Implementation and Evaluation of QoSMIC - a New Internet Multicast Routing Protocol

Shuqian Yan

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright by Shuqian Yan 1999
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-45998-5
Implementation and Evaluation of QoSMIC - a New Internet Multicast Routing Protocol

Shuqian Yan
Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto
1999

Abstract

Multicasting involves the distribution of the same data to several receivers at the same time. Efficient multicast routing can reduce the communication cost of the distribution tree. Multicast service with Quality of Service (QoS) guarantees is critical to support QoS-sensitive applications, such as telemedicine, video-conferencing, and virtual reality. However, most multicast routing protocols, such as PIM and CBT do not employ QoS-sensitive routing. QoSMIC is a promising approach to QoS-sensitive multicast routing. In this document, we implement the QoSMIC routing protocol using the Network Simulator (NS) and conduct detailed simulations under realistic network environments. We investigate the end-to-end performance and the routing efficiency of QoSMIC. Our experimental results show that QoSMIC outperforms Core-based trees, such as those formed by PIM-SM, CBT, or BGMP, by a large margin, under a wide range of network topologies and conditions. We also examine the overhead complexity of QoSMIC. Our results show that we can reasonably limit the control overhead of QoSMIC without sacrificing the optimality of the multicast tree.
Acknowledgments

First of all, I would like to express my deepest gratitude to Professor Anindo Banerjea. for his tremendous guidance, support, encouragement, and dedication throughout the whole project.

I would also like to thank Professor Irene Katzela. for her warmth, patience, and all the great discussions. She is truly a wonderful co-supervisor right here at the University of Toronto.

I want to thank my committee members, Professor A. Leon-Garcia, Professor Jinwoo Choe, and Professor Bruce Francis, for carefully reviewing my thesis and providing all the valuable comments.

I also want to express my gratitude to NSERC, for their vision in science and technology. Without their generous financial support to many young researchers, like myself, we simply will not accomplish this much in our areas.

Last but not least, I want to express my appreciation to my family, especially to my parents, for their continuous love and caring. Without all these, I will not walk this far.
Contents

Abstract ii

Acknowledgements iii

Table of Contents iv

List of Tables vii

List of Figures viii

1 Introduction 1
1.1 Motivation .................................. 1
1.2 Objective .................................. 2
1.3 Previous Work ................................ 3
1.4 Thesis Organization ........................ 4

2 The QoSMIC Internet Protocol and its Implementations 6
2.1 Overview .................................. 6
2.2 Messages and Mechanisms ...................... 9
2.2.1 The Search for Candidates ................ 10
2.2.2 Connection Establishment .................. 12
2.2.3 Leaving a Group .......................... 13
2.2.4 Candidate Selection ....................... 13
2.3 Implementation Overview ...................... 15
2.3.1 The Simulation Environment ............... 15
## 3 Measurement of End-to-end Performance

3.1 Simulation Model .................................................. 18
3.2 Results .................................................................. 21
   3.2.1 Uniform Delay Distribution ............................. 21
   3.2.2 Bi-modal Delay Distribution ......................... 25
   3.2.3 Heavy-tail Delay Distribution ....................... 27
3.3 Summary ................................................................. 30

## 4 Measurement of Overhead Complexity

4.1 Simulation Model .................................................. 32
4.2 Results .................................................................. 35
   4.2.1 Full Blast Multicast Tree Search ....................... 35
   4.2.2 Multicast Tree Search with the Directivity Mechanism ...... 39
   4.2.3 Multicast Tree Search with the Local Minima Mechanism .... 40
   4.2.4 Multicast Tree Search with the Combination of Directivity and
       Local Minima Mechanisms ........................................ 41
   4.2.5 Multicast Tree Search with the Combination of Directivity and
       Local Minima and Fractional Choice mechanisms .......... 42
4.3 Summary ................................................................. 45

## 5 Measurement of Routing Efficiency

5.1 Simulation Model .................................................. 47
5.2 Results .................................................................. 48
   5.2.1 Uniform Load Distribution ............................. 48
   5.2.2 Bi-modal Load Distribution ......................... 49
   5.2.3 Truncated Heavy-tail Load Distribution ............ 50
5.3 Summary ................................................................. 52
6 Conclusions

6.1 Contributions .................................................. 54

6.2 Future Work .................................................... 56

Bibliography ......................................................... 57
List of Tables

2.1 The different roles of the routers. ............................................. 9
2.2 An explanation of the messages in QoSMIC. ................................. 10

3.1 Waxman flat graphs of different average degree of nodes and number of duplex links. .......................................................... 20
3.2 The end-to-end delay observed by receiver in seconds using the uniform delay distributions. ......................................................... 24
3.3 The end-to-end delay observed by a receiver in seconds using the heavy tailed delay distributions: Waxman flat euclidean graph (node degree 3.78). ....................................................... 28

5.1 Percentage of blocked receivers using the uniform load distribution: MBone graph. ................................................................. 50
5.2 Percentage of blocked receivers using the heavy-tail load distribution: MBone graph. ................................................................. 52
5.3 Percentage of blocked receivers using the heavy-tail load distribution: Waxman flat Euclidean graph (node degree 3.78). ................. 52
List of Figures

2.1 An overview of QoSMIC................................................. 6
2.2 The Take-Over procedure.............................................. 12

3.1 Fraction of nodes below an end-to-end performance value using the 
uniform delay distributions: MBone graph.................................. 21
3.2 Fraction of nodes below an end-to-end performance value using the uni-
form delay distributions: Waxman flat euclidean graphs (node degree 
2.5). ....................................................................................... 22
3.3 Fraction of nodes below an end-to-end performance value using the uni-
form delay distributions: Waxman flat euclidean graphs (node degree 
3.78 and 4.32). ....................................................................... 23
3.4 Fraction of nodes below an end-to-end performance value using the uni-
form delay distributions: Waxman flat euclidean graphs (node degree 
4.94 and 5.52). ....................................................................... 23
3.5 Average end-to-end delay vs. the percentage of high delay links using 
the bi-modal delay distributions: Waxman flat euclidean graph (node 
degree 3.78). ........................................................................... 26
3.6 Average end-to-end delay as a function of the average node degree 
using bi-modal(20) distribution: Waxman flat euclidean graph.......... 27
3.7 Fraction of nodes below an end-to-end delay value using the heavy 
tailed delay distributions ($\beta = 0$): Waxman flat euclidean graphs (node 
degree 3.78). ........................................................................... 29
3.8 Fraction of nodes below an end-to-end delay value using the heavy
tailed delay distributions ($\alpha = 0.5$): Waxman flat euclidean graphs
(node degree 3.78). ........................................ 29

4.1 Measurement results using full blast msearch as a function TTL value:
Waxman flat Euclidean graph (node degree 3.4). .................. 36

4.2 Overhead complexity using full blast msearch as a function TTL value:
Waxman flat euclidean graph (node degree 3.4). .................. 37

4.3 An approximate model to compare the set-up latency between QoSIC
and PIM-SM. .................................................. 39

