, No 1, 90-102

[2] **Victor O. K. Li,Fellow, IEEE, AND Zaichen Zhang,** 2002. Internet Multicast Routing and Transport Control Protocols, *Proceedings Of The IEEE*, Vol:90, 360-391

[3] **Pragyansmita Paul , S. V. Raghavan** . A Survey Of Multicast Routing Algorithms and Protocols, *Indian Institute of Technology Madras*, India

[4] **Antti Vainio,** 2001. IP MulticastRouting Algorithms and Protocols, *Helsinki University of Technology Department of Computer Science and Engineering*, Helsinki

[5] **Muhammad Banikazami,** 1997. IP Multicasting: Concepts, Algorithms and Protocols, *Ohio State University , Department of Computer Science and Engineering*.

[6]**Aaron Strigel and G.Manimaran,** 2002. A Survey Of QoS Multicasting Issues, *IEEE Communications Magazine*, June,

[7]**Bill Fenner, Mark Handley, Hugh Holbrook, Isidor Kouvelas,** 2003. *draft-ietf-pim-sm-v2-new-08.txt.*

[8]**Bill Fenner, Mark Handley, Roger Kermode, David Thaler**, 2003. *draft-ietf-pim-sm-bsr-03.txt.*

[9]**Michael J. Donahoo, Kenneth L. Calvert, and Ellen W. Zegura**, 1997. Center selection and migration for wide-area multicast routing, *Journal of High-Speed Networking*, Vol 6 , Issue 2.

[10]**D.G. Thaler and C.V. Ravishankar**, 1997. Distributed center-location algorithms, *IEEE Journal on Selected Areas in Communications*, Vol 15, No 3, 291-303.

[11]**Ting-Yuan Wang,Lih-Chyau Wuu and Shing-Tsaan Huang**, 2003. A scalabale Core Migration Protocol For Dynamic Multicast Trees", *Journal Of Information Science and Engineering*, Vol19, 479-501.

[12]**Ying-Dar Jason Lin, Nai-Bin Hsu, Ren-Hung Hwang**, 2002. RPIM-SM: extending PIM-SM for RP relocation, *Computer Communications* ,Vol 25,No 18, 1774-1781.

[13]**Ritesh Mukherjee and J. William Atwood**, 2003. Rendezvous Point Relocation in Protocol Independent Multicast-Sparse Mode, *Telecommunications Systems*, Vol 24, No 2, pp. 207-220.

[14]**Bhed Bahadur Bista, Goutam Chakraborty, Toyoo Takata**, 2003. An Efficient RP(Rendezvous Point) Replacement Mechanism for Rendezvous-based Multicast Routing, *Proceedings of the 23 rd International Conference on Distributed Computing Systems Workshops*

[15]**Changdong Liu, Myung J. Lee, Tarek N. Saadawi**, 1998. Core-Manager Based Scalable Multicast Routing, *Proceedings of International Conference on Communication, IEEE,* pp. 1202-1207.

[16]**H. C. Lin and Z. H. Lin**, 2002. Selection of Candidate Cores for Core-Based Multicast Routing Architectures, *Proceedings of the IEEE ICC'2002*.

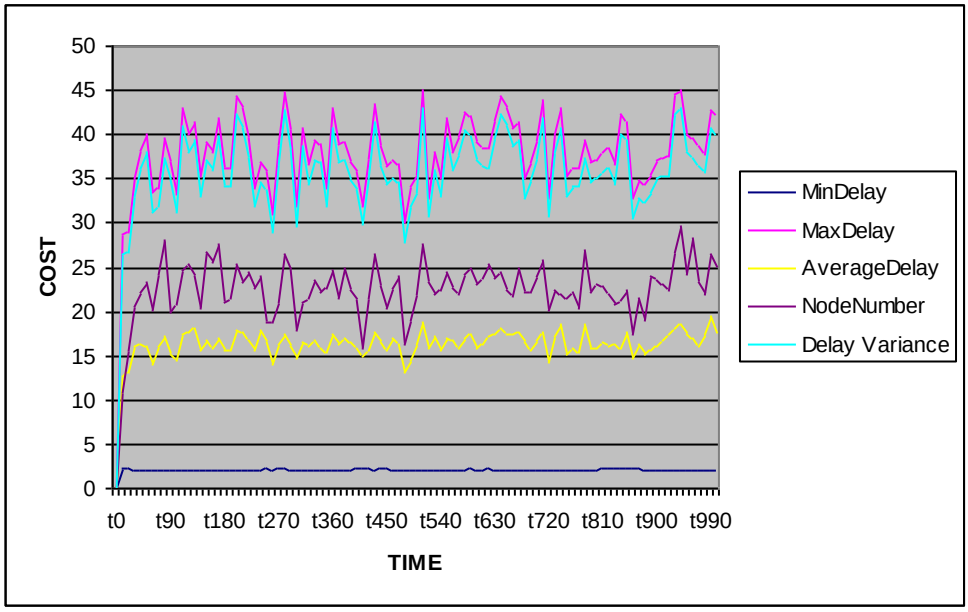
[17]**Hend Koubaa, Éric Fleury**, 2000. Active Multicast Core Migration, *IEEE International Conference on Networks (ICON'00).*

[18]**K. Fall, K. Varadhan**, 2000. The ns Manual, *http://www.isi.edu/nsnam/ns/doc/*

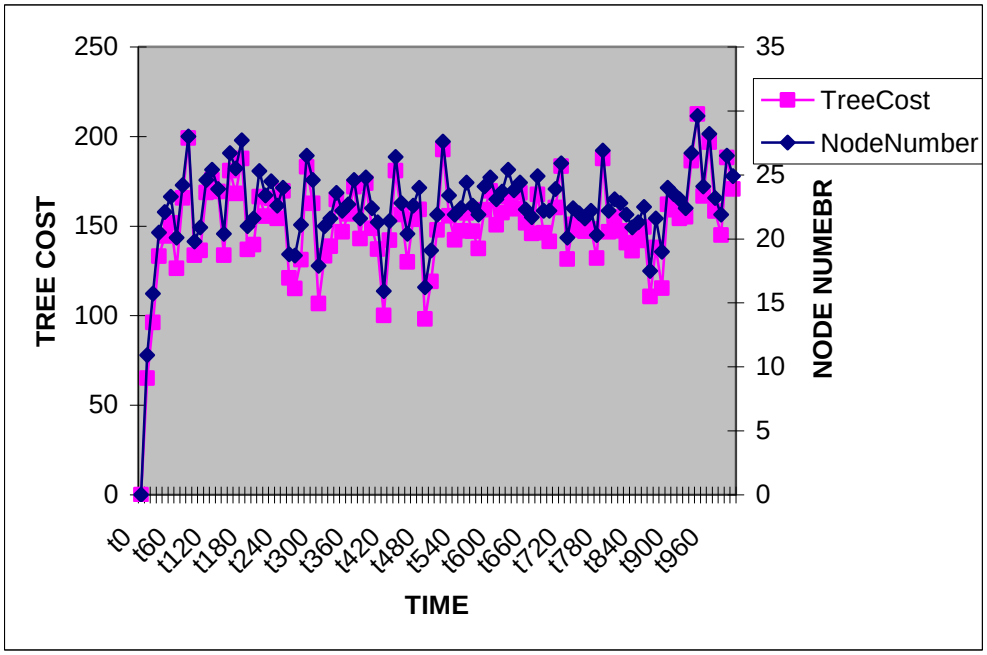
[19] GT-ITM: Georgia Tech Internetwork Topology Models, *http://www.cc.gatech.edu/projects/gtitm/*

[20]**B. M. Waxman**, 1998. Routing of Multipoint Connections, *IEEE Selected Areas in Journal of Commun.*, Vol SAC6, pp. 1617-1622.

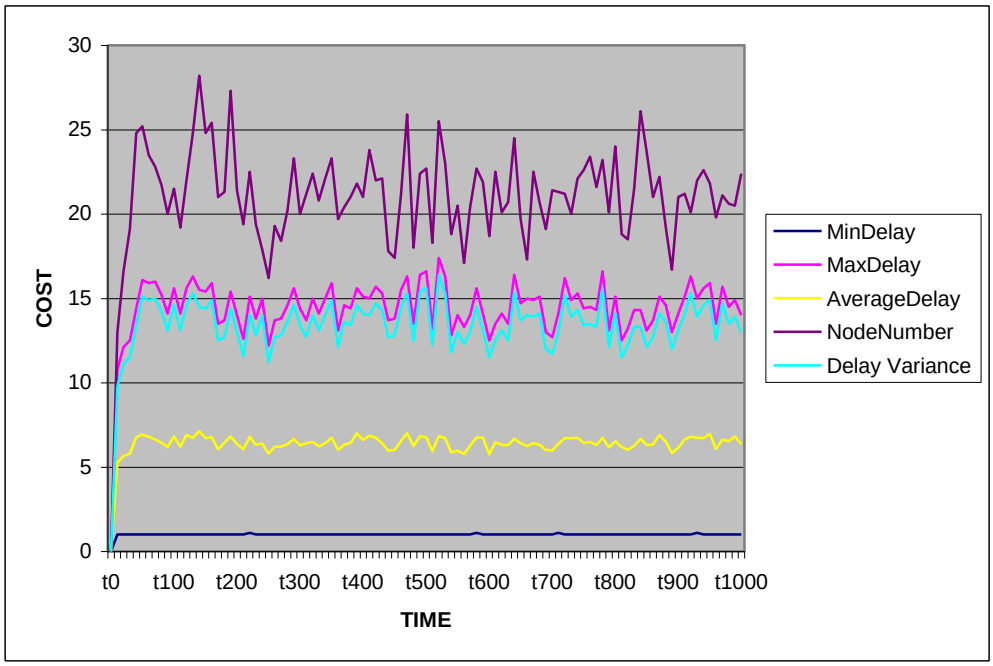
APPENDIX A



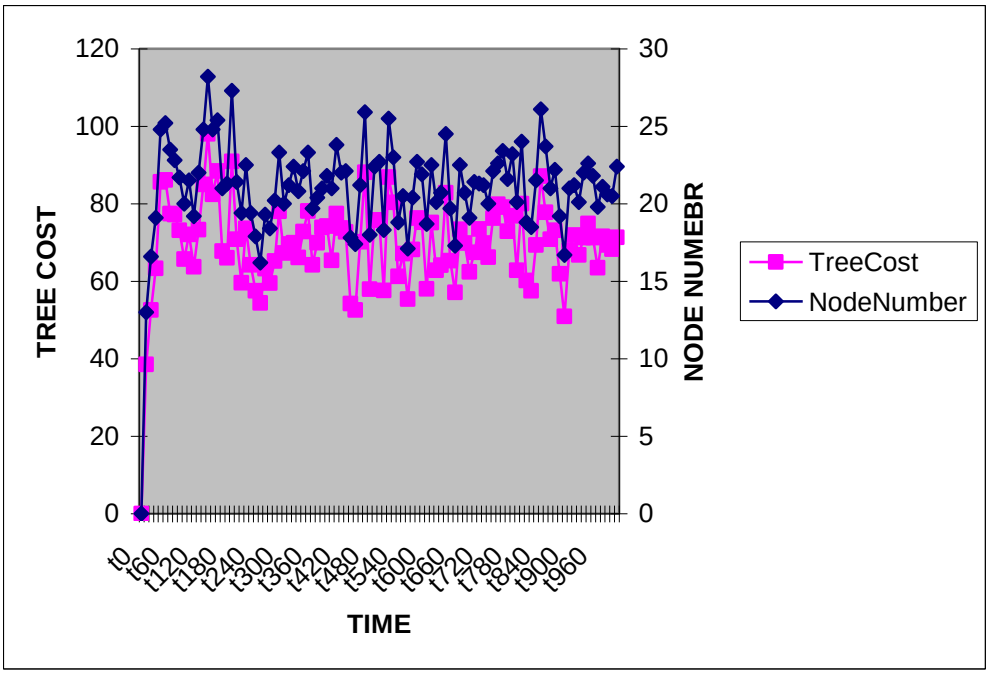
EK-A 1: PIM-SM, Flat Random Network, 50 nodes with α=0.5, Scenario Type : RC(Relay Chat)



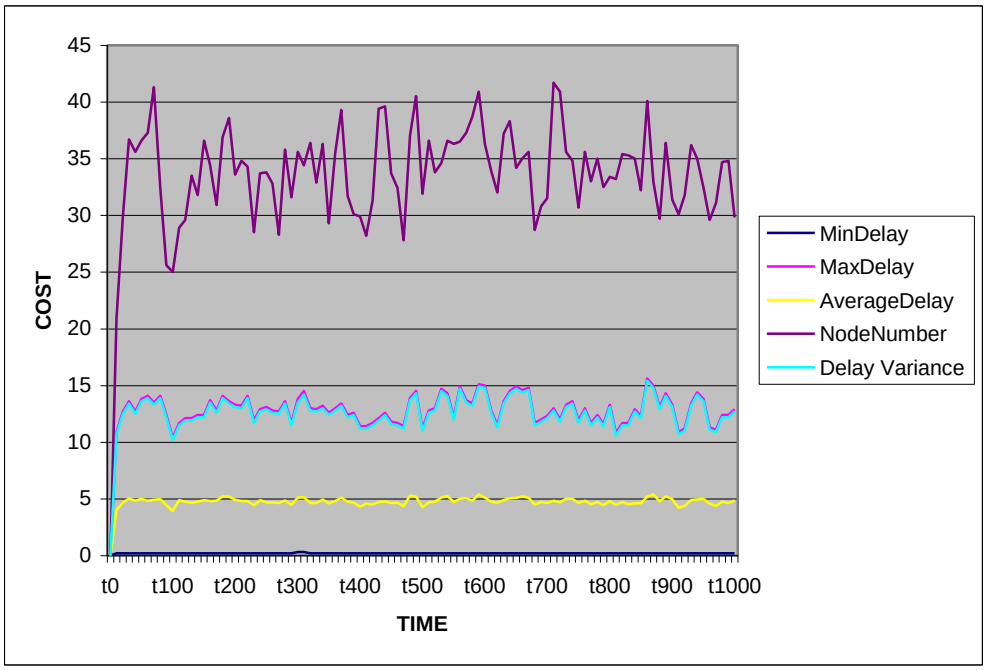
EK-A 2: PIM-SM, Flat Random Network, 50 nodes with α=0.5, Scenario Type : RC(Relay Chat)



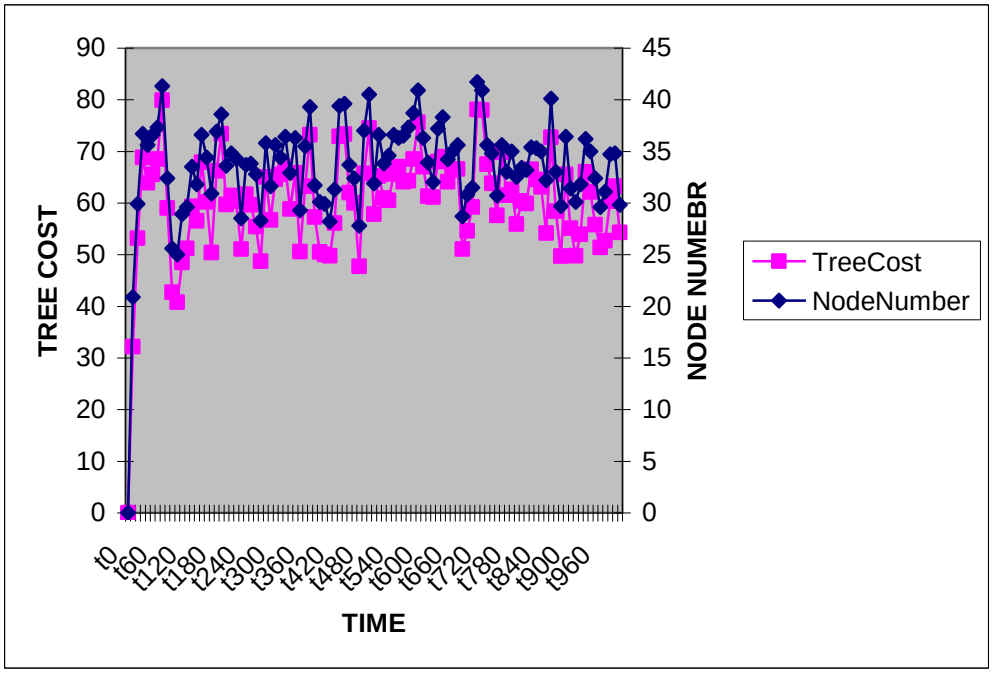
EK-A 3: PIM-SM, Flat Random Network, 50 nodes with α=1, Scenario Type : RC(Relay Chat)



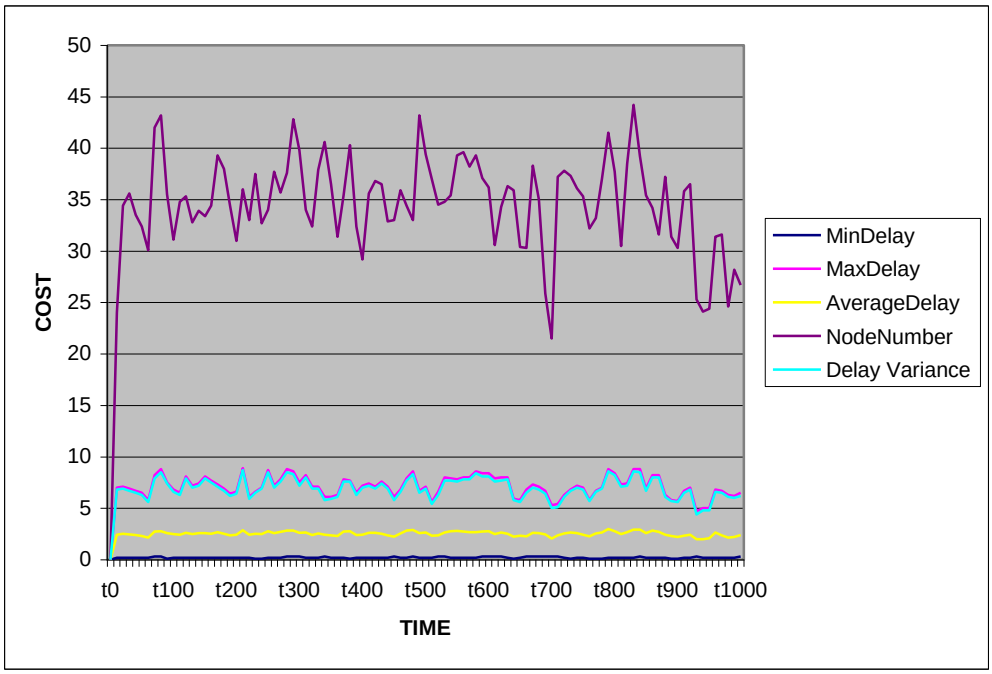
EK-A 4: PIM-SM, Flat Random Network, 50 nodes with α=1, Scenario Type : RC(Relay Chat)