4.4 Comparison on the Number of Candidates between different Candidate
selection mechanism. This is also the number of BID messages received
by the new router. (For this series of figures, we use the following
representation on the x axis: “d(5)” refers to the directivity mechanism
with a threshold value of 5, similarly for d(0); “l” refers to the Local
Minima mechanism; “d+l” refers to the combination of directivity and
the local minima mechanisms; and “d+l+f” refers to the combination
of directivity, local minima and the fractional choice mechanisms.) . . 42

4.5 Comparison on the Number of hops per successful join between different
Candidate selection mechanism. .................................. 43

4.6 Comparison on the Number of BID-ORDER messages between different
Candidate selection mechanism. .................................. 43

4.7 Comparison on the Number of BID messages between different Candid-
ate selection mechanism. ........................................ 44

5.1 Percentage of blocked receivers as a function of average degree of nodes
using the uniform load distribution: Waxman flat euclidean graph. . . 49

5.2 Percentage of blocked receivers vs. percentage of highly loaded links
using the bi-modal load distribution. ............................... 51

5.3 Percentage of blocked receivers as a function of average degree of nodes
with 20% highly loaded links: Waxman flat euclidean graph. ........ 51
Chapter 1

Introduction

1.1 Motivation

Multicasting can be defined as the distribution of the same information stream from one to many nodes concurrently. A multicast connection can substitute for many unicast connections carrying the same information, thereby reducing the load on the network. In the last few years, multicast routing has attracted a lot of attention from both the academic and industrial communities, since many emerging applications are multicast in nature, such as tele-conferencing, tele-education, and computer supported collaborative work.

The motivation for this work can be best illustrated through a real-life analogy: driving a car. When we drive to work, we want multiple highways so that we can pick the least congested one. In addition, we need a radio station to give us up-to-date information before we get stuck in traffic. In the context of communication, traffic-aware routing has been used extensively in the telephone network. However, it is not used in the current Internet.

The Internet is a packet switching network that principally provides best-effort service. There are no guarantees for the services and applications that run over it; applications may "starve", end-to-end delays may be arbitrary, and packets may be lost. Traditional Internet routing protocols do not consider Quality of Service (QoS)\(^1\)

---

\(^1\)Quality of Service (QoS) denotes the user-perceived quality, described in terms of requirements
requirements in their routing and provide a single path for each participant, based on hop-count as the only routing metric. Some of these protocols are heavily dependent on the initial core selection and suffer from poor end-to-end performance.

The Internet Engineering Task Force (IETF) has created a working group in multicast routing under the name of Inter-Domain Multicast Routing (IDMR), whose responsibility is to provide protocols and propose standards.

QoSMIC is a promising approach to QoS-sensitive multicast routing that has been presented for consideration to the IDMR working group as an Internet-Draft[2]. QoSMIC identifies multiple paths for each new member of the group, which chooses the path that suits its QoS requirements best. The routing decisions consider the quality requirements of the users. In addition, the protocol limits the role of pre-configuration decisions, which causes problems in the deployment of previous protocols.

QoSMIC is intended for long-lived multimedia sessions with QoS requirements, such as video-on-demand, video-conferencing, or remote education. The work done in finding the route can be amortized over the duration of the session. As such, some of the trade-offs made in the QoSMIC protocol favor quality over speed. QoSMIC is intended to exist as an option to a fast, cheap routing protocol, such as PIM-SM, which would be used to route short-lived multicast sessions, such as news distribution or stock updates.

The QoSMIC protocol establishes a basis for more research and implementation work. Hence, we decide to perform in-depth study of QoSMIC in the forms of simulation and measurement.

1.2 Objective

The objective of this work is two fold: modeling and evaluation. First, we want to model the QoSMIC protocol in more detail in the Internet environment. Second, we want to investigate its network performance through detailed simulations, and based such as end-to-end delay, jitter, packet loss, and traffic characteristics such as average bit rate, peak bit-rate, and burst size.
on the insights gained, to further fine tune the protocol according to the properties and needs of Internet sessions. The specific types of measurements we would like to perform are:

- **End-to-end Performance.** Evaluate the end-to-end performance of QoS-MIC to quantify the benefits of doing QoS-sensitive routing for multicast applications.
- **Overhead Complexity.** Investigate the flexibility and scalability of QoS-MIC in various networks by looking into its overhead complexity.
- **Routing Efficiency.** Determine the effectiveness of QoS-MIC in utilizing network resources. The effectiveness can be measured in terms of the percentage of receivers prevented from joining the multicast session due to insufficient network resources.

### 1.3 Previous Work

In this section, we present a brief overview of the current Internet multicast protocols. We can distinguish two categories among the Internet multicast protocols based on QoS considerations: QoS-oblivious protocols [1] [5] [13] [10]; and QoS-sensitive protocols [3] [4] [16].

**QoS-oblivious protocols.** These protocols provide one route when a new member joins, and QoS is not considered in the selection of the route. Examples of such protocols are Core Based Trees (CBT) [1], Protocol Independent Multicasting - Sparse Mode (PIM-SM) [5], Border Gateway Multicast Protocol (BGMP) [13], and Multicast Internet Protocol (MIP) [10]. All these protocols assume rooted trees with a **core router** as the center of the distribution tree. In BGMP, for example, each domain "owns" an address space and is the root of the distribution tree with such an address. The core is chosen in the beginning, and it does not change until the end of the session. In all these protocols, the routing is based on the Reverse Shortest Paths algorithm: the tree is the union of the reverse shortest paths between the core and each destination. Reverse Shortest Paths are used because they can be computed
from the unicast routing database without requiring additional information. CBT creates only Shared Trees, while PIM-SM and BGMP start with Shared Trees and permit the creation of Source-Based Trees for very active sources (high data rate). The MIP protocol introduces new approaches to administrative and correctness issues and guarantees loop-free Shared Trees and Source-Based Trees. The assumption of rooted trees simplifies routing, but introduces the core selection [14] and address partitioning [6] problems, which can affect the protocol performance significantly.

*QoS-sensitive protocols.* These protocols offer multiple routes to a new member, who selects the best such path. Carlberg and Crowcroft [3] [4] suggested the Yet Another Multicast (YAM) protocol, which creates a Shared Tree considering multiple routes. The destination router searches its neighbourhood and finds routers that are part of the tree of the desired group. This way, the new router selects the most promising route. Routing in YAM relies mainly on this neighbourhood search, but this procedure can increase the control overhead significantly. Zappala et al. [16] suggested a multicast protocol that provides alternate routes to avoid a bottleneck link. This protocol does not seem applicable in the case where QoS suffers from several mildly congested links. Finally, both protocols use only the static Quality of Route (QoR) metrics, defined by Zappala et al. to refer to multiple static parameters of the route, such as link capacity, delay, or reliability.

### 1.4 Thesis Organization

This document is arranged in the following manner.

First, in Chapter 2, we will give an overview on the design of QoSMIC. We will also describe the implementation details about QoSMIC and introduce the simulation platform, the Network Simulator (NS), in which we have implemented the Shared Tree version of QoSMIC.

After that, we present our measurement work in three different chapters: end-to-end performance, overhead complexity, and routing efficiency. For all our experiments, we evaluate and compare the performance of QoSMIC with the existing
multicast routing protocols, such as PIM-SM, YAM, and DVMRP. We use the simple Shared-tree implementation of PIM-SM, available in the NS distribution, which is representative of any Core-based Shared Tree protocol based on Reverse Shortest Paths (RSP) routing.

**End-to-end Performance** (Chapter 3). We measure both the end-to-end delay and end-to-end path-length of QoSMIC and compare its performance against those of PIM-SM and DVMRP. We show that the QoS-sensitive routing can improve the end-to-end performance significantly, in some cases almost tripling the number of satisfied users for a given delay requirement as compared to PIM-SM.

**Overhead Complexity** (Chapter 4). We measure the control overhead for both QoSMIC and YAM. We focus on the tree setup portion of the protocol. We show that the QoSMIC protocol causes a reasonable and controllable increase in the number of control messages without degrading the performance in QoS-sensitive routing. We further fine tune the parameters used in the protocol according to the properties and needs of different Internet sessions.

**Routing Efficiency** (Chapter 5). We investigate the effectiveness of QoSMIC in utilizing network resources. The effectiveness is measured in terms of the percentage of receivers prevented from joining the multicast session due to insufficient network resources. We investigate different load distribution functions on the effect of the protocol. We show that for a given load distribution, QoSMIC always yields more call admissions than PIM-SM, while at the same time finding the path that best meets the application’s QoS requirements.

Finally, in Chapter 6, we summarize our work and identify avenues for further research.
Chapter 2

The QoSMIC Internet Protocol and its Implementations

In this chapter, we present QoSMIC, a protocol for the Internet that supports QoS-sensitive multicast applications. In section 2.1, we provide an overview on the design of QoSMIC. In section 2.2, we describe in detail the messages and mechanisms used by QoSMIC. In section 2.3, we offer an overview on the implementations of QoSMIC and we briefly describe the simulation platform: the Network Simulator (NS).

2.1 Overview

Figure 2.1: An overview of QoSMIC.
2.1 Overview

QoSMIC creates Shared Trees by default and Source-Based Trees when needed. In both cases, the protocol offers alternate paths for each connection (see Figure 2.1(a)).

In QoSMIC, the notion of a Manager router of a group is introduced. The Manager administers a specific multicast group and facilitates the joining of the new group members. The fundamental difference between a core router and a Manager is that the distribution tree is not necessarily rooted at the Manager. This way, the selection of the Manager has marginal effect on the topology of the tree. Furthermore, we can have multiple Managers and change the Manager during the lifetime of the group without any data loss, as we will see later in this section.

Joining a group. The Designated router of the new member, which we call New router, will try to connect to the most promising router of the tree, according to the QoS metrics defined by the application. The New router identifies several in-tree routers as Candidates or potential points to join to the tree. This search is conducted by two procedures: one from the side of the New router (Local Search), and one from the side of the tree (Multicast Tree Search) (see Figure 2.1(b)), both operating in parallel. For the moment, we consider a purely intra-domain or purely inter-domain scenario.

1. The Local Search Procedure. The New router searches its neighbourhood, using probe messages, with scope limited by the use of the time-to-live (TTL) field for its closest in-tree router. Every in-tree router that receives the probe message and responds with an “advertisement” message unicast to the New router. This message lets the New router know that the in-tree router is a Candidate, i.e., a possible point to join the tree. This procedure is also used in the YAM protocol.

2. The Multicast Tree Search Procedure. The New router contacts the Manager router of the group, and the Manager router “informs” the tree of the New router. Some in-tree routers are selected as Candidates. They advertise themselves to the New router with a unicast “advertisement” message. In the next section, we will describe several mechanisms, centralized and distributed, to select Candidates.
Eventually, the New router compares all the identified connection paths, and selects the best among them according to the needs of the application. The New router sends one more message (JOIN) towards the selected Candidate, to set up the routing state along the path and the chosen router starts forwarding the data. The routing state is fixed or pinned, and the route does not change once it has been established. It should be clear from the above description that QoSMIC protocol corresponds to an implementation of the Greedy algorithm [12]. The New router searches for the "closest" point of the tree and connects there.

The path chosen by the advertisement messages depends on the static Quality of Route (QoR) metrics of the point-to-point routing table. The specific metric used by the routing protocol depends on the needs of the application. For example, real-time applications will prefer paths of minimum end-to-end delay, while data transfers may prefer paths of maximum bandwidth or low packet-loss ratio. Currently point-to-point routing protocols already consider multiple static metrics.

However, the advertisement messages can collect up-to-date dynamic QoS metrics along the path on which they travel (e.g., by interacting with RSVP [18], or using measurement based metrics). Thus, although the paths that the Candidates choose are restricted by the static information in the routing information base, the New router selects among these paths using dynamic routing information.

Source-Based Trees. Using Shared Trees minimizes the routing table information to an entry per group. However, Shared Trees may fail in the following two-cases. First, when a group has many highly active sources simultaneously, the bandwidth of the shared links may not be able to accommodate all the traffic. Second, when the QoS requirements of a user are not met along the Shared Tree, we have to find a different source-to-destination path. In both cases, we resort to Source-Based Trees. The switch from a Shared Tree to a Source-Based Tree of a specific source is initiated by the receiver. The procedure for establishing Source-Based Tree is similar to the procedure of the Shared Tree and uses the Local Search and the Multicast Tree Search procedure to identify routers in the Source-Based Tree. To avoid packet duplication, the Shared Tree is pruned on a source-specific basis, for the sources for which Source-
Based Trees have been established.

2.2 Messages and Mechanisms

In this section, we describe the messages and mechanisms used by QoSMIC in detail. Next, we explain the process of Candidate selection. Table 2.1 lists the different roles of the routers. Table 2.2 contains a list of the protocol messages.

<table>
<thead>
<tr>
<th>Router Name</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manager router</td>
<td>Supervises the group and facilitates the joining procedure: does not necessarily participate in the data distribution tree.</td>
</tr>
<tr>
<td>Candidate router</td>
<td>Considered as possible joining point for a new connection.</td>
</tr>
<tr>
<td>Designated router</td>
<td>Connected to a set of users (e.g. LAN). receives request from users (e.g. IGMP) and initiates searches.</td>
</tr>
<tr>
<td>New router</td>
<td>Designated router of a LAN containing the new member.</td>
</tr>
<tr>
<td>Destination router</td>
<td>Designated router of a LAN that has active group members.</td>
</tr>
<tr>
<td>Intree router</td>
<td>All routers that are part of the distribution tree.</td>
</tr>
<tr>
<td>Intermediate router</td>
<td>Intree routers that is not a Destination or a Source.</td>
</tr>
<tr>
<td>Border router</td>
<td>Router that connects multiple domains.</td>
</tr>
</tbody>
</table>

Table 2.1: The different roles of the routers.