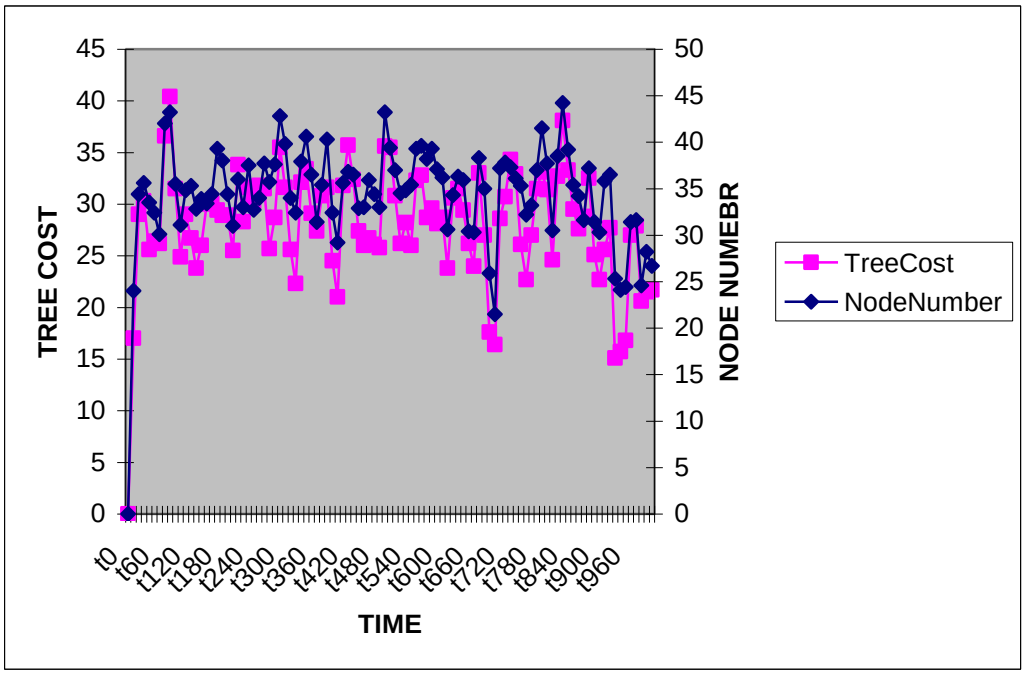
EK-A 5: PIM-SM, Flat Random Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat)



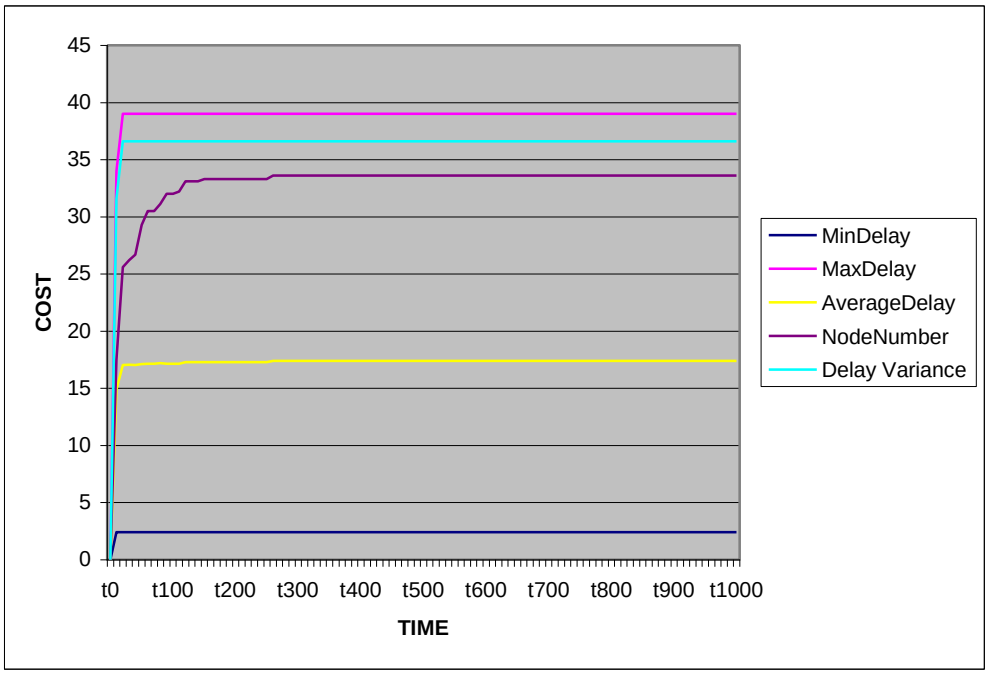
EK-A 6: PIM-SM, Flat Random Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat)



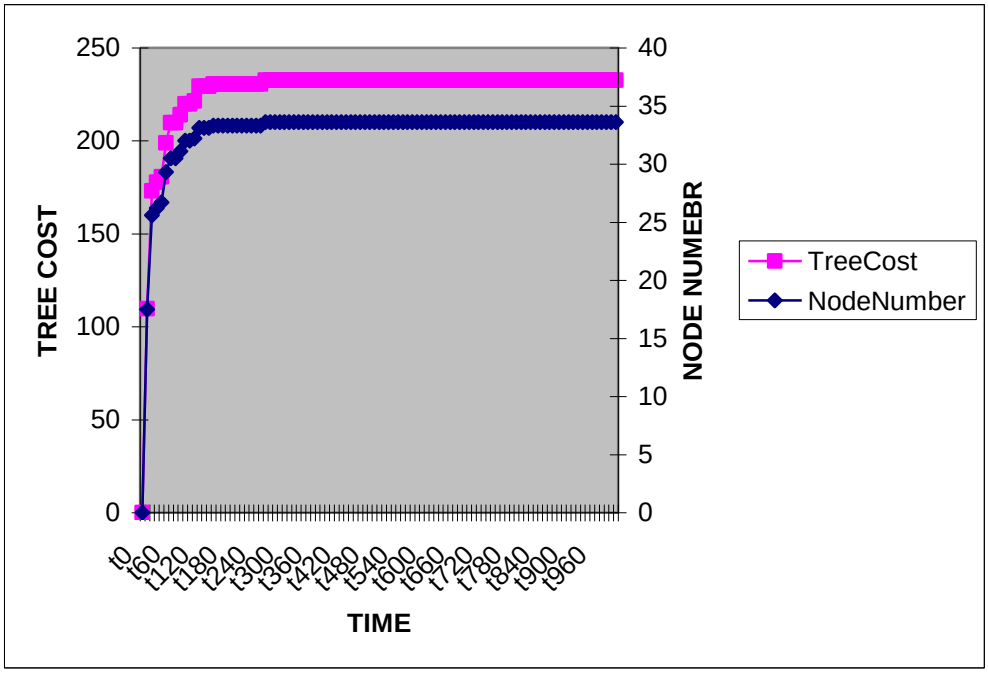
EK-A 7: PIM-SM, Flat Random Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat)



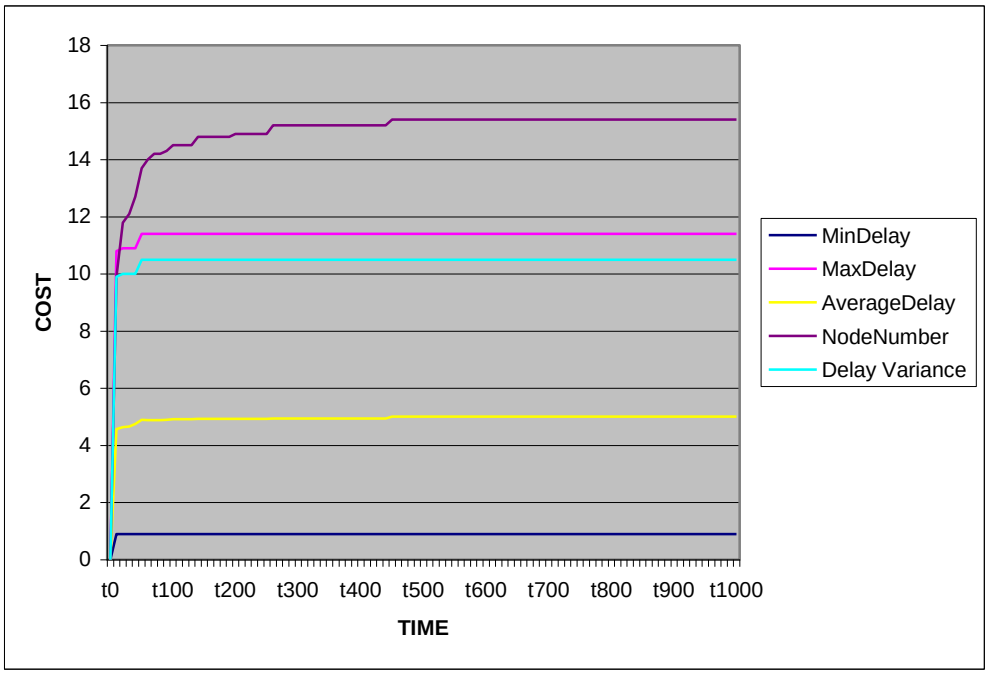
EK-A 8: PIM-SM, Flat Random Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat)



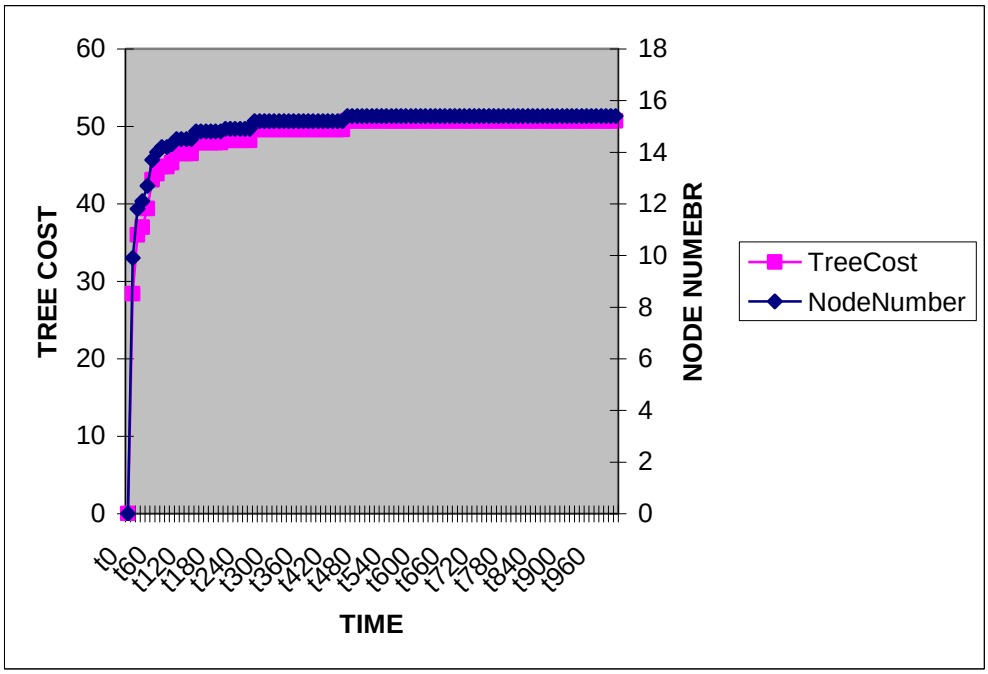
EK-A 9: PIM-SM, Flat Random Network, 50 nodes with α=0.5, Scenario Type:VC(Video Conference)



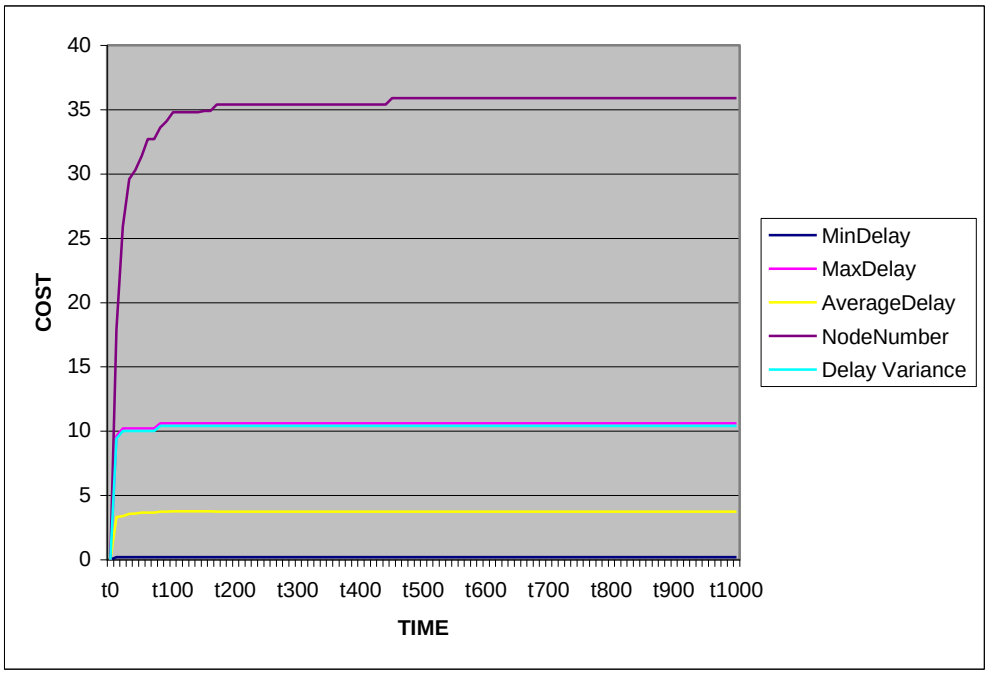
EK-A 10: PIM-SM, Flat Random Network, 50 nodes with α=0.5, Scenario Type:VC(Video Conference)



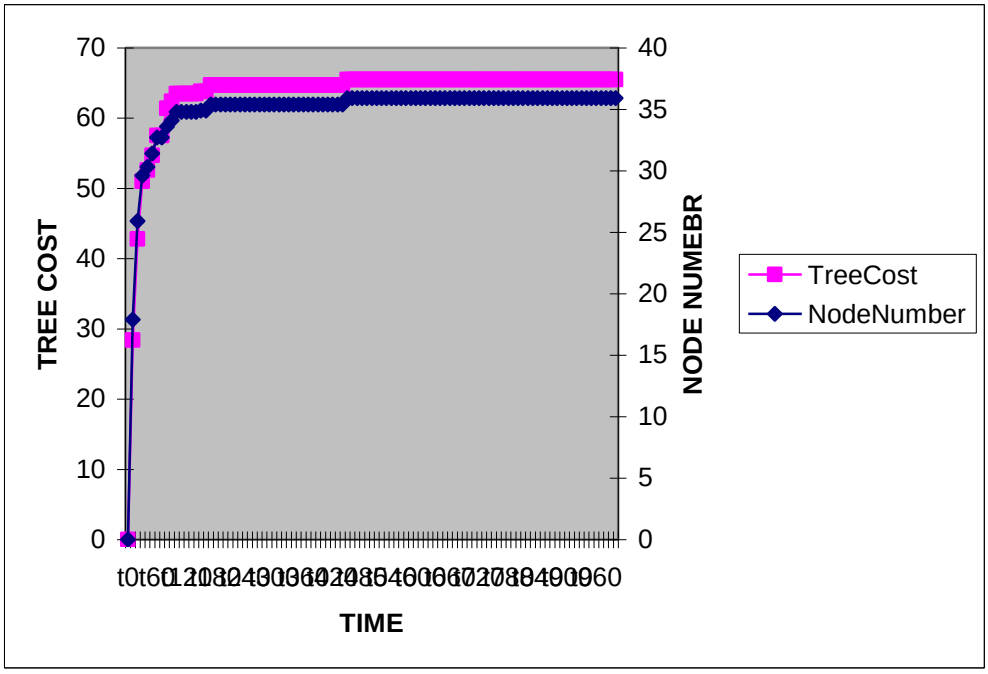
EK-A 11: PIM-SM, Flat Random Network, 50 nodes with α=1, Scenario Type:VC(Video Conference)



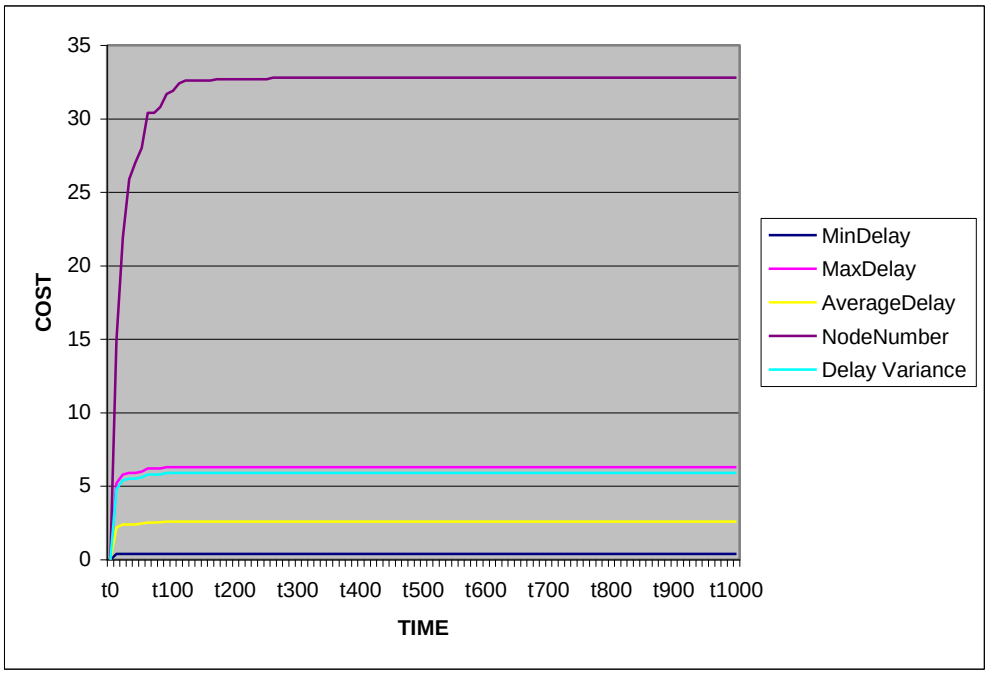
EK-A 12: PIM-SM, Flat Random Network, 50 nodes with α=1, Scenario Type:VC(Video Conference)



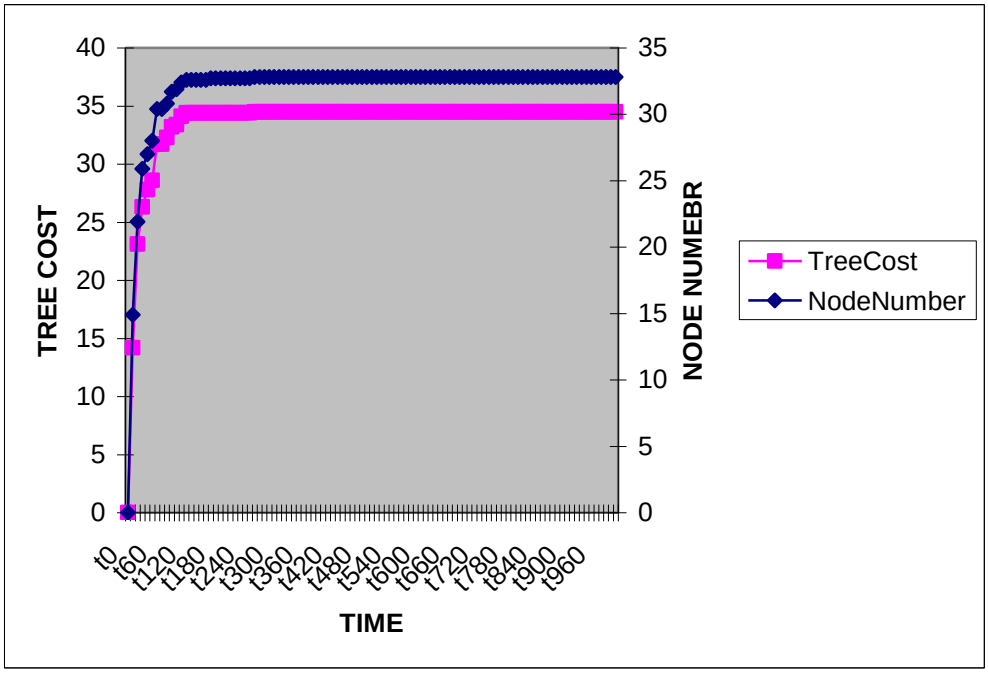
EK-A 13: PIM-SM, Flat Random Network, 100 nodes with α=0.5, Scenario Type:VC(Video Conference)



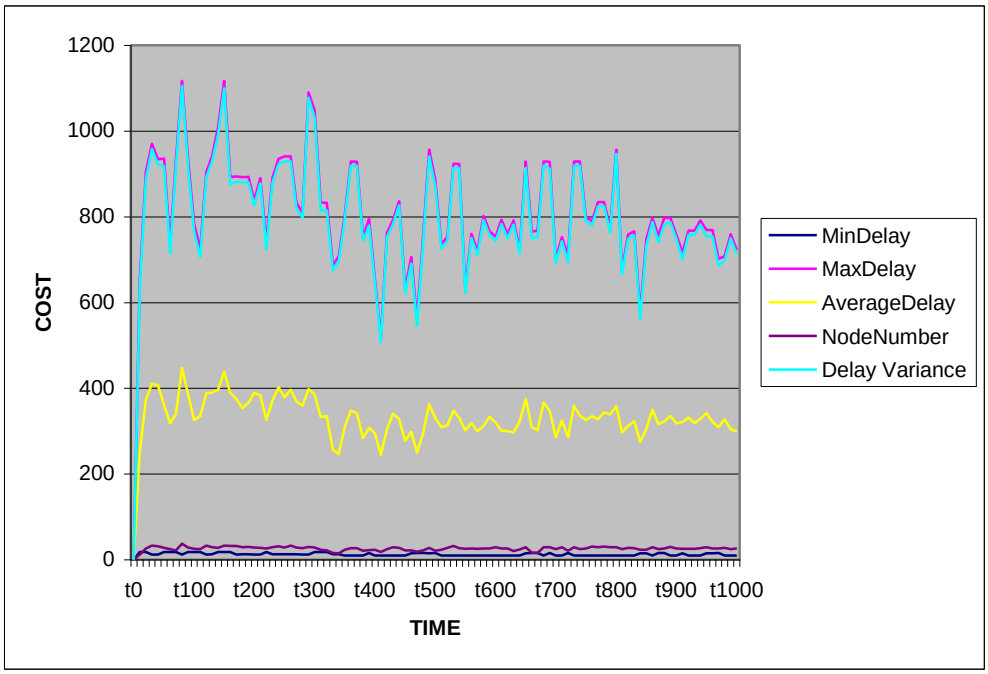
EK-A 14: PIM-SM, Flat Random Network, 100 nodes with α=0.5, Scenario Type:VC(Video Conference)



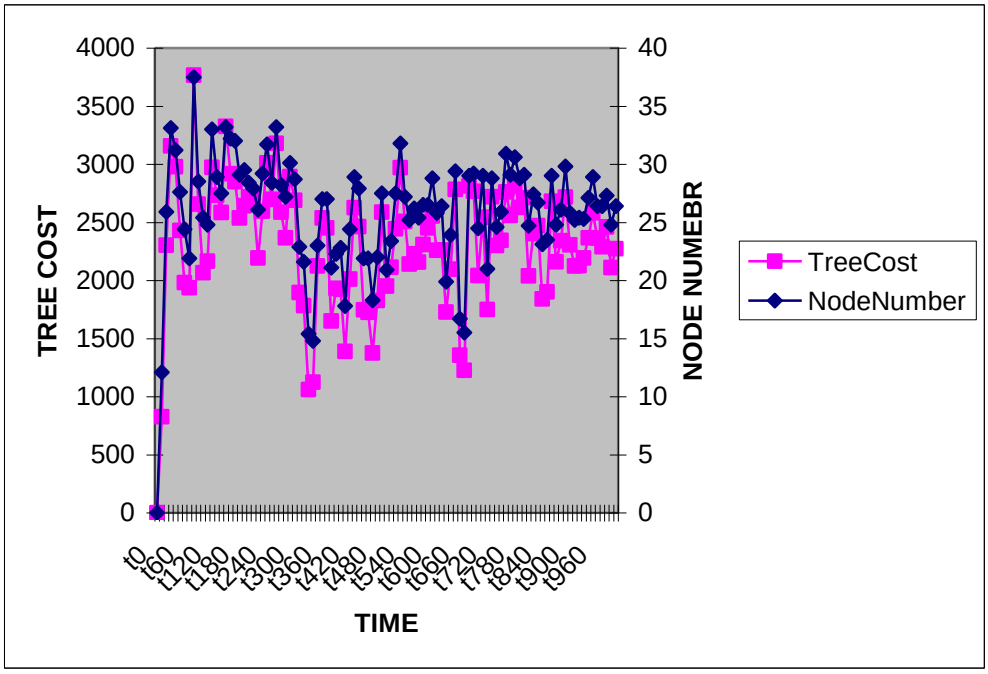
EK-A 15: PIM-SM, Flat Random Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)



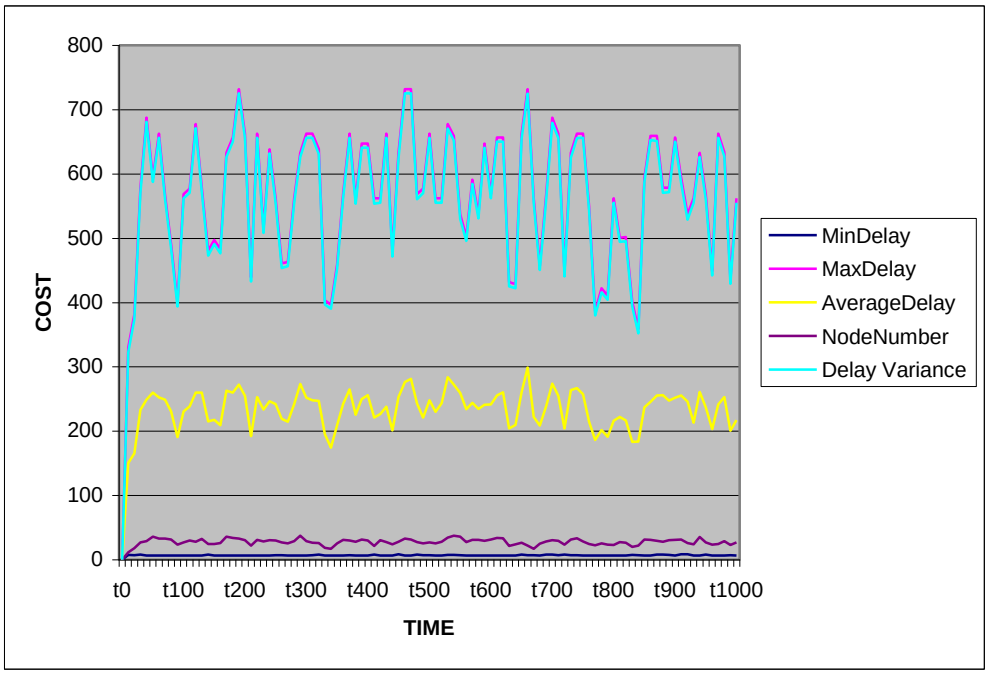
EK-A 16: PIM-SM, Flat Random Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)



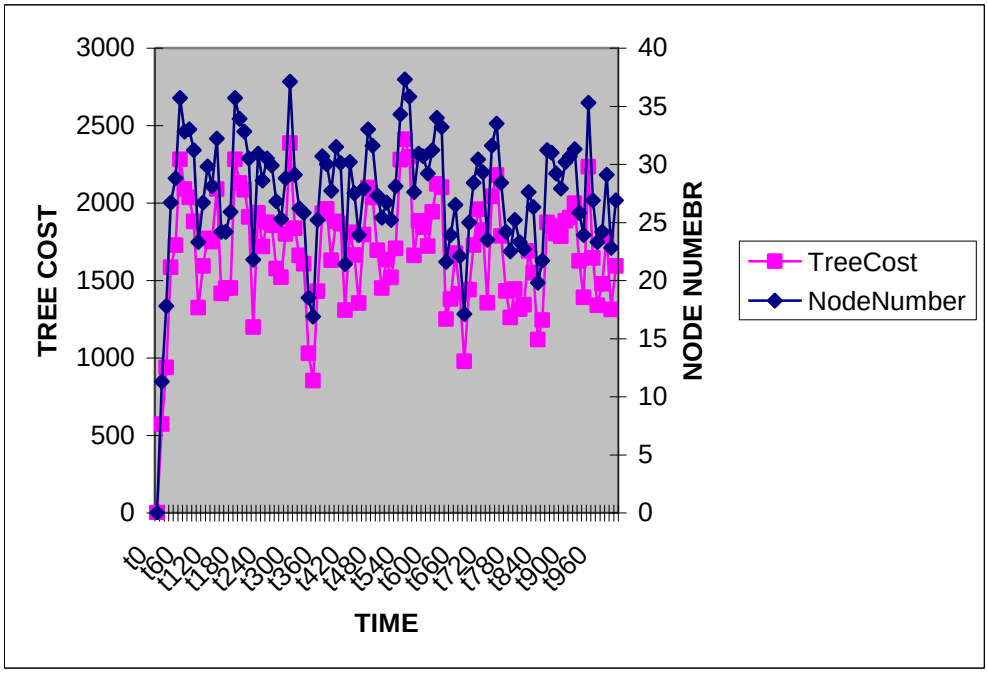
EK-A 17: PIM-SM, Transit-Stub Network, 50 nodes with α=0.5, Scenario Type:RC(Relay Chat)



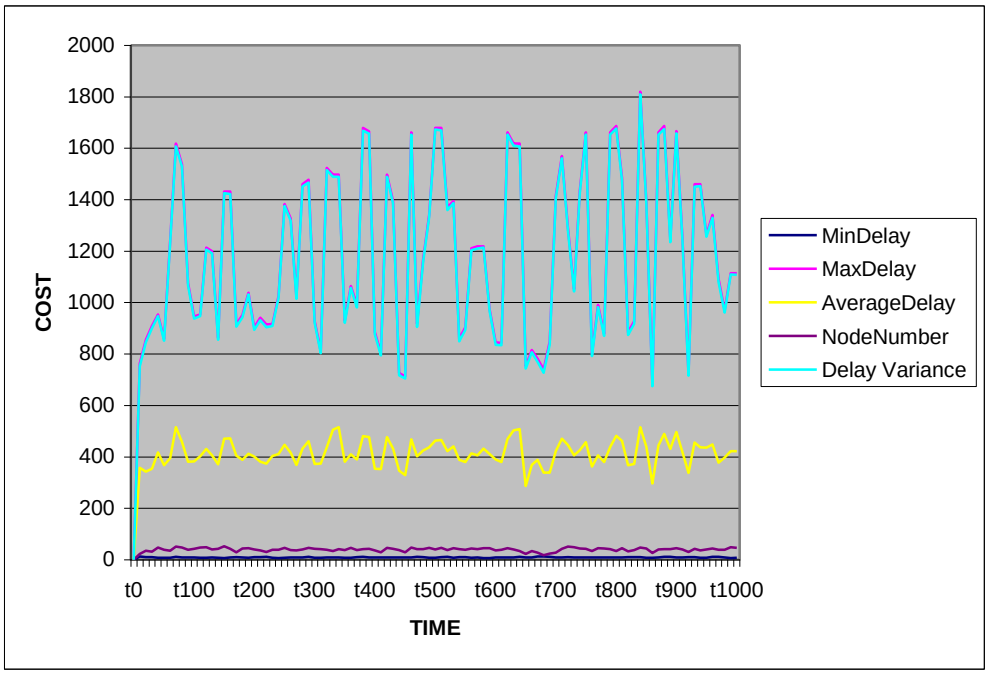
EK-A 18: PIM-SM, Transit-Stub Network, 50 nodes with α=0.5, Scenario Type:RC(Relay Chat)



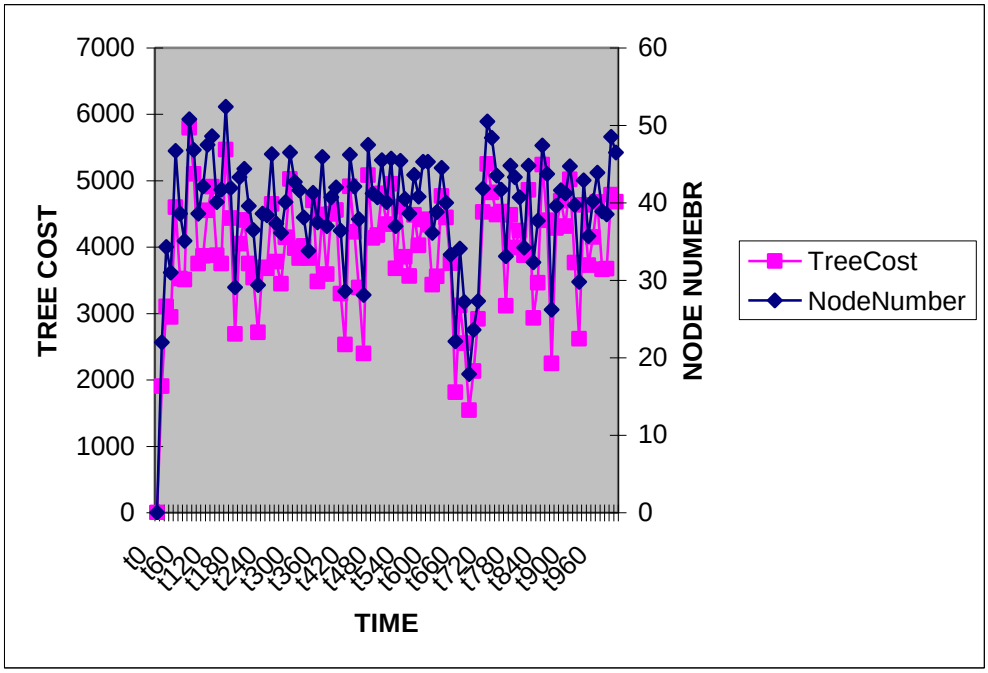
EK-A 19: PIM-SM, Transit-Stub Network, 50 nodes with α=1, Scenario Type:RC(Relay Chat)



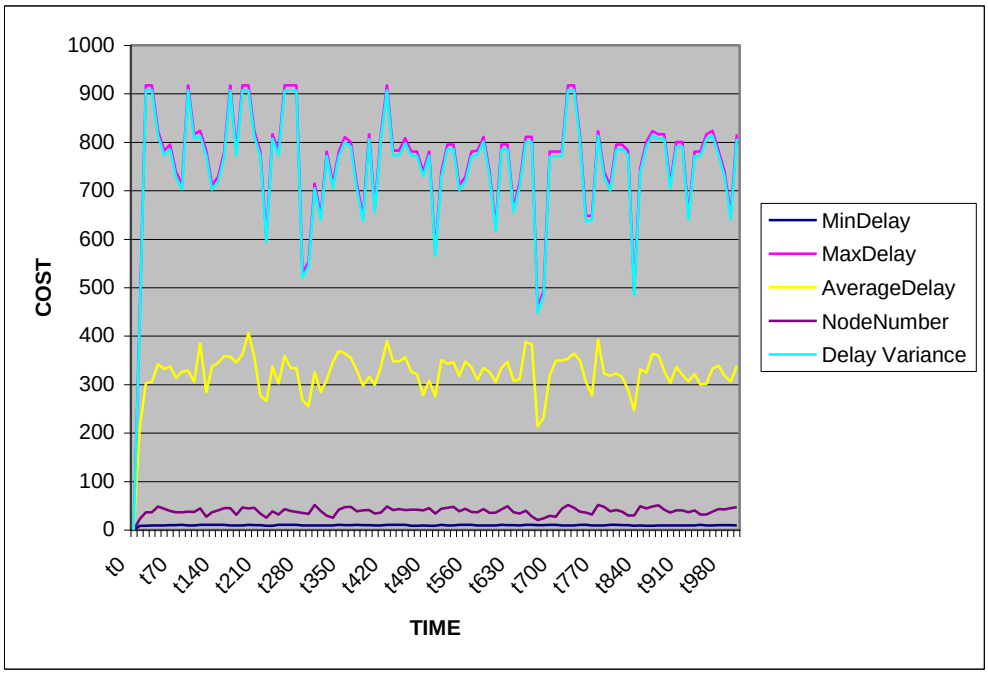
EK-A 20: PIM-SM, Transit-Stub Network, 50 nodes with α=1, Scenario Type:RC(Relay Chat)