Every router maintains multicast routing information, which enables it to forward each multicast data packet to an appropriate number of links. In more detail, each router maintains a routing table for each link. A link has an entry in its table for each group for which the link is part of the distribution tree. We denote such an entry for the Shared Tree of group G by (*,G); the symbol "*" represents “all sources”. A data packet of a group is forwarded to all links that have (*,G) entry, except from the link that it arrived from. A router discards a data packet that arrives on a link that does not have a (*,G) entry, which means it is not part of the tree. Furthermore, a table entry can refer to a specific source of a multicast group (source specific information).
2.2 Messages and Mechanisms

<table>
<thead>
<tr>
<th>MESSAGE</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BID-REQ</td>
<td>New router searches locally for intree neighbours.</td>
</tr>
<tr>
<td>BID</td>
<td>Candidate “proposes” to the New router; Message collects dynamic QoS metrics along the path.</td>
</tr>
<tr>
<td>M-JOIN</td>
<td>New router contacts the Manager who initiates a Multicast Tree Search.</td>
</tr>
<tr>
<td>BID-ORDER</td>
<td>The Manager “orders” the intree nodes to send bids.</td>
</tr>
<tr>
<td>JOIN</td>
<td>The New router sends a message up its selected path to the tree to establish/refresh routing state.</td>
</tr>
<tr>
<td>PRUNE</td>
<td>Departing Destination tears down unneeded part of tree; can be for a source or for a group.</td>
</tr>
</tbody>
</table>

Table 2.2: An explanation of the messages in QoSMIC.

and it is denoted as (S,G) for source S and group G. Finally, the state of a link, (*,S) or (S,G), may be tentative: the link is considered in the distribution tree, but no data is forwarded on that link. The state of a link is modified by protocol control messages, as we will see later in this section. In addition, state can be expired: adjacent nodes exchange messages periodically, and refresh the state of their common links. If such a state refresh does not happen, the state of the link expires.

2.2.1 The Search for Candidates

We assume that the New router receives a request to join a multicast group. To simplify the presentation, we assume that the new member is a receiver\(^1\). The Join is received via IGMP on a LAN at the intra-domain level, or from an intra-domain protocol at a Border router at the inter-domain level. If the New router is part of the group already, the connection is established locally. If the New router is not part of the

\(^1\)A source joins a group in a way similar to a receiver; it connects to the closest router of the Shared Tree. A difference is that the dynamic metrics of the path are collected “towards the tree”, i.e., in the direction the data will flow.
the group, the Local Search and the Multicast Tree Search procedures are employed.

**Local Search.** The New router tries to identify neighboring intree routers.

1. The New router “floods” a BID-REQ message in its neighbourhood. Reverse path multicasting with scope controlled by the Time To Live (TTL) field is used to control the message complexity of this phase. This procedure is the same as proposed in YAM, but because QoSMIC also has the Multicast Tree Search, the TTL value can be kept very small. The advantage is quantified later in Chapter 4 using simulation results.

2. Every intree router that receives a BID-REQ message, becomes a Candidate router, and replies with a BID message, which is unicast to the New router. The BID message on its way collects information on the expected performance of the path, based on dynamic QoS metrics. The Candidate router considers the New router as a tentative dependent, and cannot leave the tree unless the tentative status is timed out.

3. The New router collects BID messages. The procedure terminates unsuccessfully, if the New router does not receive any replies before the expiration of a timer set for this purpose. Otherwise, we enter the phase of establishing the connection (see section 2.2.2).

**Multicast Tree Search.** The New router contacts the Manager and the Manager causes some of the intree nodes to propose themselves as Candidate routers. The selection of Candidates is an important aspect of QoSMIC and it is discussed in section 2.2.4. The sequence of actions is as follows:

1. New router sends an M-JOIN message to the Manager of the group.

2. The Manager “orders” a bidding session with a BID-ORDER message. Some subset of the routers that receive the BID-ORDER are selected as Candidates.

3. The Candidates unicast BIDs to the New router. The BIDs are identical to the BIDs in the Local Search.
2.2 Messages and Mechanisms

For both the Local Search and the Multicast Tree Search, if an intree router receives a BID message for the same tree, the router takes the place of the Candidate by "dumping" the original BID and initiating a new BID message. This procedure is called Take-Over (see Figure 2.2), and it is essential in preventing loops.

2.2.2 Connection Establishment

Having performed the BIDing phase, we will examine how the connection is established.

1. The New router selects the best Candidate according to the dynamic QoS metrics collected by the BIDs.

2. The New router sends a JOIN message to the Candidate who sent the best BID. The JOIN message traverses the path used by the BID message in the opposite direction, and establish routing state along the path.

3. When the chosen Candidate receives the JOIN message, it starts transmitting data packets on the newly set-up path towards the New router.

4. If an intree router receives a JOIN message, it performs a Take-Over, terminates the message and starts forwarding data on the newly set-up path. This way again it avoids the creation of cycles.

Figure 2.2: The Take-Over procedure.
2.2 Messages and Mechanisms

2.2.3 Leaving a Group

A Designated router receives a leave request through the same protocol that communicated the join request. The request can be for a whole group or for a source of a group. Whenever a router senses\(^2\) a change in the membership, it removes the link from the distribution tree, and checks whether it has become a leaf of the related tree. If so, it sends a PRUNE message up the tree and removes the state for the tree from its database, thereby ceasing to be an intree router for that tree.

2.2.4 Candidate Selection

During the Multicast Tree Search, the Manager must select an appropriate subset of the intree routers as Candidates. This can proceed in a centralized or a distributed way.

1. **Centralized Selection.** If the Manager has sufficient knowledge of the tree and the network topology, the Manager can select the set of Candidates directly. This can happen if the domain is running a link state protocol, or if the Manager has access to a centralized routing information database. In this case, the Manager identifies promising Candidates, either on demand or statically. Then the Manager unicasts the BID-ORDER to them. If the selection is based on accurate information, we can find the right Candidates with limited overhead cost. Note, that with this mechanism. QoSMIC can behave like CBT or BGMP: a Manager can trivially act as a core router by considering itself as the only Candidate.

2. **Distributed Selection.** In the absence of centralized information, each intree router has to decide whether to become a Candidate in a distributed way. We assume that each router has a static estimate of its distance to the New router. We have the Manager join the tree as a source and multicast the BID-ORDER

\(^2\)The membership change can be either the result of an IGMP message, a PRUNE message from a neighbouring router, or an expiration of the soft state if an update is not received for some time.
along the tree. All intree routers receive the BID-ORDER and decide in a distributed way if they should become the Candidates. Naturally, the more intree routers become Candidates, the more “accurately” we find the closest Candidate to the New router, but the more the control overhead. The following orthogonal mechanisms has been proposed that can be used in combination to strike the balance in this trade-off.

(a) **Directivity.** The distant routers is discouraged from becoming Candidates. The BID-ORDER keeps track of the (so far) minimum distance of the tree and the New router. A router with a distance greater than this minimum does not become a Candidate, and if the relative distance exceeds a threshold, the BID-ORDER is not forwarded further.

(b) **Local Minima.** In absence of global knowledge, the locally optimal routers are selected as Candidates. As the BID-ORDER message travels along a branch, the distance to the New router of the previous two routers is included in the message. If router $i + 1$ sees that router $i$ was closer to the destination than both routers $i + 1$ and $i - 1$, it sends a message back to router $i$, which becomes a Candidate.