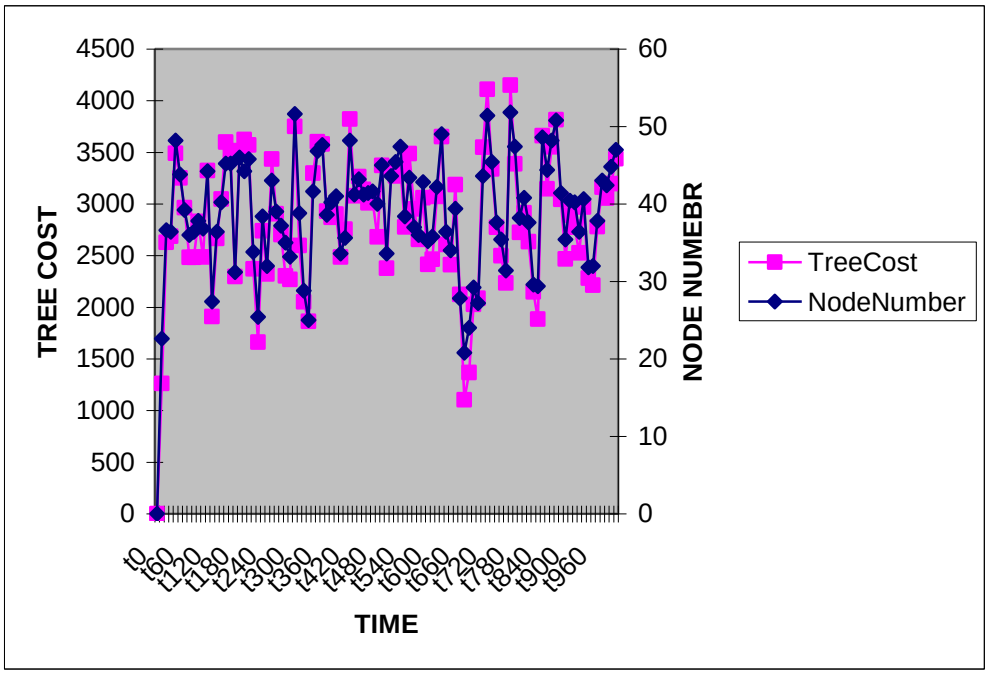
EK-A 21: PIM-SM, Transit-Stub Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat)



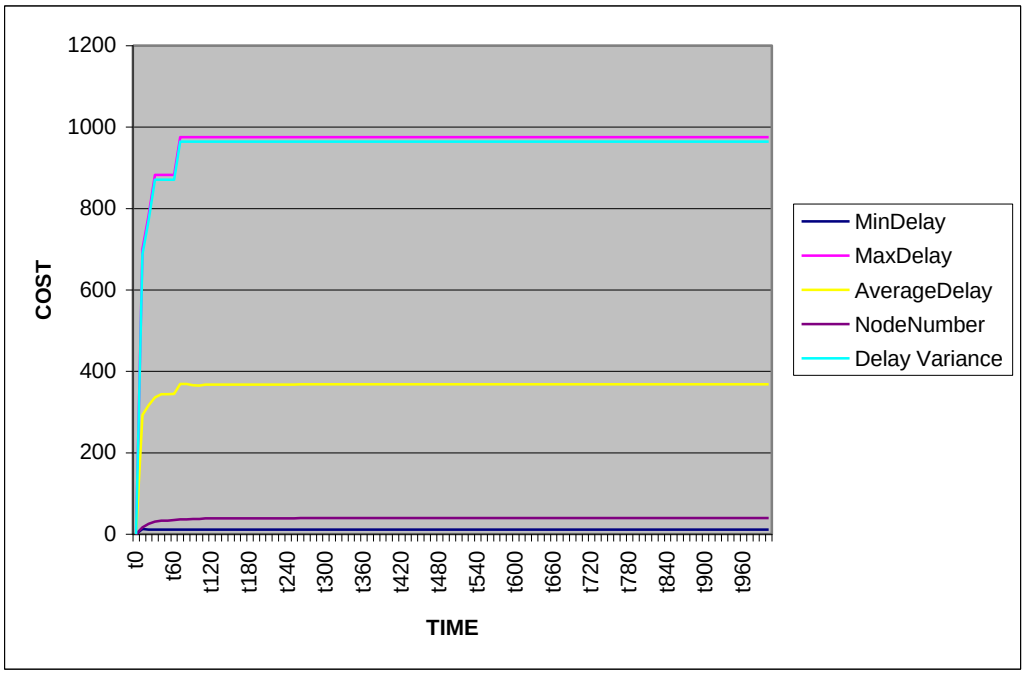
EK-A 22: PIM-SM, Transit-Stub Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat)



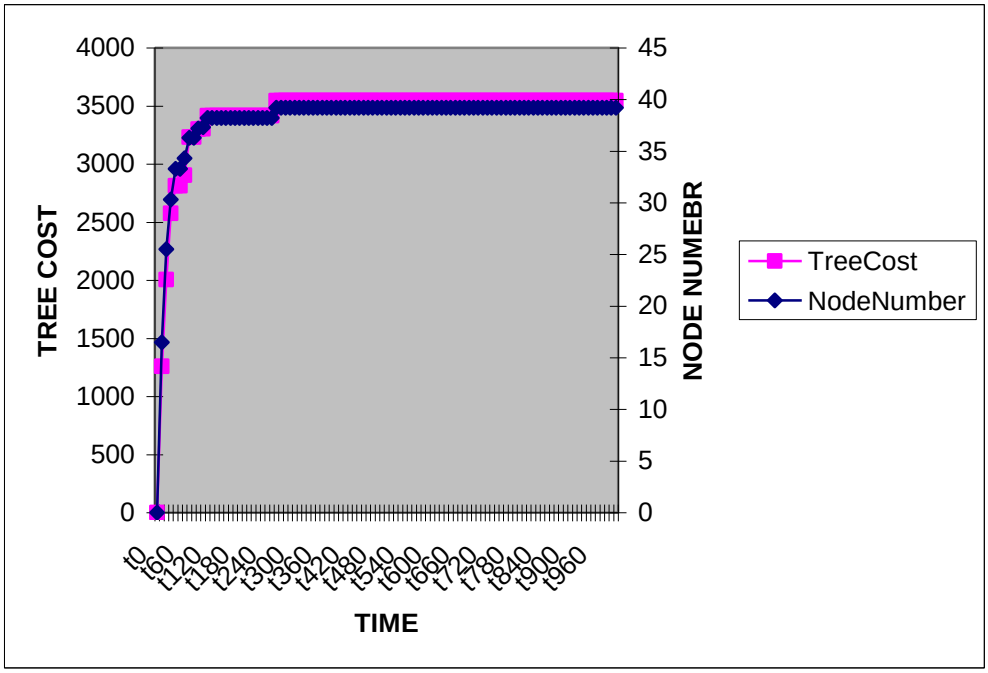
EK-A 23: PIM-SM, Transit-Stub Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat)



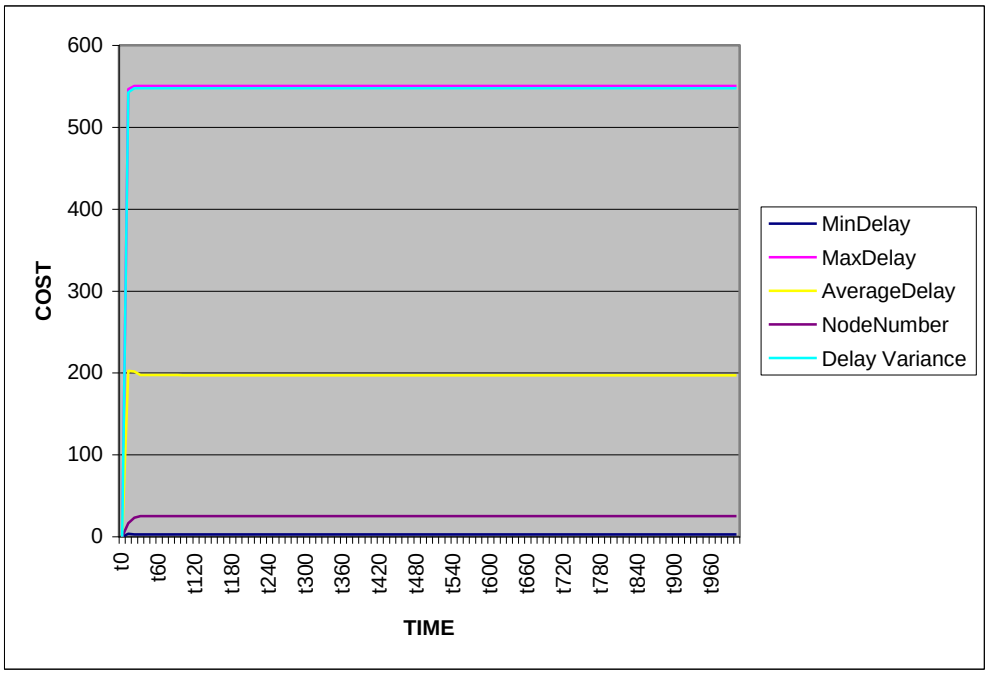
EK-A 24: PIM-SM, Transit-Stub Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat)



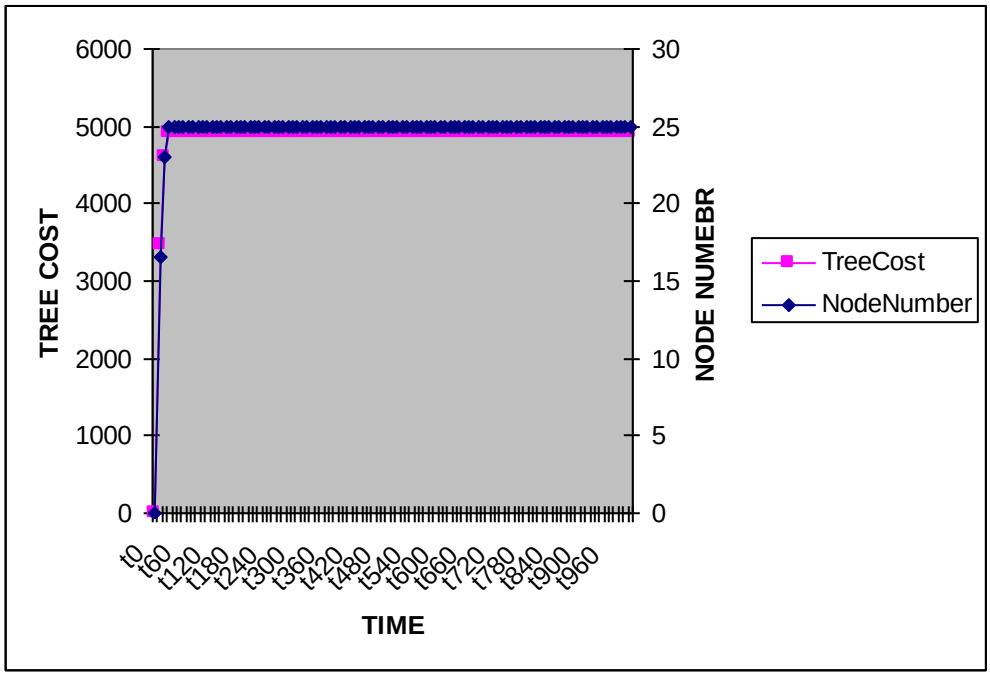
EK-A 25: PIM-SM, Transit-Stub Network, 50 nodes with α=0.5, Scenario Type:VC(Video Conference)



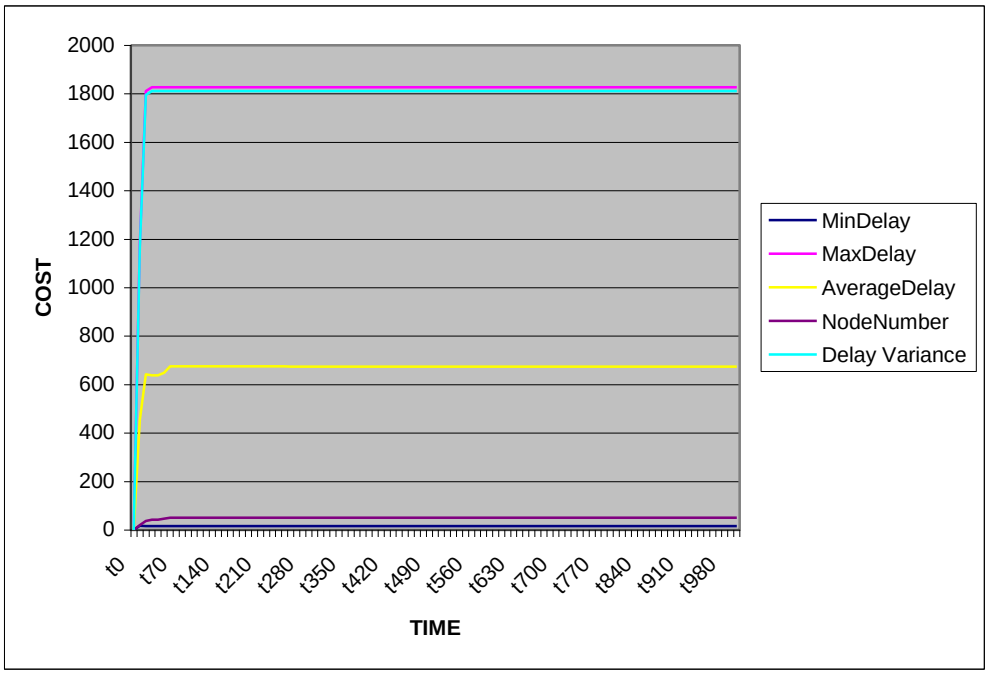
EK-A 26: PIM-SM, Transit-Stub Network, 50 nodes with α=0.5, Scenario Type:VC(Video Conference)



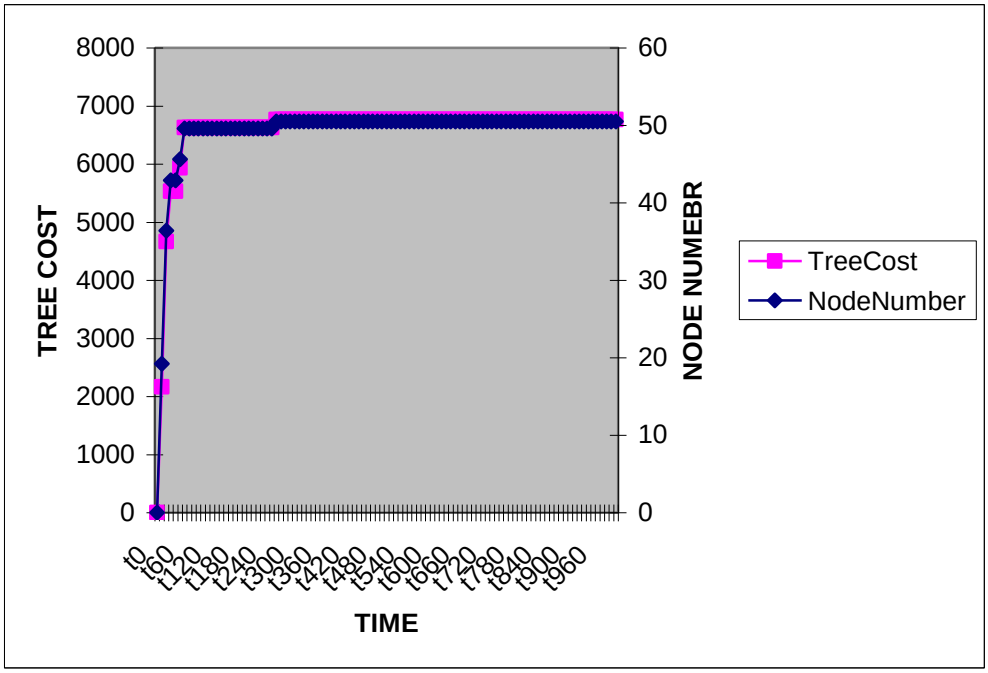
EK-A 27: PIM-SM, Transit-Stub Network, 50 nodes with α=1, Scenario Type:VC(Video Conference)



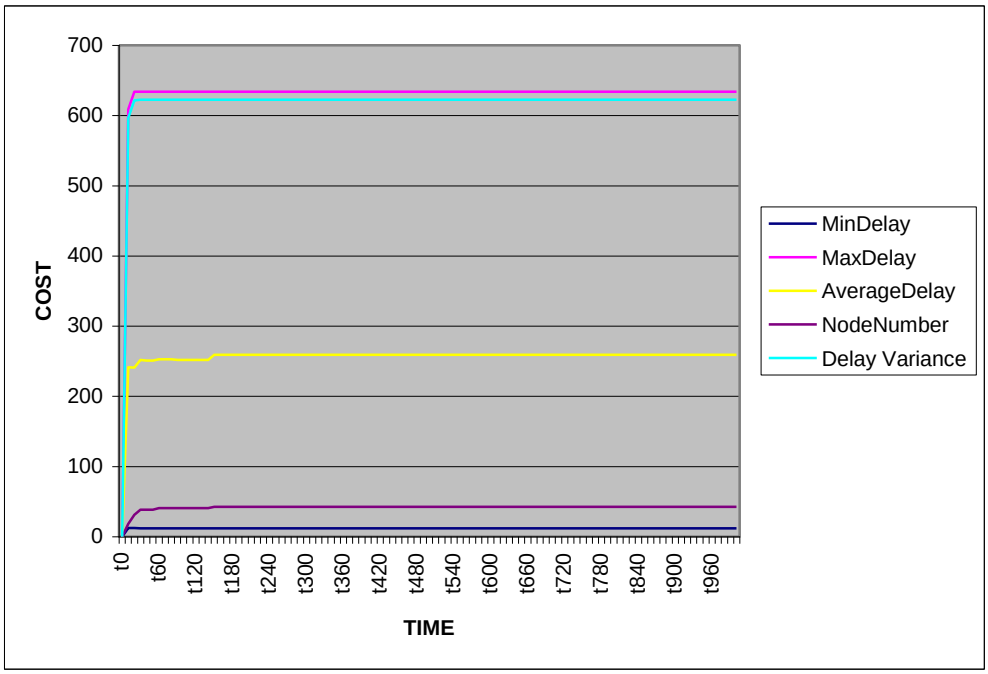
EK-A 28: PIM-SM, Transit-Stub Network, 50 nodes with α=1, Scenario Type:VC(Video Conference)



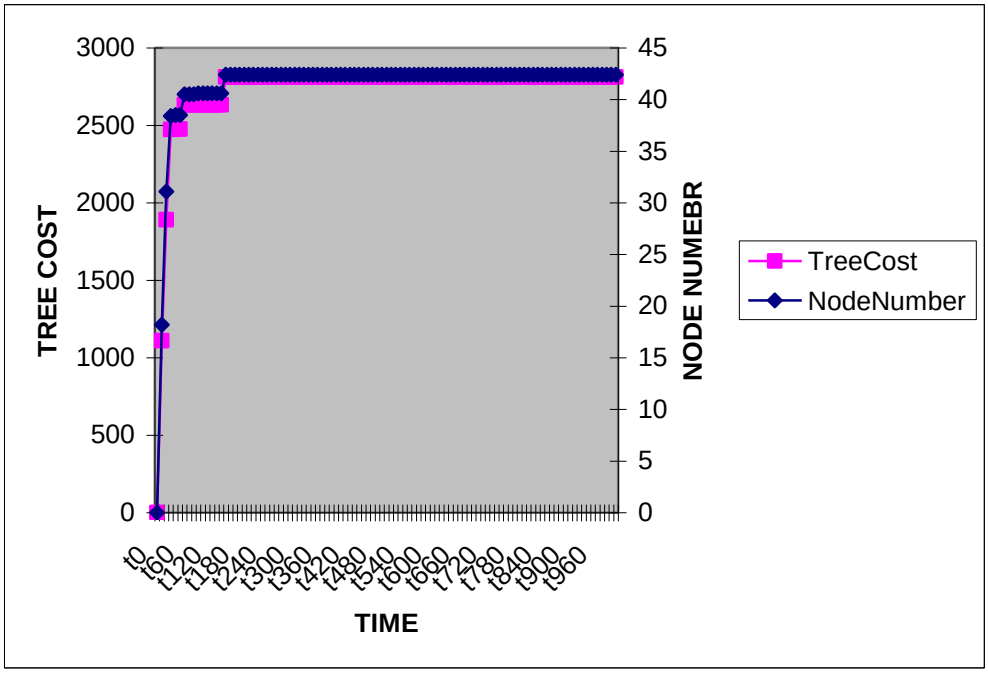
EK-A 29: PIM-SM, Transit-Stub Network, 100 nodes with α=0.5, Scenario Type:VC(Video Conference)



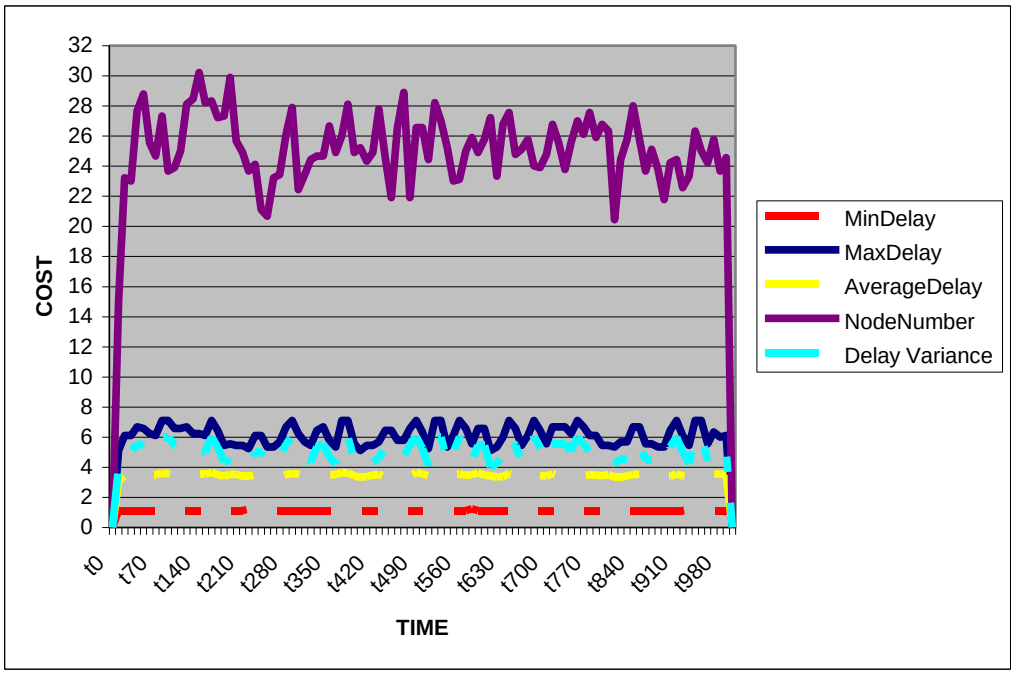
EK-A 30: PIM-SM, Transit-Stub Network, 100 nodes with α=0.5, Scenario Type:VC(Video Conference)



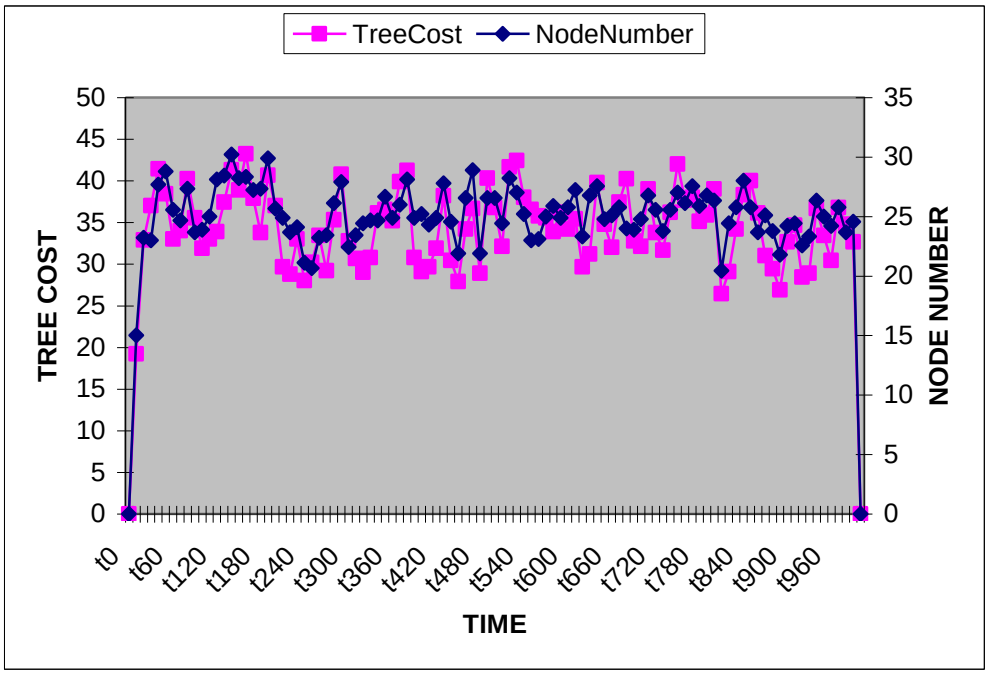
EK-A 31: PIM-SM, Transit-Stub Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)



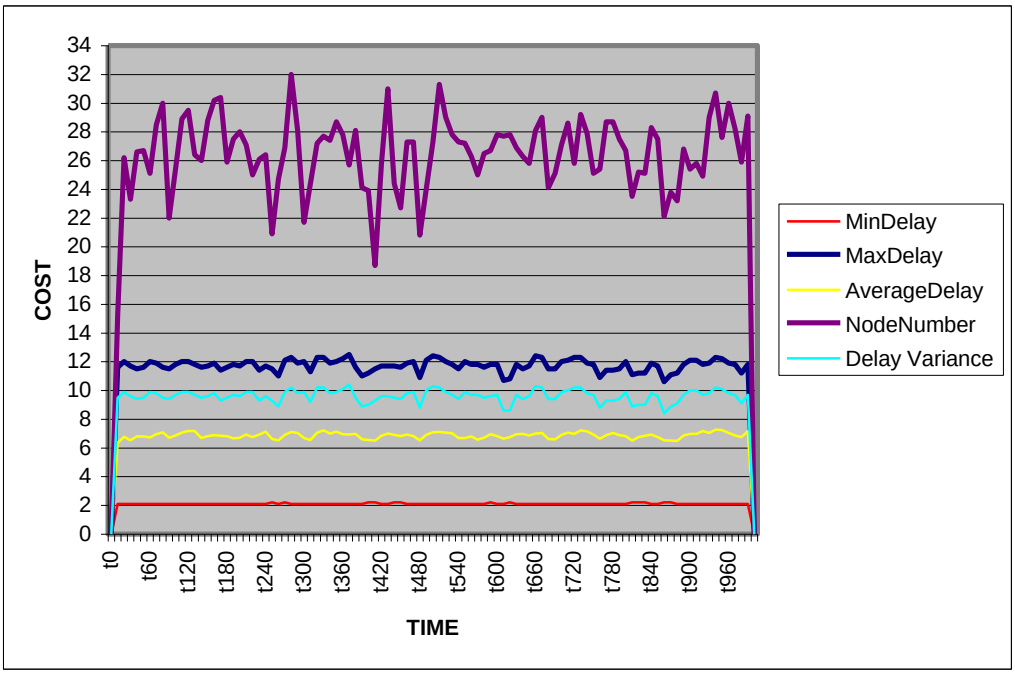
EK-A 32: PIM-SM, Transit-Stub Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)



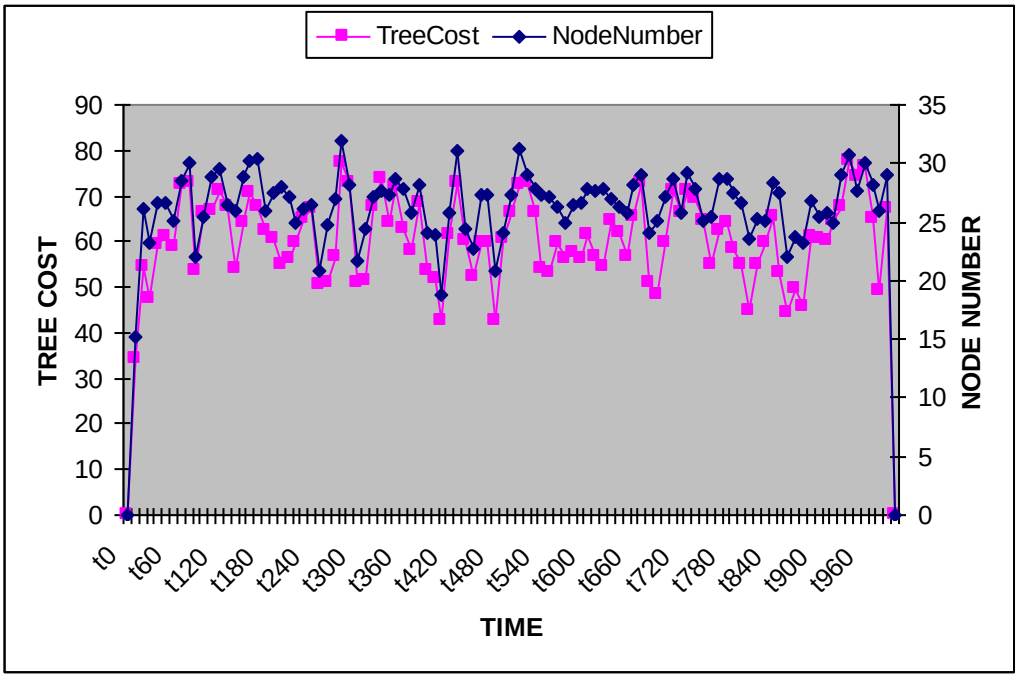
EK-A 33: SCMP, Flat Random Network, 50 nodes with α=1, Scenario Type:RC(Relay Chat)



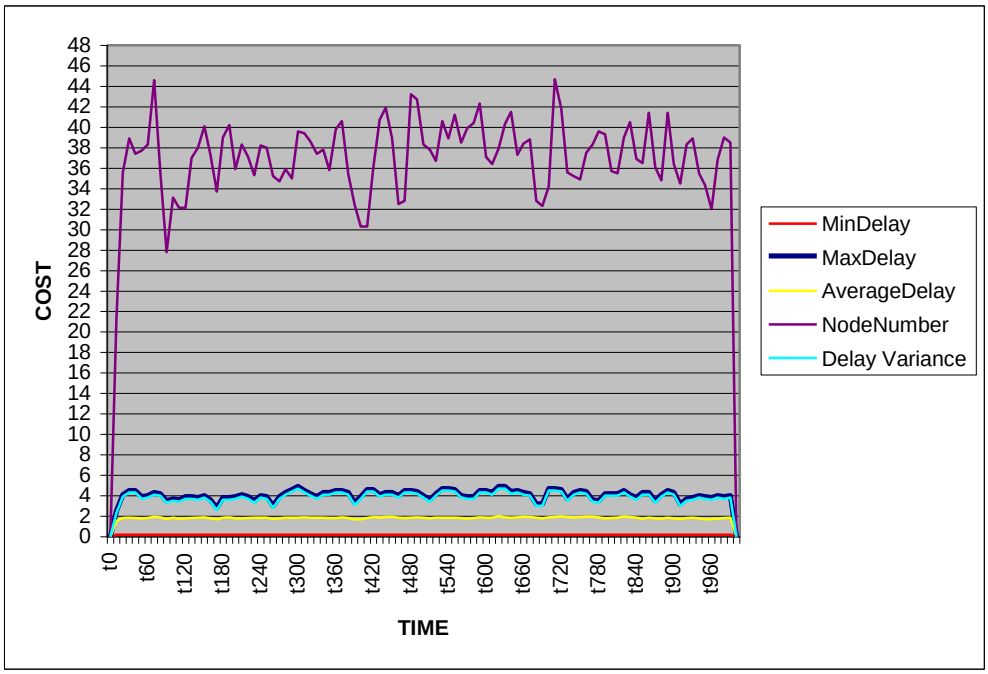
EK-A 34: SCMP, Flat Random Network, 50 nodes with α=1, Scenario Type:RC(Relay Chat)



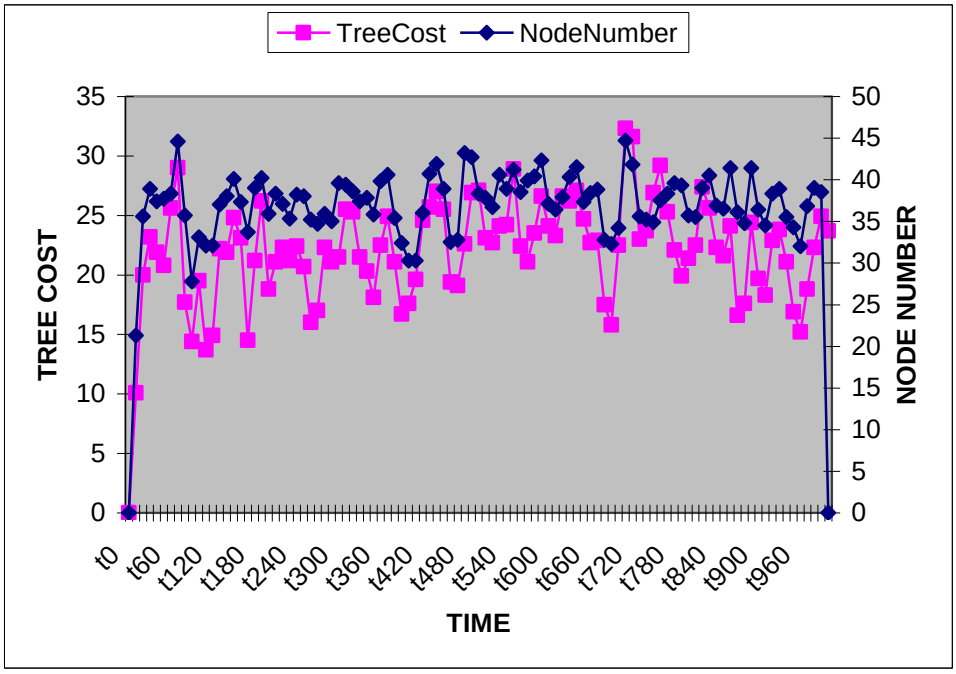
EK-A 35: SCMP, Flat Random Network, 50 nodes with α=0.5, Scenario Type:RC(Relay Chat)



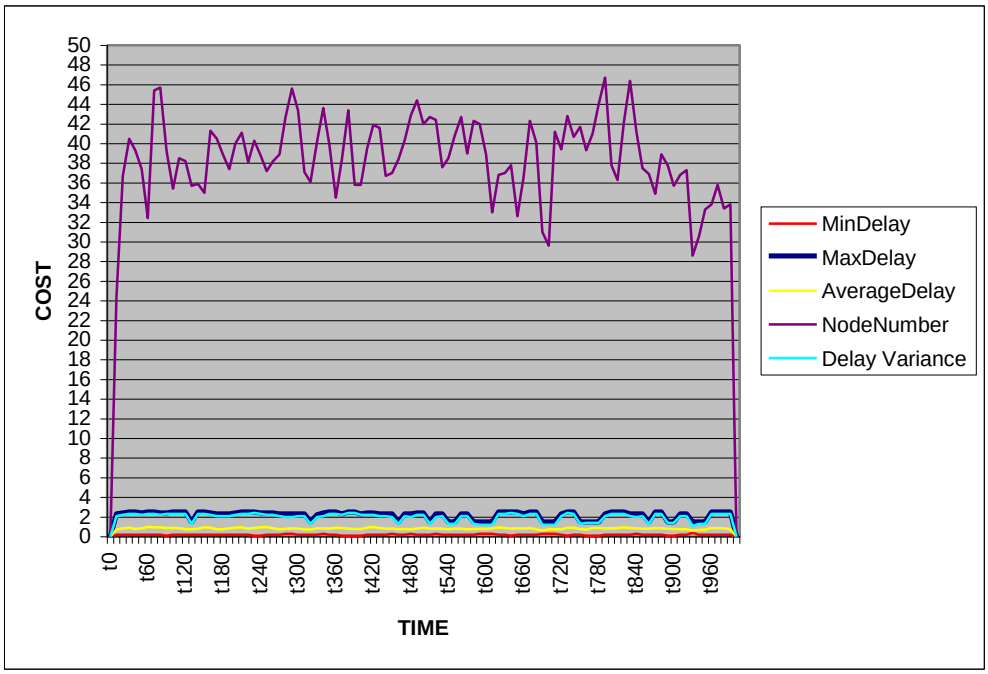
EK-A 36: SCMP, Flat Random Network, 50 nodes with α=0.5, Scenario Type:RC(Relay Chat)