(c) **Fractional Choice.** In this mechanism, we can choose as Candidates a representative fraction $(1/n)$ of either all intree routers or the ones that meet the other two criteria.

**Choice of Mechanisms.** The choice of different mechanisms will have to consider the network topology and the traffic behavior. If topology information for the entire domain is available, Manager Selection offers the lowest control overhead for a reasonable selection of Candidates. In the absence of such information, Fractional Choice and Directivity together is simple to implement, and may lead to satisfactory solution with a careful choice of parameters. Local Minima, Fractional Choice and Directivity is more sophisticated and promises improved results.
2.3 Implementation Overview

For our work, we use the Network Simulator (NS) version 2.1b5 by UCB/VINT/LBNL as the basic simulation platform. The implementation of QoSMIC lies in the redefining of protocol specific actions and handling of various control messages by individual routers, which is described next.

2.3.1 The Simulation Environment

In NS, the main classes in the implementation of multicast routing are the class McastProtoArbiter and the class McastProtocol. In addition, there are some methods and configuration parameters that have been defined in the Simulator, Node, Classifier, and Replicator objects for multicast routing.

McastProtoArbiter class. There is one McastProtoArbiter object per multicast capable node. Each McastProtoArbiter object maintains a list of multicast protocols. This arbiter supports the ability for a node to run multiple multicast routing protocols. Whenever there is a multicast related action in a node, the node will access its multicast protocols through this McastProtoArbiter. Basic functions are join-group{}, leave-group{}, and upcall{}. These functions translate into calls to corresponding functions of the same name in each of the multicast protocols running at that node.

McastProtocol class. This is the base class for the various multicast routing strategy and protocol objects. It contains basic multicast functions: join-group{}, leave-group{}, handle-cache-miss{}, handle-wrong-iif{}, drop{}. Protocol specific actions are appropriately redefined by the individual multicast protocols that inherit these functions.

The node entry for a multicast node is the switch. It looks at the highest bit to decide if the destination is a multicast or unicast packet. Multicast packet are forwarded to a multicast classifier which maintains a list of replicators; there is one replicator per (source,group,incoming interface) tuple. Replicators copy the incoming packet and forward them to all outgoing interfaces.
For our work, we augment NS with an implementation of the Shared Tree portion of QoSMIC. We do not explore the Source Based Tree aspect of QoSMIC, mainly due to the scaling problems of source specific trees.

2.3.2 Router Actions

Here we study the actions of a router as triggered by control messages.

**BID-REQ(*/S,G,New router,TTL)** message.

On reception of a (*/S,G) BID-REQ message\(^3\), a router that is part of the tree in question replies with a BID message. In a Source-Based Tree, the Candidate incorporates in the message estimates of the QoS it experiences (quality of the path from the source to the Candidate), when these are available. Note, that this way, we can maximize the end-to-end QoS that the New router will receive. In more detail, the BID message is of the form BID(*/S.G.New router.Candidate.Metrics): it includes the address of the Candidate, and estimates of the QoS of the path. The BID message on its way updates its metrics at each router to estimate the quality of the path.

**MJOIN(*/S,G,New router)** message.

This message is unicasted to the Manager of the group. For a (*/G) message, the Manager sends BID-ORDER(*/S.G.bid-mechanism). The “bid-mechanism” represents the selection mechanisms that we saw earlier. The BID-ORDER can be either multicast over the tree, or unicasted to pre-selected routers depending on the mechanism in use.


The action on this message depends on the “bid-mechanism”. In general, the message is either unicasted to the chosen Candidates or multicast over the tree. In the first case, on receiving the message, the router unicasts a BID(*/S,G,Metrics) to the New router. In the second case, the router examines the criteria defined by the bid-mechanism. If they are fulfilled, a BID message is sent to the New router. In addition, the BID-ORDER is forwarded further along the tree.

\(^3\)We use (*/S,G) as an abbreviation of the expression (*/G) or (S,G)
2.4 Summary

BID(*/S,G,New router,Candidate,Metrics) message.

On receiving this message, if the router is the New router specified in the message, it updates a list of the best BIDs. An intree router invokes the TAKE-OVER procedure, and the router replace the Candidate. Any other router that receives a BID message, updates the Metrics field and forwards the packet.

JOIN(*/S,G,New router,Candidate) message.

This message travels towards the selected Candidate. When the Candidate router receives the message, it considers the connection as permanent (as opposed to tentative) and sets up the path. Any intree router that receives the message invokes the TAKE-OVER procedure and starts forwarding data.

2.4 Summary

In this chapter, we have presented QoSMIC, a protocol for establishing multicast distribution trees for applications with QoS requirements. Unlike previous protocols, QoSMIC eliminates core selection and uses a greedy approach to add new participants into the existing multicast tree. QoSMIC uses a Local Search procedure similar to YAM, but with a small TTL scope, to handle new participants who are close to the existing tree. Farther participants are joined to the tree using the Multicast Tree Search procedure that involves different mechanisms to select the appropriate Candidate join points. We have implemented a simulation model of QoSMIC in NS to conduct detailed simulations under realistic network environments.
Chapter 3

Measurement of End-to-end Performance

In this chapter, we simulate the end-to-end performance of QoSMIC to quantify the benefits of doing QoS-sensitive routing. We experiment with various network topologies and link delay distributions to observe the effect on the end-to-end performance between protocols. Our simulation results show that QoS-sensitive routing can improve the end-to-end performance significantly, in some cases, almost tripling the number of satisfied users for a certain delay requirement as compared to PIM-SM. In section 3.1, we provide an overview of our simulation model. In section 3.2, we present our simulation results and discuss the implications from these series of experiment. In section 3.3, we summarize our work in this chapter.

3.1 Simulation Model

In this section, we provide an overview of our simulation model.

For our experiments, we look at the effect of using two different path selection criteria for QoSMIC: hop-count and end-to-end delay. Note, that we focus on the maximum end-to-end delay that a receiver sees among the group sources.

We examine the performance of the following protocols.

QoSMIC(hop). This version of QoSMIC will select among several paths the
minimum hop count from the existing tree to the joining node. This captures the dynamic (non-core) aspect of QoSMIC, but not its ability to handle QoS metrics.

QoSMIC(delay). This version of QoSMIC selects the path with the minimum end-to-end delay path. For each path, we consider the maximum end-to-end delay among sources, and then we select the path with the minimum such delay\(^1\).

PIM-SM. We use the simple Shared Tree implementation of PIM-SM that is currently available in NS. With respect to the performance criteria being considered, this implementation is equivalent to any Core-based Shared Tree protocols based on Reverse Shortest Paths (RSP) routing, such as PIM-SM (Shared Tree only), CBT, and BGMP.

DVMRP. We use the existing NS implementation of DVMRP. The protocol creates source specific minimum hop trees, which are expected to show low end-to-end delay. Although it is not a scalable solution, it provides a good reference point. Note, that both PIM-SM and QoSMIC can switch to Source-based trees, to improve their end-to-end delay, at the expense of increased routing state. In this work, we do not evaluate this feature.

We measure both the end-to-end delay and end-to-end path-length for each source-receiver pair, since both of these metrics are important to provide Quality of Service guarantees.