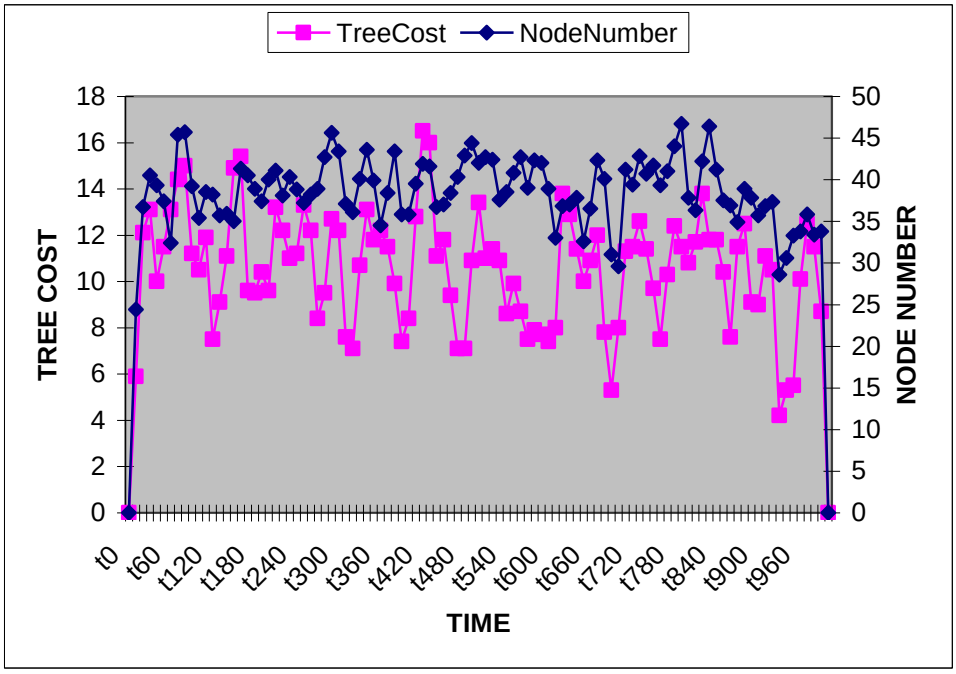
EK-A 37: SCMP, Flat Random Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat)



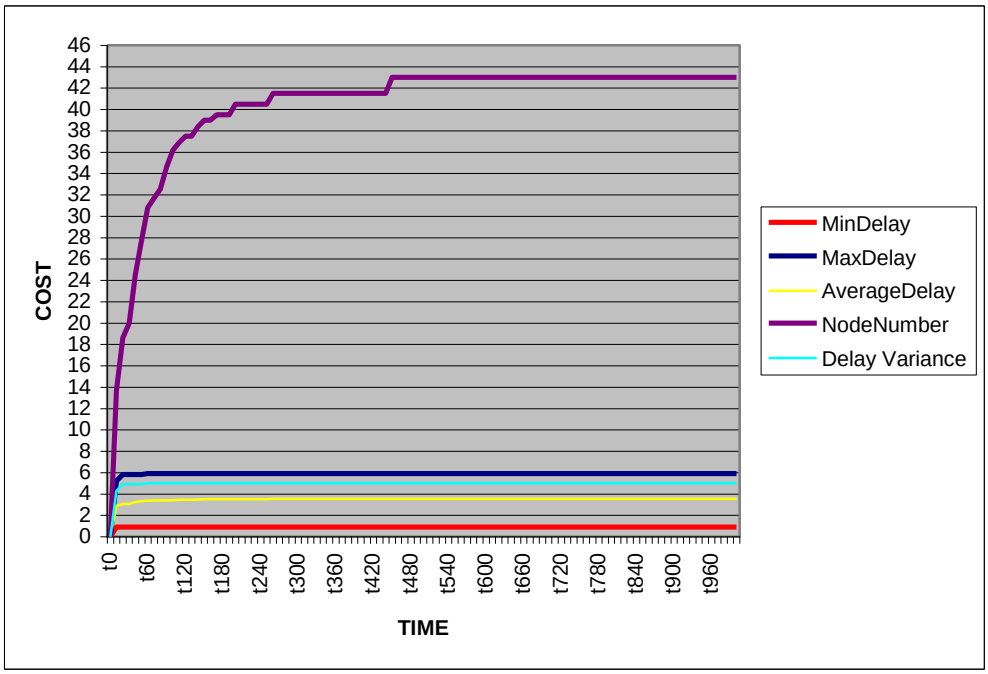
EK-A 38: SCMP, Flat Random Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat)



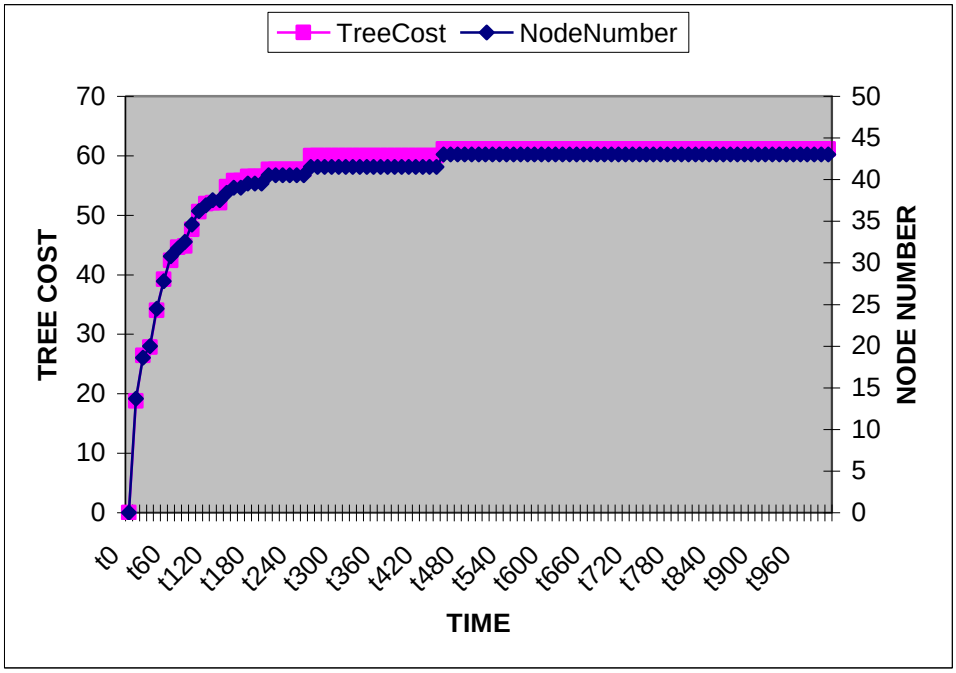
EK-A 39: SCMP, Flat Random Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat)



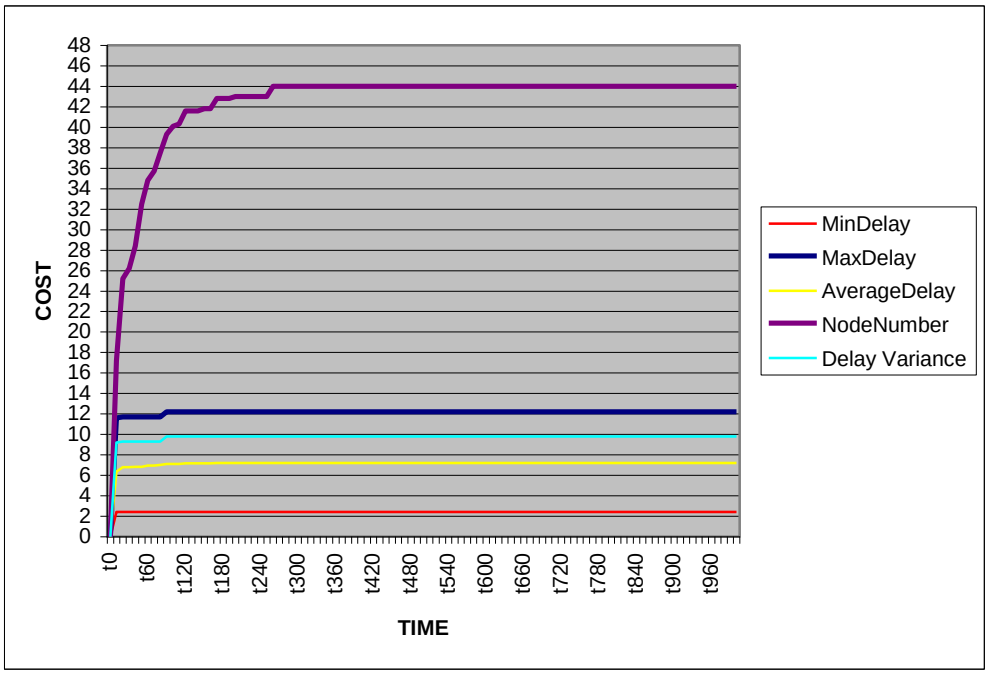
EK-A 40: SCMP, Flat Random Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat)



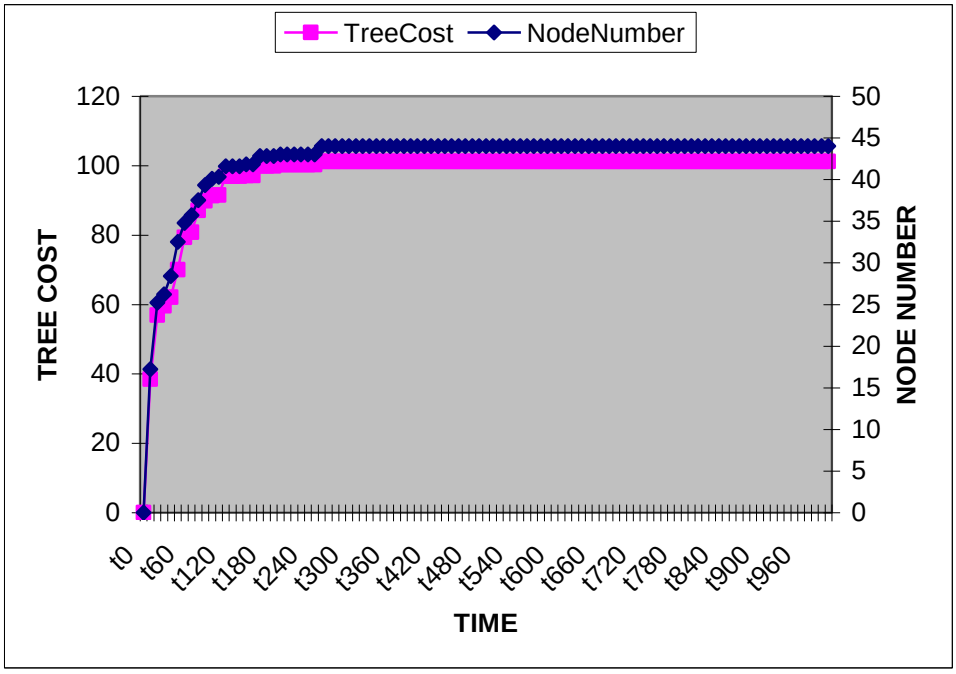
EK-A 41: SCMP, Flat Random Network, 50 nodes with α=1, Scenario Type:VC(Video Conference)



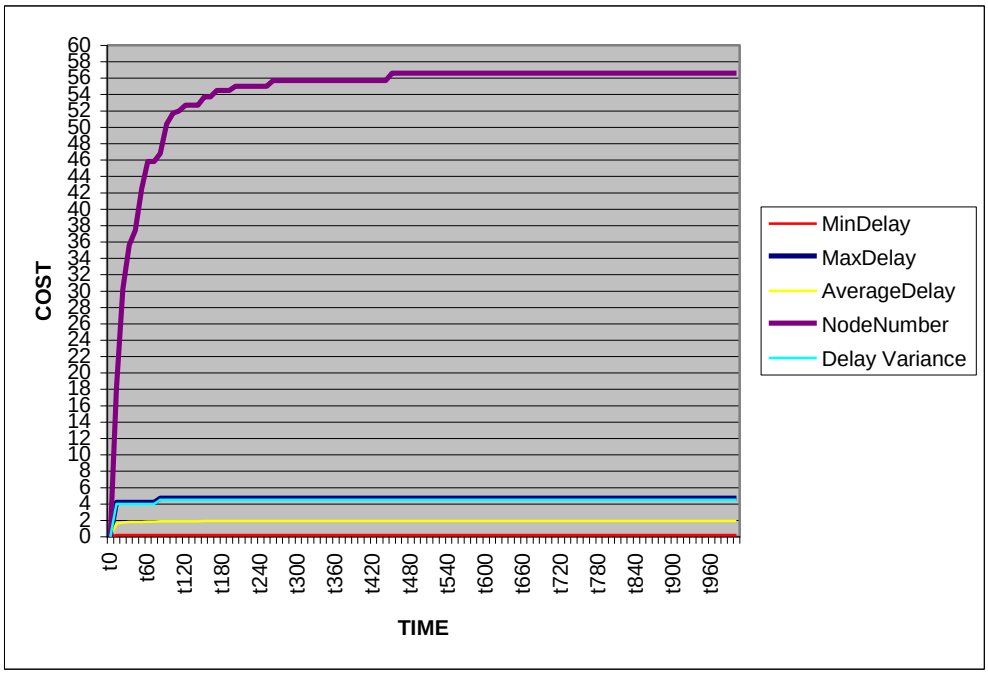
EK-A 42: SCMP, Flat Random Network, 50 nodes with α=1, Scenario Type:VC(Video Conference)



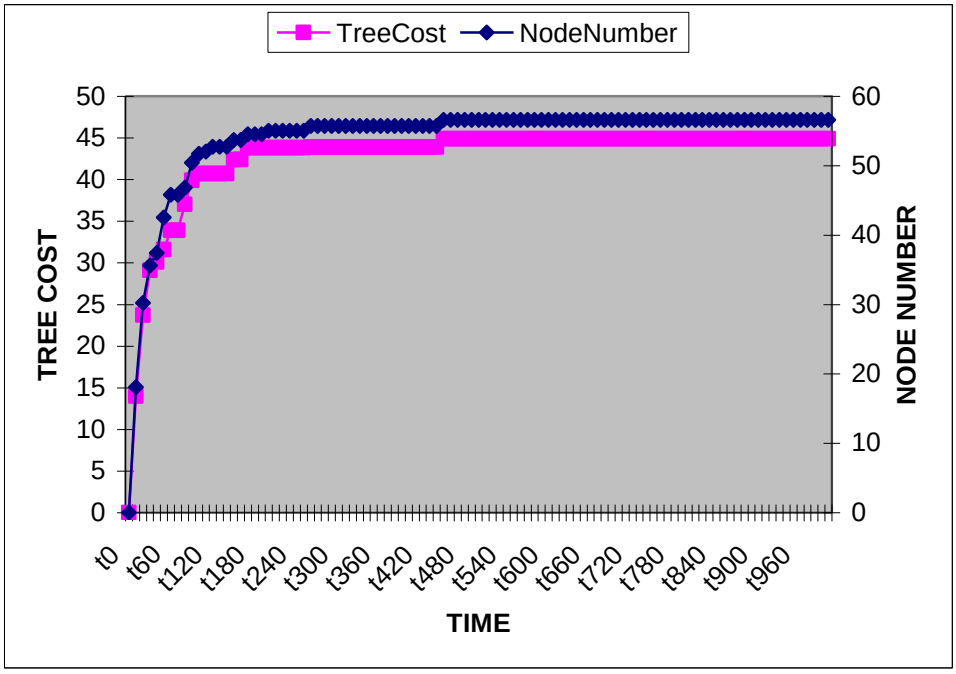
EK-A 43: SCMP, Flat Random Network, 50 nodes with α=0.5, Scenario Type:VC(Video Conference)



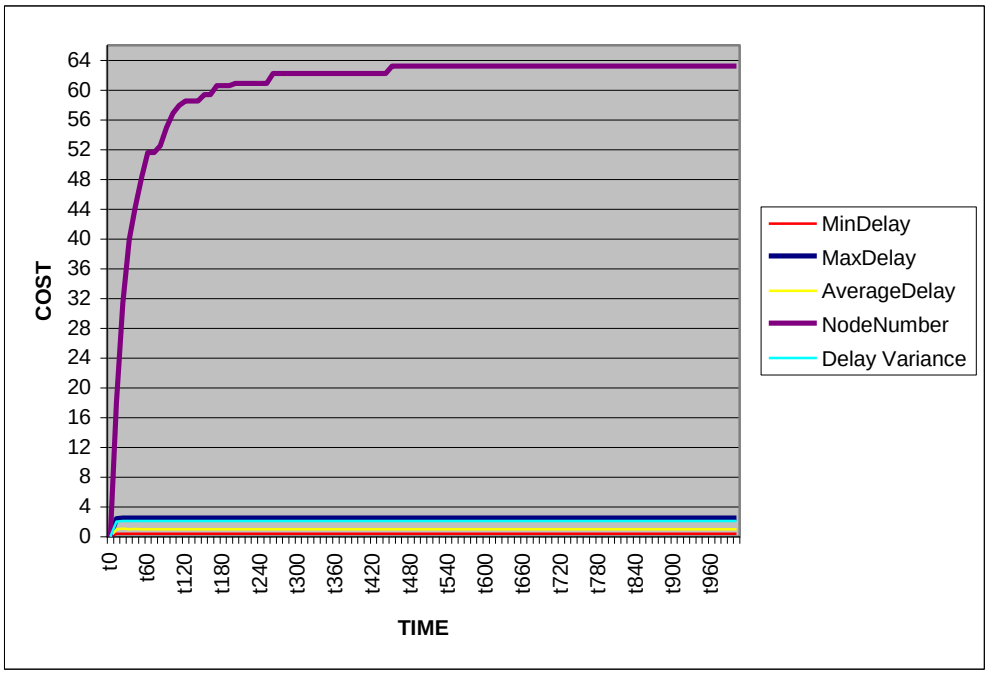
EK-A 44: SCMP, Flat Random Network, 50 nodes with α=0.5, Scenario Type:VC(Video Conference)



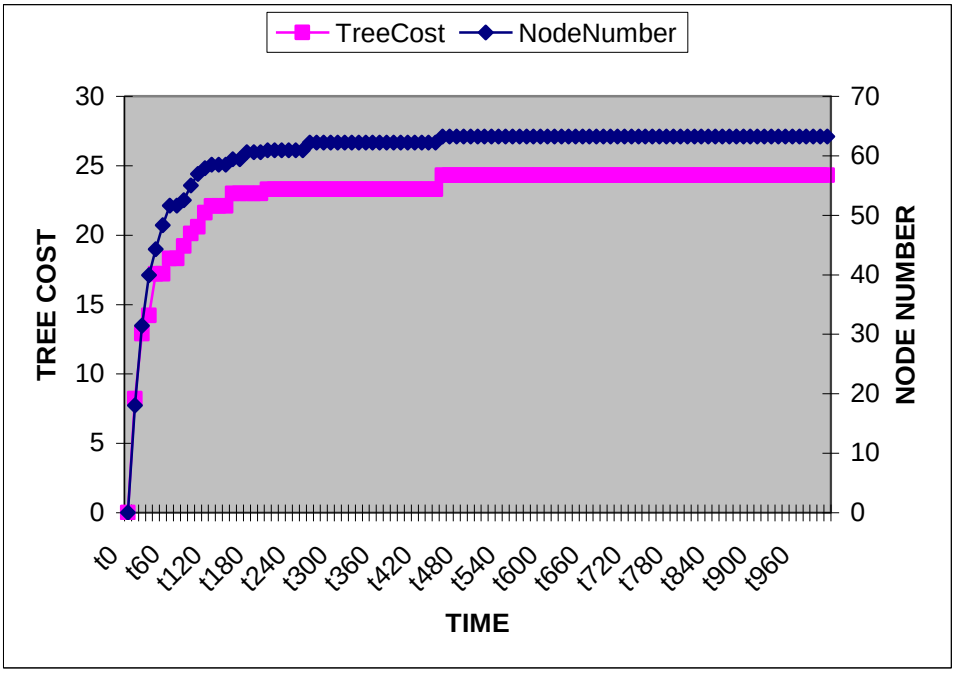
EK-A 45: SCMP, Flat Random Network, 100 nodes with α=0.5, Scenario Type:VC(Video Conference)



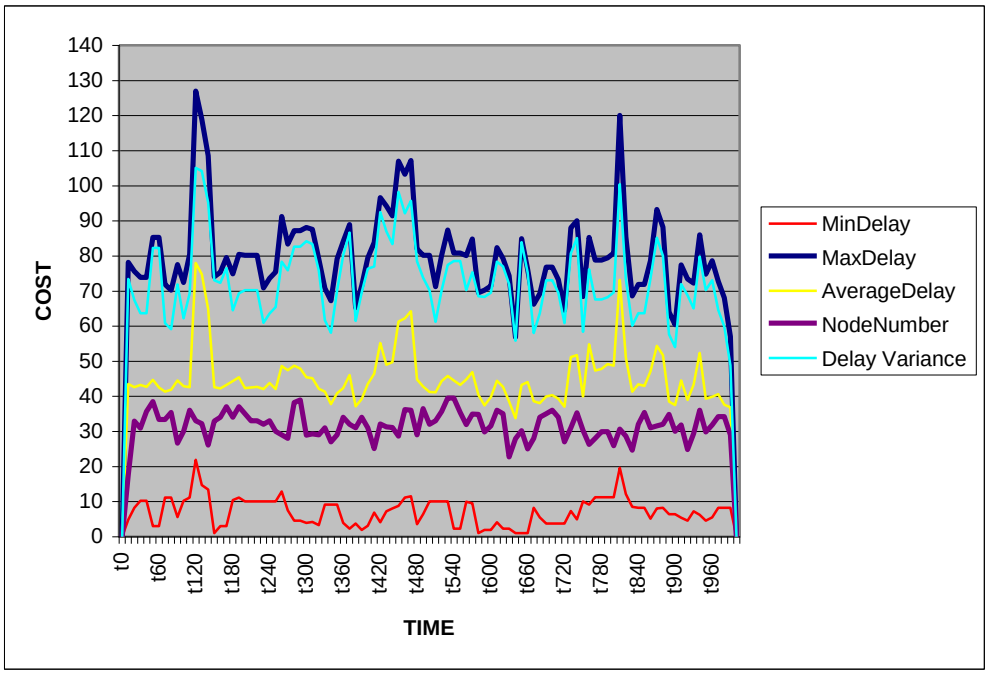
EK-A 46: SCMP, Flat Random Network, 100 nodes with α=0.5, Scenario Type:VC(Video Conference)



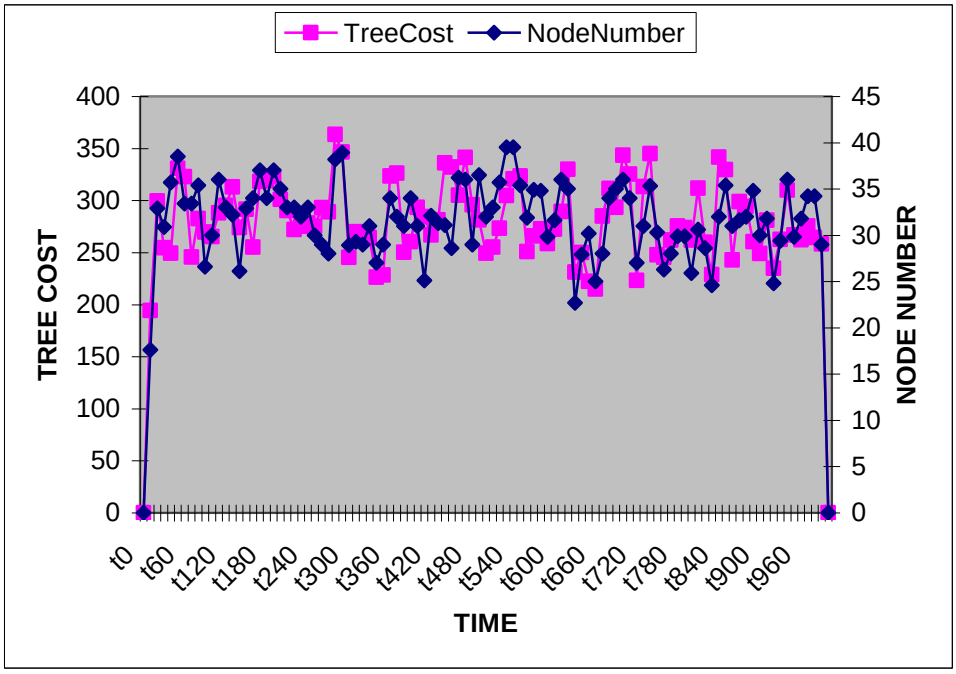
EK-A 47: SCMP, Flat Random Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)



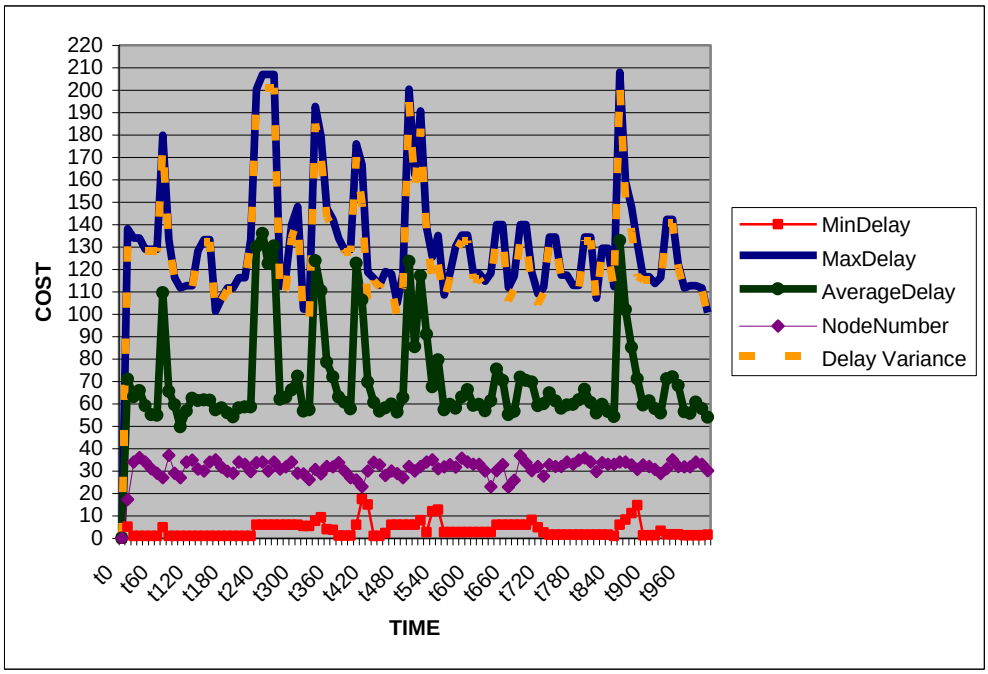
EK-A 48: SCMP, Flat Random Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)



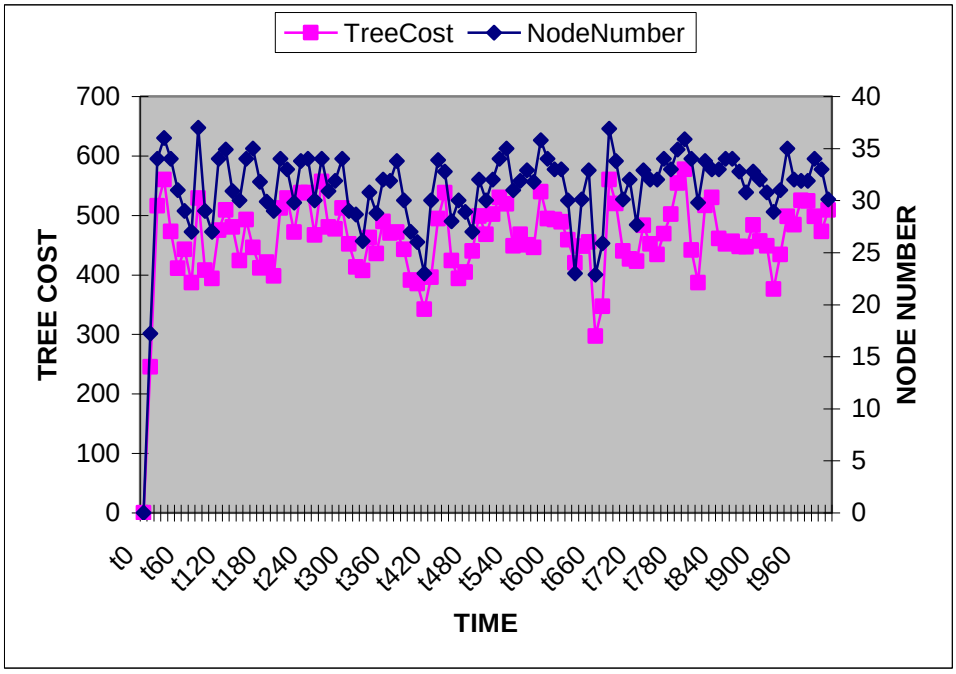
EK-A 49: SCMP, Transit Stub Network, 50 nodes with α=1, Scenario Type:RC(Relay Chat)



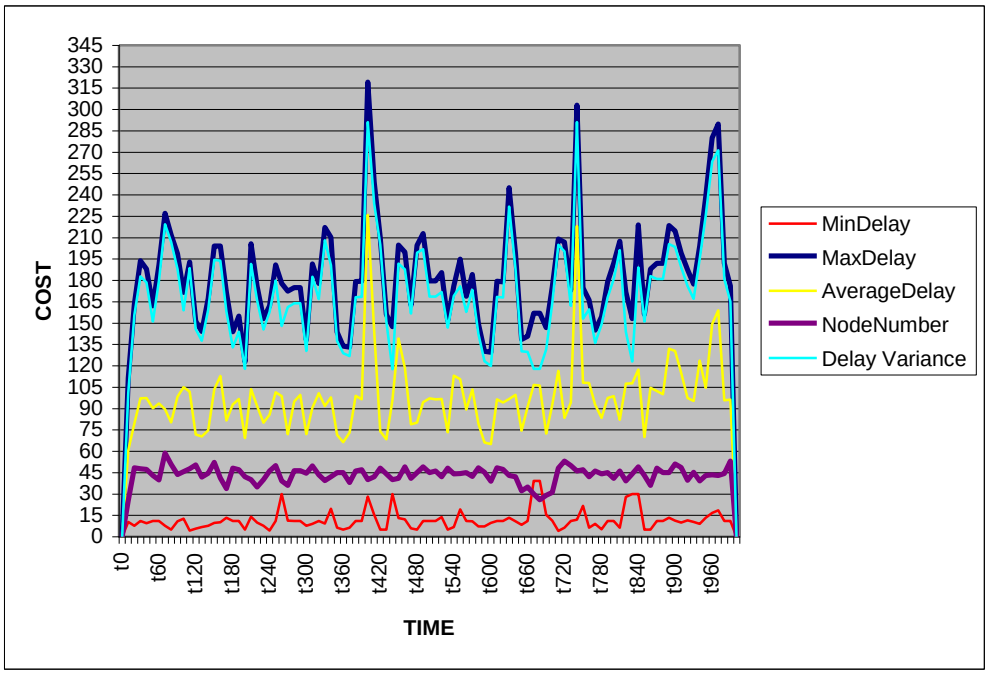
EK-A 50: SCMP, Transit Stub Network, 50 nodes with α=1, Scenario Type:RC(Relay Chat)



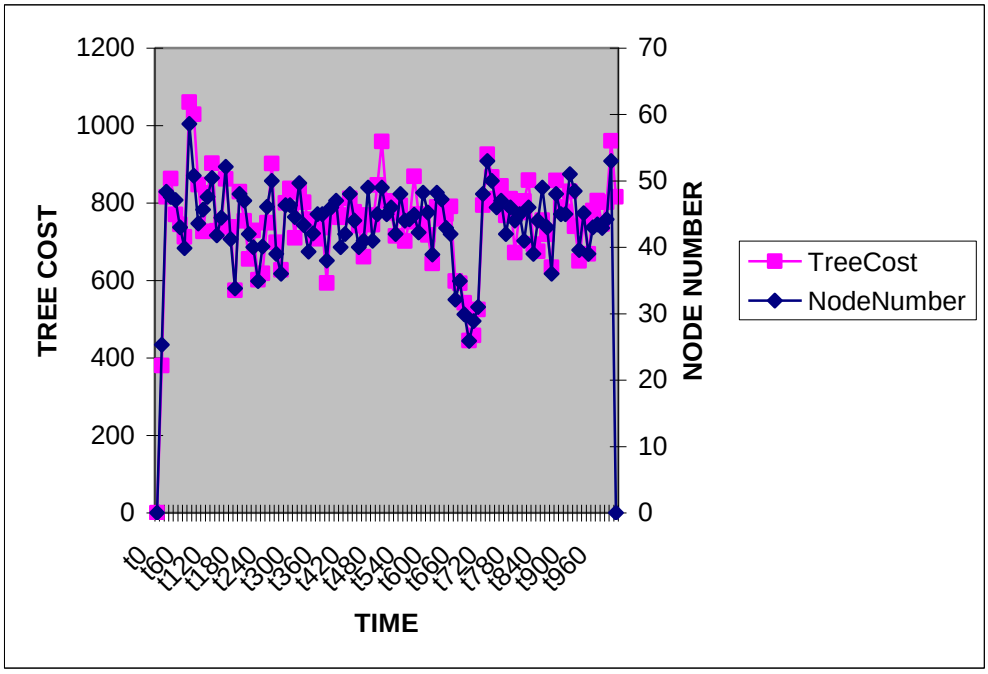
EK-A 51: SCMP, Transit Stub Network, 50 nodes with α=0.5, Scenario Type:RC(Relay Chat)



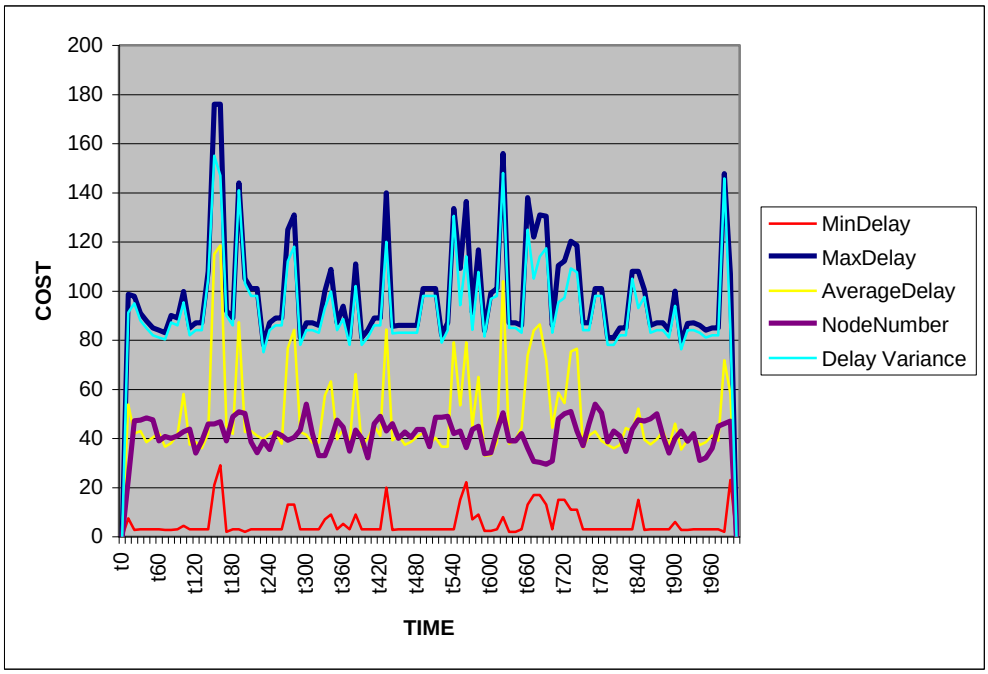
EK-A 52: SCMP, Transit Stub Network, 50 nodes with α=0.5, Scenario Type:RC(Relay Chat)



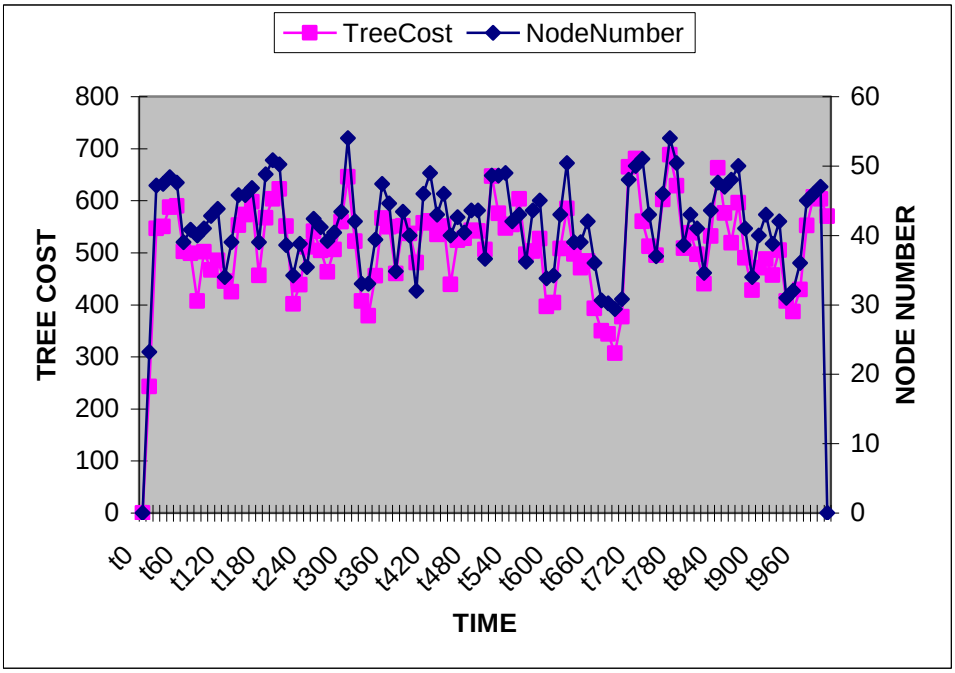
EK-A 53: SCMP, Transit Stub Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat)