We use real and artificial graphs in our experiment. First, we use a partial MBone topology with 255 nodes and 266 duplex links available in the NS package. Second, we use the flat-random Waxman graphs with 100 nodes and different number of links, which are often used to model the inter-domain topology [17] [15]. Table 3.1 shows the types of Waxman graphs we used in our experiment. All graphs used are undirected. We do not consider asymmetric links in this work.

The delay of each link is randomly chosen. We experiment with various delay dis-

\(^1\) Typically in QoS routing, delay is used as a constraint, while other criteria, such as hop-count or remaining bandwidth, is used as the optimization criteria. However, by using delay as an optimization criteria, we can produce a single graph like Figure 3.1, where we can see the number of receivers that would have succeeded with different delay bounds, without having to repeat the experiment with multiple delay bounds.
3.1 Simulation Model

<table>
<thead>
<tr>
<th>Topology</th>
<th>Number of duplex links</th>
<th>Average degree of node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat 1</td>
<td>125</td>
<td>2.5</td>
</tr>
<tr>
<td>Flat 2</td>
<td>189</td>
<td>3.78</td>
</tr>
<tr>
<td>Flat 3</td>
<td>216</td>
<td>4.32</td>
</tr>
<tr>
<td>Flat 4</td>
<td>247</td>
<td>4.94</td>
</tr>
<tr>
<td>Flat 5</td>
<td>276</td>
<td>5.52</td>
</tr>
<tr>
<td>Flat 6</td>
<td>297</td>
<td>5.94</td>
</tr>
</tbody>
</table>

Table 3.1: Waxman flat graphs of different average degree of nodes and number of duplex links.

Distributions: uniform distribution, bi-modal distribution\(^2\), and heavy-tail distribution [8]. The link delay distribution is intended to capture the effect of several sources of delay as congestion, switching delays, etc. However, instead of computing this delay based on a model of the reservation state of the network, we simply assign a delay to each link from a distribution, and experiment with different distributions to observe the sensitivity of our results to such changes. This methodology neglects the effect of the current session on future sessions, but we address that by only simulating a single session per experiment instance. Thus, the assigned delays to the links represent the effect of the "current state" of the network due to all existing traffic, on the single session that we simulate.

Note that we are mainly interested in comparing the end-to-end delay between protocols and not in its absolute value, since the absolute value directly depends on the mean of the delay distribution used on the links.

\(^2\)We assign high delay values uniformly distributed between [90 ms, 100 ms] to certain percentage of links in the topology, while all the rest of the links are assigned with a delay value uniformly distributed between [1 ms, 10 ms].
3.2 Results

In this section, we present our simulation results, corresponding to the three link delay distributions just mentioned. For each topology, we repeat the experiment for a total of 100 iterations. For each run of the experiment: one multicast session comprised of 10 randomly selected receivers is created with each receiver joining to the group in a random order. The group also has two randomly selected sources. In the absence of well understood models of multicast participant behaviour, this model is as good as any, and does not favor any protocol. Note that the join-only membership behaviour has been shown to perform similar to the join-leave behavior [7], hence we do not investigate the join-leave behavior here. For PIM, the core router is also randomly selected, which corresponds closely to its method of core selection based on hashing the multicast address [5].

3.2.1 Uniform Delay Distribution

We perform six experiments with the uniform delay distribution, these corresponds to one MBone graph and five Waxman flat graphs of different node degrees. The link delays are uniformly distributed in the range of [10ms,100ms].

![Figure 3.1: Fraction of nodes below an end-to-end performance value using the uniform delay distributions: MBone graph.](image)
Figure 3.1 compares the end-to-end performance of QoSMIC, PIM and DVMRP on the MBone topology. Figure 3.1(a) shows the fraction of receivers that obtain end-to-end delays lower than or equal to a certain value, expressed as a function of the delay value on the X-axis. Figure 3.1(b) shows the end-to-end path-length in the same way. Table 3.2 reports the maximum and average end-to-end delay seen by the receivers.

We observe that QoSMIC provides end-to-end delays and hop-counts close to the performance of the trees created by DVMRP, which exhibit approximately half the delay of PIM’s shared trees. We remind the reader that the network state created by QoSMIC is the same as PIM per node (and smaller if we consider the fewer links in the tree), while DVMRP requires more state by a factor of \( s \), the number of sources. However, there is also very little difference between QoSMIC(delay) and QoSMIC(hop). On investigation, we found the reason is due to the smaller number of alternate paths available in the MBone topology. Most of the MBone topology is composed of stub-trees, hanging of a sparse backbone mesh.

![Figure 3.1](image)

(a) End-to-end delay. (b) End-to-end path.

Figure 3.2: Fraction of nodes below an end-to-end performance value using the uniform delay distributions: Waxman flat euclidean graphs (node degree 2.5).

From Figure 3.2 to Figure 3.4, we show the end-to-end performance of QoSMIC, PIM and DVMRP on the Waxman flat random graphs with average node degree varying from 2.5 to 5.52. This series of graphs clearly indicate that the end-to-
3.2 Results

Figure 3.3: Fraction of nodes below an end-to-end performance value using the uniform delay distributions: Waxman flat euclidean graphs (node degree 3.78 and 4.32).

Figure 3.4: Fraction of nodes below an end-to-end performance value using the uniform delay distributions: Waxman flat euclidean graphs (node degree 4.94 and 5.52).
### 3.2 Results

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Metric</th>
<th>PIM</th>
<th>DVMRP</th>
<th>QoSMIC(hop)</th>
<th>QoSMIC(delay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mbone graph</td>
<td>Max</td>
<td>3.986</td>
<td>2.027</td>
<td>2.630</td>
<td>2.613</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>1.873</td>
<td>0.968</td>
<td>1.015</td>
<td>1.006</td>
</tr>
<tr>
<td>Waxman flat node degree 2.5</td>
<td>Max</td>
<td>1.517</td>
<td>0.875</td>
<td>1.111</td>
<td>1.01</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.73767</td>
<td>0.3703</td>
<td>0.45634</td>
<td>0.4367</td>
</tr>
<tr>
<td>Waxman flat node degree 3.78</td>
<td>Max</td>
<td>1.216</td>
<td>0.748</td>
<td>1.129</td>
<td>0.973</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.623</td>
<td>0.335</td>
<td>0.433</td>
<td>0.401</td>
</tr>
<tr>
<td>Waxman flat node degree 4.32</td>
<td>Max</td>
<td>0.853</td>
<td>0.465</td>
<td>0.696</td>
<td>0.629</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.441</td>
<td>0.228</td>
<td>0.319</td>
<td>0.277</td>
</tr>
<tr>
<td>Waxman flat node degree 4.94</td>
<td>Max</td>
<td>0.707</td>
<td>0.506</td>
<td>0.776</td>
<td>0.664</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.3795</td>
<td>0.2154</td>
<td>0.289</td>
<td>0.2745</td>
</tr>
<tr>
<td>Waxman flat node degree 5.52</td>
<td>Max</td>
<td>0.763</td>
<td>0.461</td>
<td>0.723</td>
<td>0.665</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td>0.369</td>
<td>0.2067</td>
<td>0.2808</td>
<td>0.2496</td>
</tr>
</tbody>
</table>

*Table 3.2: The end-to-end delay observed by receiver in seconds using the uniform delay distributions.*
end performance of QoSMIC is always better as compared to PIM regardless of the network topologies.