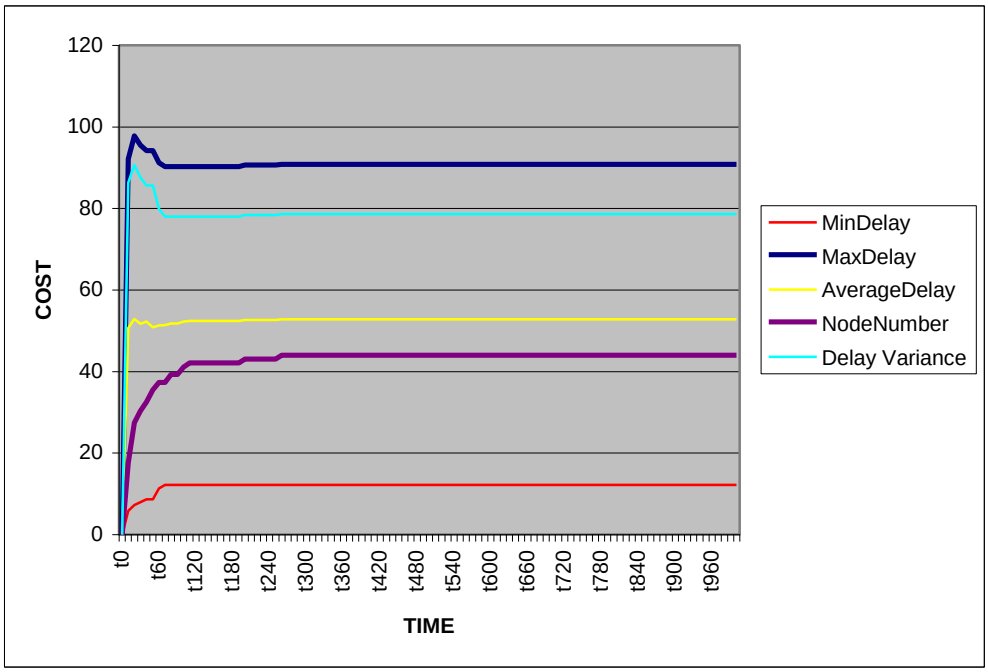
EK-A 54: SCMP, Transit Stub Network, 100 nodes with α=0.5, Scenario Type:RC(Relay Chat)



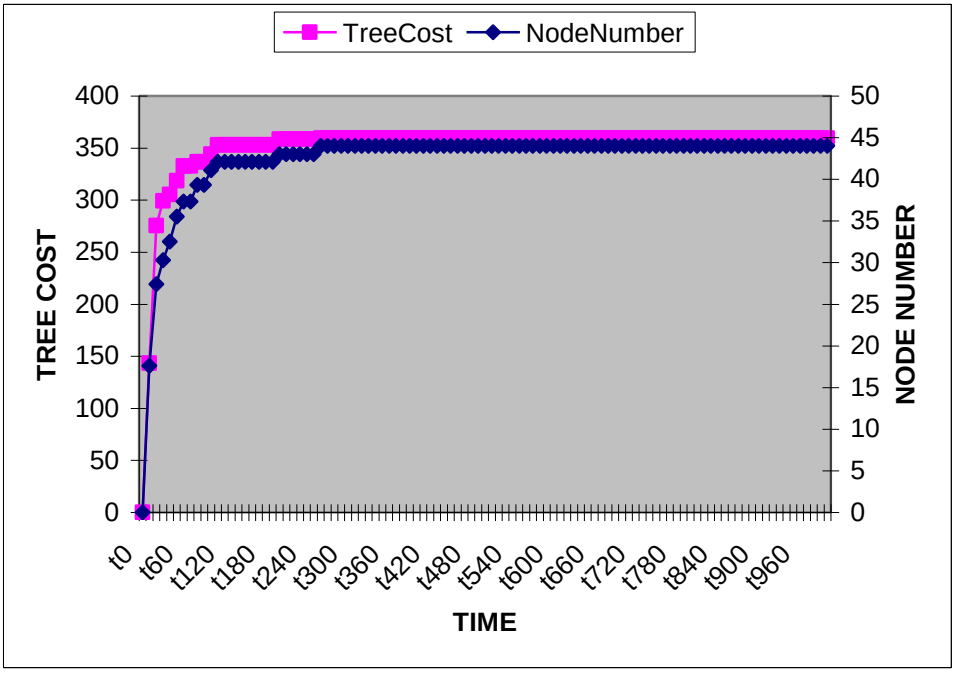
EK-A 55: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat)



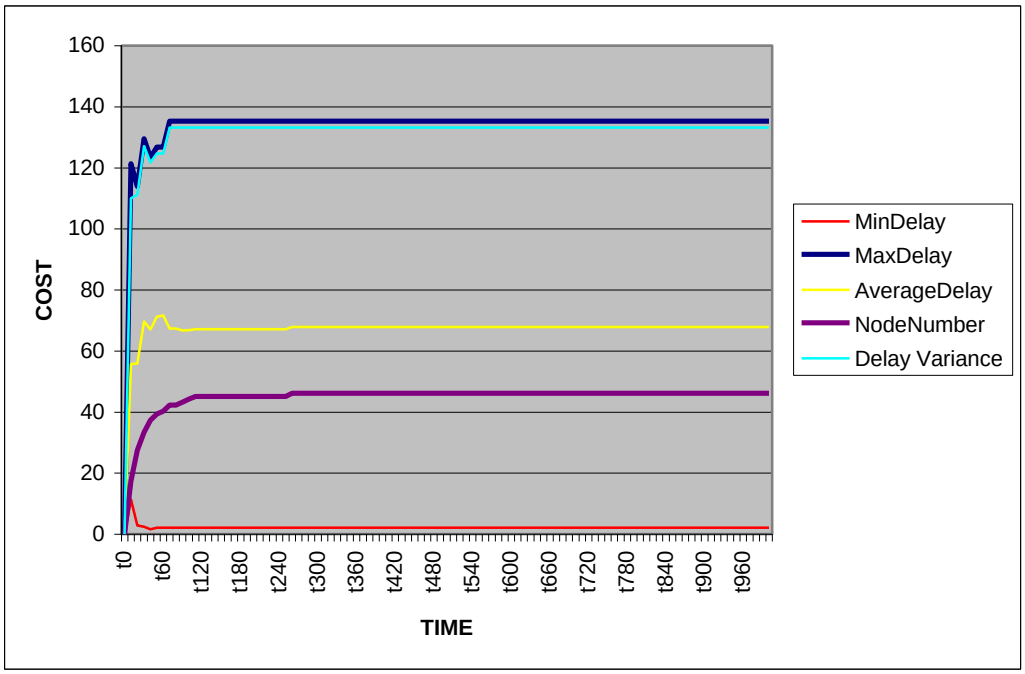
EK-A 56: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:RC(Relay Chat)



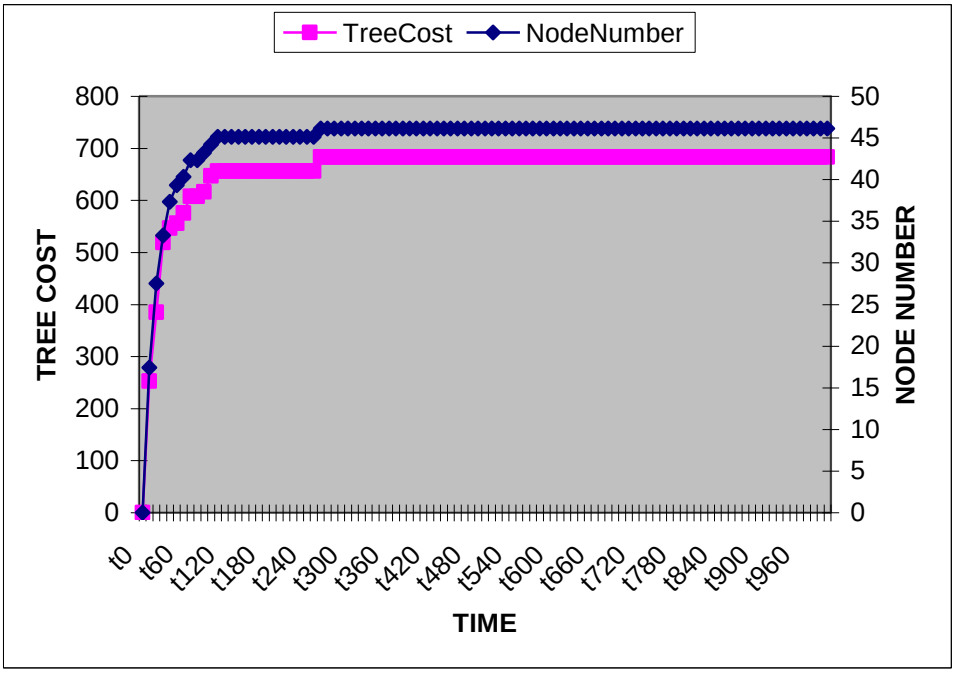
EK-A 57: SCMP, Transit Stub Network, 50 nodes with α=1, Scenario Type:VC(Video Conference)



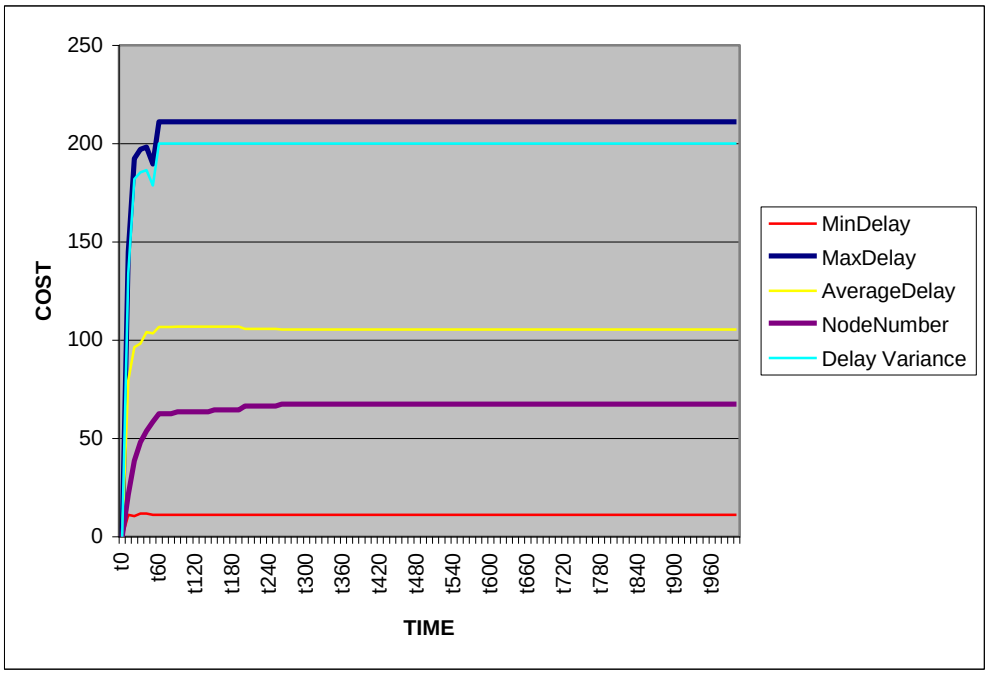
EK-A 58: SCMP, Transit Stub Network, 50 nodes with α=1, Scenario Type:VC(Video Conference)



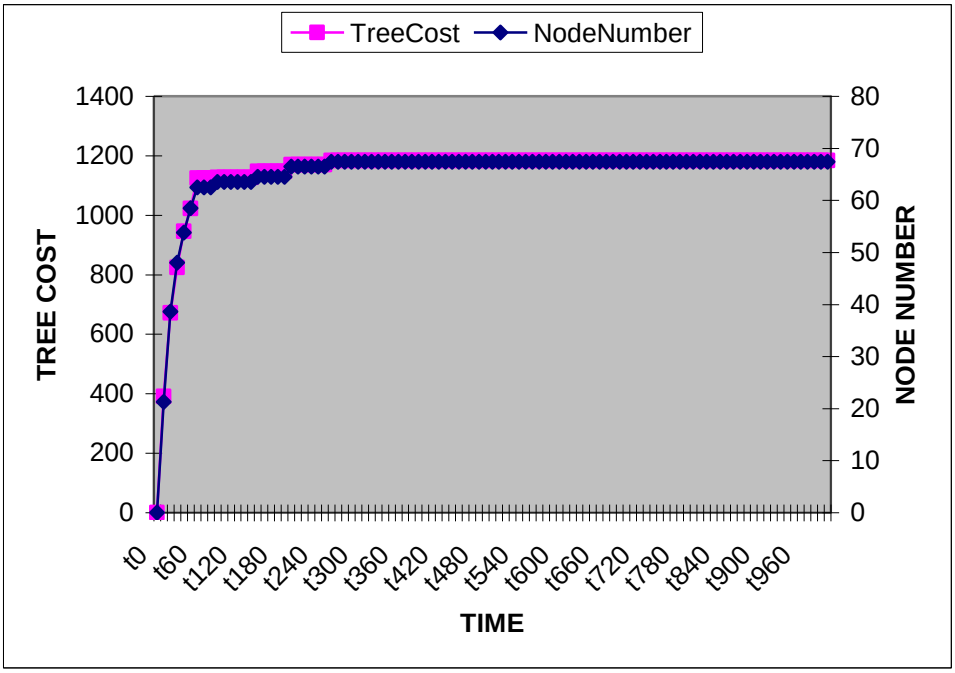
EK-A 59: SCMP, Transit Stub Network, 50 nodes with α=0.5, Scenario Type:VC(Video Conference)



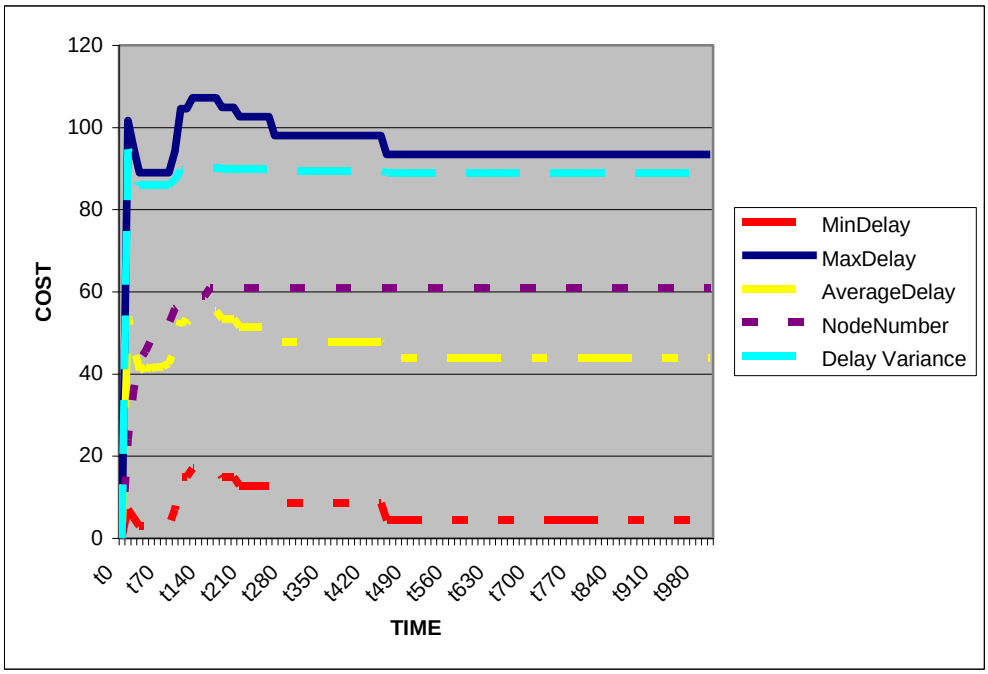
EK-A 60: SCMP, Transit Stub Network, 50 nodes with α=0.5, Scenario Type:VC(Video Conference)



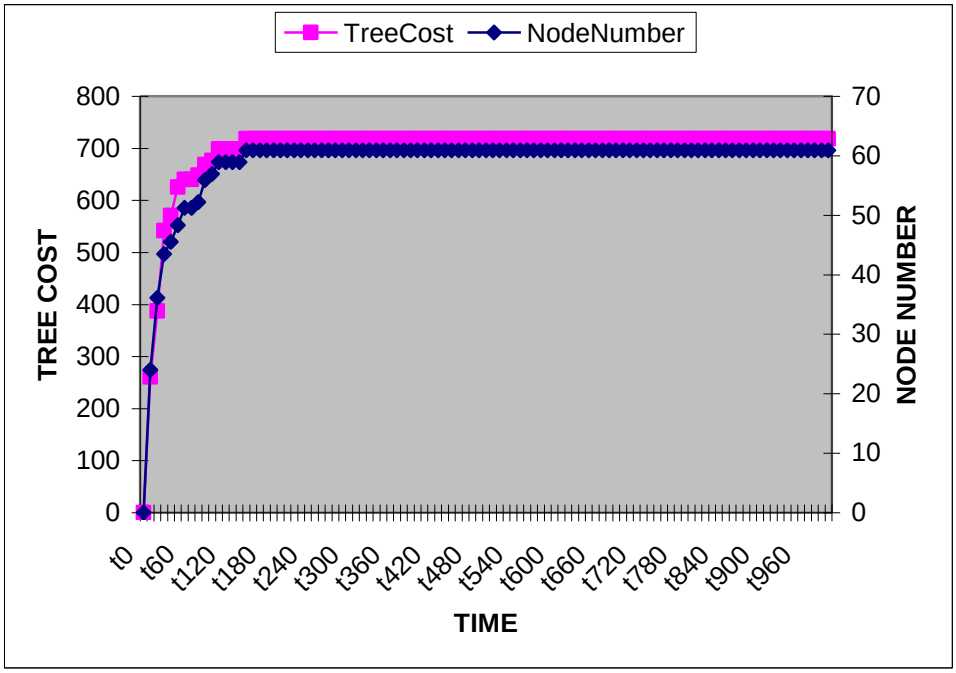
EK-A 61: SCMP, Transit Stub Network, 100 nodes with α=0.5, Scenario Type:VC(Video Conference)



EK-A 62: SCMP, Transit Stub Network, 100 nodes with α=0.5, Scenario Type:VC(Video Conference)



EK-A 63: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)



EK-A 64: SCMP, Transit Stub Network, 100 nodes with α=1, Scenario Type:VC(Video Conference)

RESUME

Osman Belgi Özen was born in Adana in 1979. Having graduated from Private Turkmen College in Mersin, he passed the university entrance examinations and started his education at Istanbul Technical University in the field of Control and Computer Engineering in 1997. He graduated from Istanbul Technical University in 2001. He has been working as senior software engineer since 2001.