In Figure 3.3(a), when we compare the end-to-end delay performance of QoSMIC with PIM, QoSMIC(hop) already improves the blocking probability for delay sensitive applications considerably. For example, for a 0.4 second delay bound, the blocking is reduced from 70% blocked receivers (for PIM) to 28% blocked receivers. An alternative way of viewing the same data is that the average delay is reduced by about 30%. In addition, because of the dynamic (non-core) aspect of QoSMIC(hop), the multicast tree is able to adapt to the position of the receivers better than PIM-SM's Shared Tree.

On the same graph, when we look at QoSMIC(delay), the blocking for the same delay bound of 0.4 second is further reduced from 28% to 20%. This represents a 2.7 fold increase in the number of satisfied participants over PIM-SM. The additional improvement in blocked calls comes from the consideration of the QoS information.

Furthermore, the end-to-end delay of QoSMIC is a good compromise between the core-based tree and the source-based tree, allowing the more scalable shared tree approach to be used for delay sensitive applications.

### 3.2.2 Bi-modal Delay Distribution

Measurements of Internet traffic [9] suggest that Internet traffic load is skewed, with most links underutilized, and a few links heavily congested. Though we have not seen similar measurement studies addressing Internet link delay, our intuition suggests that Internet link delay might also be skewed, since a highly loaded link is also likely to experience high delays on the link. With this in mind, we believe that the bi-modal distribution is a reasonable model to be used in our experiments.

For this set of experiments, we generate random numbers from a number of bi-modal distributions to represent the link delays. These distributions are parameterized by the percentage of high delay links. For example, the bi-modal(20) distribution contains 20% of links whose delays are uniformly distributed in the range of [90ms,100ms], while the rest of the 80% links have delays that are uniformly dis-
tributed in the range of [0ms, 10ms]. We evaluate the end-to-end performance of QoSMIC, PIM-SM, and DVMRP for each of these distributions.

Figure 3.5: Average end-to-end delay vs. the percentage of high delay links using the bi-modal delay distributions: Waxman flat euclidean graph (node degree 3.78).

Figure 3.5 shows the average end-to-end delay as a function of the percentage of high delay links between the protocols. We also compute the 95% confidence interval, and they are within 5% from the mean value.

We observe that the end-to-end delay of QoSMIC is as good or even better than any other protocols for the bi-modal(10), (20), and (30) distributions. In other words, in a network with a small fraction of highly congested links, QoSMIC performs as well or even better than DVMRP, though DVMRP is using a more expensive per-source tree approach.

To further verify our findings, we conduct the same measurement using the bi-modal(20) distribution in different waxman flat graph with average nodes degree varying between 2.5 to 5.94. Figure 3.6 shows the average end-to-end delay vs. the average node degree between the protocols. We also compute the 95% confidence interval. We observe that regardless of the topology, QoSMIC always does better in the end-to-end performance as compared to all the other protocols in question for the
3.2 Results

bi-modal(20) distribution.

The low range of the parameterized bi-modal distributions, that is, the bi-modal(10), (20), and (30) distributions, are interesting because they match what little data there is about Internet load and delay distributions. In addition, many different distributions found in real-life, such as distributions of file size, packet size, or movie popularity, appear to follow a similar 80-20 rule. The fact that QoSMIC performs especially well in this lower range is promising. However, we need more data on Internet link delay distributions to confirm this.

Figure 3.6: Average end-to-end delay as a function of the average node degree using bi-modal(20) distribution: Waxman flat euclidean graph.

3.2.3 Heavy-tail Delay Distribution

Measurements of network traffic over the last few years have revealed a number of surprises, one of them is the skewed distribution that we already mentioned in Section 3.2.2. The other surprise that is worth mentioning is the prevalence of the infinite variance phenomenon, or more generally, of heavy-tailed distributions in networking-related activities (e.g., CPU processor time, video frame size, call holding times, inter-packet times for interactive applications, burst size in data bulk transfer, WWW item
Table 3.3: The end-to-end delay observed by a receiver in seconds using the heavy-tailed delay distributions: Waxman flat euclidean graph (node degree 3.78).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Metric</th>
<th>PIM</th>
<th>DVMRP</th>
<th>QoSMIC(hop)</th>
<th>QoSMIC(delay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 0.5$, $\beta = 0$</td>
<td>Max</td>
<td>1.2306</td>
<td>0.713</td>
<td>0.7259</td>
<td>0.855</td>
</tr>
<tr>
<td>$\alpha = 1.5$, $\beta = 0$</td>
<td>Max</td>
<td>0.1827</td>
<td>0.104</td>
<td>0.13214</td>
<td>0.13035</td>
</tr>
<tr>
<td>$\alpha = 0.5$, $\beta = -1$</td>
<td>Max</td>
<td>3.1065</td>
<td>2.67</td>
<td>2.74</td>
<td>2.726</td>
</tr>
<tr>
<td>$\alpha = 0.5$, $\beta = 1$</td>
<td>Avg</td>
<td>0.279</td>
<td>0.1953</td>
<td>0.2414</td>
<td>0.1688</td>
</tr>
</tbody>
</table>

For our experiment, we use the Waxman flat graph with an average node degree of 3.78. We generate heavy-tailed random numbers using different parameters, e.g., the characteristic exponent $\alpha$, and the symmetry parameter $\beta$. The characteristic exponent measures the “thickness” of the tails of the distribution. Thus, the larger the value of $\alpha$, the less likely it is to observe values of the random variable, which are far away from its central location. $\beta$ is the skewness parameter. $\beta = 0$ implies a distribution symmetric about $\alpha$.

Table 3.3 reports the maximum and average end-to-end delay seen by a receiver for four different combinations of $\alpha$ and $\beta$. Figure 3.7 and Figure 3.8 give the fraction of receivers that obtain end-to-end delays equal to or lower than a certain value as a function of delay values on the X-axis.

We show that across all the heavy-tailed distributions, QoSMIC again outperforms PIM-SM. Moreover, in Figure 3.8(b), we see that QoSMIC performs the best among all the other protocols when $\alpha = 0.5, \beta = 1$. On investigation, we find out that for this set of random delay values generated with $\alpha = 0.5, \beta = 1$, there are 20% of highly congested links in the topology, whose link delays are between [90ms, 100ms]. This
3.2 Results

Figure 3.7: Fraction of nodes below an end-to-end delay value using the heavy tailed delay distributions ($\beta = 0$): Waxman flat euclidean graphs (node degree 3.78).

Figure 3.8: Fraction of nodes below an end-to-end delay value using the heavy tailed delay distributions ($\alpha = 0.5$): Waxman flat euclidean graphs (node degree 3.78).
interesting finding supports our earlier observation that the end-to-end performance of QoSMIC is the best as compared to all the other multicast routing protocols in question in a network with a small percentage of highly congested links. For the other heavy-tailed distributions presented in Table 3.3, the percentage of highly congested links (link delay between [90ms,100ms]) are about 30%-40% for $\alpha = 0.5$, and for $\alpha = 1.5, \beta = 0$, all the links in the topology are not heavily congested.

3.3 Summary

In this chapter, we examine and compare the end-to-end performance of QoSMIC, PIM-SM and DVMRP in various network topologies. We show that QoS-sensitive routing protocols can reduce call blocking significantly for applications with strict QoS requirements, in some cases increasing the number of satisfied participants by a factor of three as compared to PIM-SM. If we imagine a commercial scenario, with paying customers, it is easy to see the importance of QoS-sensitive routing protocols. Further, QoSMIC exhibits better end-to-end performance than PIM-SM regardless of the network topologies and different assumptions on the link delay distributions. We also find out that in a network with approximately 20% of highly congested links, QoSMIC would outperforms both DVMRP and PIM-SM. This is in spite of the fact that DVMRP is using per source trees and QoSMIC is using a shared tree. However, in the absence of real Internet traffic measurements, we cannot make any strong claim of the applicability of this result to the Internet.

The performance gain of QoS-sensitive routing protocols, such as QoSMIC, comes from two aspects. First, QoSMIC identifies a number of paths and constructs a tree in a dynamic way that adapts to the location of the group members. QoSMIC searches for paths between the new router and the neighboring routers of the tree, instead of joining to a pre-determined core. Second, given the multiple path routing, QoSMIC improves the routing further by selecting the path that meets the application's QoS criteria. This is done in a way that eliminates all the traditional disadvantages of load-sensitive routing, such as instability, looping, and increased routing state in the
network.

In the next chapter, we will examine the overhead complexity of QoSMIC and show that QoSMIC causes a reasonable and controllable increase in the number of control messages without degrading the performance of QoS-sensitive routing.
Chapter 4

Measurement of Overhead Complexity

In this chapter, we evaluate the overhead complexity of QoSMIC. We show that QoSMIC causes a reasonable and controllable increase in the number of control message without degrading the performance of QoS-sensitive routing. We evaluate the effectiveness of various Candidate selection mechanisms used in the Multicast Tree Search procedure and propose good parameter settings that can be used in the protocol across a range of network sizes and topologies. In section 4.1, we provide an overview of the simulation model for this set of experiment. In section 4.2, we present our simulation results. In section 4.3, we summarize our work in this chapter.

4.1 Simulation Model

In this section, we provide an overview of the simulation model used in this set of experiments.

For our experiments, we examine the overhead complexity of QoSMIC and YAM. As mentioned in Chapter 1, YAM was the first multi-path protocol, but the scalability of its control message overhead is a possible weakness, since it relies on an extensive Local Search procedure to find alternate paths for a new receiver. QoSMIC can be seen as a superset of YAM, since it employs both the Local Search and the Multicast
Tree Search procedures to locate possible Candidates for the receiver to join to the group. The Local Search procedure in QoSMIC can be limited to a very small scope which will result in less overhead in the network. The Multicast Tree Search procedure is designed to be scalable, to avoid the potentially exponential growth of overhead with search scope found in the Local Search procedure.

We focus on the tree setup portion of the protocol. For the Candidate search procedures, the Local Search procedure and the Multicast Tree Search procedure are operated in a serial manner. If the Local Search falls short, we can fall back to the Multicast Tree Search procedure. During the Multicast Tree Search procedure, we assume that the Manager router does not have sufficient knowledge of the tree and the network topology, such that the Candidate selection is done in a distributed fashion by employing various Candidate selection mechanisms (as described in Chapter 2 on Candidate selection) to reduce the number of Candidates and the number of BID messages.

We perform the experiments in several steps. First, we use the full blast Multicast Tree Search procedure\(^1\) without turning on any of the specific Candidate selection mechanism mentioned. This will give us a good indication of how bad the overhead can be from the Multicast Tree Search procedure without any of these mechanisms. In addition, we vary the TTL value to see its effect on the overhead resulted from the Local Search procedure, and based on that, we can pick a reasonable TTL value to limit the scope of the Local Search procedure. Then, we gradually turn on each Candidate selection mechanism or combinations of them to observe the reduction on the number of Candidates and the number of BID messages occurred during the Multicast Tree Search procedure. In all cases, the Local Search is used before turning on the Multicast Tree Search procedure.

We measure the following metrics in each step of the experiment. Here we want to remind the readers that the control messages are counted separately for each hop they traverse, such that a single message traveling three hops would result in a metric

\(^1\)In the full blast Multicast Tree Search procedure, the BID-ORDER message is sent to every node in the tree and every in-tree node sends a BID message to the new router.
of three.

- **Setup time.** The setup time is defined as the average time elapsed between a receiver's first joining request until it receives the first data packet from the multicast group.

- **Number of hops per successful join.** This is the average number of hops used for a receiver to successfully connected onto the existing multicast tree.

- **Number of BID-REQ message.** This is the average number of BID-REQ messages transmitted in the network for each receiver.

- **Number of MJOIN message.** This is the average number of MJOIN messages transmitted in the network for each receiver who falls back to the Multicast Tree Search procedure after the Local Search fails.

- **Number of BID-ORDER message.** This is the average number of BID-ORDER messages transmitted in the network for each receiver who relies on Multicast Tree Search procedures.

- **Number of BID message.** This is the average number of BID messages transmitted in the network for each receiver.

- **Percentage of successfully connected receivers.** The percentage of successfully connected receivers from the Local Search procedures and the Multicast Tree Search procedure.

We use both real and artificial graphs in our experiment. First, we use the same MBone topology that was used in the previous set of experiments, which consists of 255 nodes and 266 duplex links. Second, we use a Waxman flat Euclidean graph with 100 nodes and an average node degree of 3.4.

The link delay and link bandwidth are set to arbitrary values and are identical across all the links in the topology, since the focus of our measurements is on the overhead which is independent of these values.
4.2 Results

In this section, we present our simulation result. We use the same experiment setup as was described in chapter 3.

4.2.1 Full Blast Multicast Tree Search

We show the experimental results for the effect of increasing values of TTL on the overhead complexity of the Local Search procedure in the Waxman flat graph in Figure 4.1 and Figure 4.2. We did not compute the confidence interval in these figures, because we are not interested in the absolute numbers, but the relative effect between different TTL values.

In Figure 4.1(b), we observe that, with $TTL = 0$, QoSMIC depends solely on the Multicast Tree Search procedure to successfully connect a receiver onto the multicast tree, while YAM fails to make any connections, since it relies uniquely on the Local Search procedure.

As the TTL value gradually increases, YAM succeeds in certain percentage of connections, but not all until the TTL reaches a value of 5 and beyond. QoSMIC is able to succeed in every single connection, because it has both the Local Search and the Multicast Tree Search procedures to rely on. This suggests that the Local Search alone would need large TTL values, which increases the overhead complexity significantly. This can be seen from the increase in the number of BID-REQ message as the value of TTL increases. For example, in Figure 4.2(a), for a TTL value of 5, the maximum number of BID-REQ messages can be as large as 238 for the network used in this experiment.

Thus, we want to limit the scope of the Local Search procedure to take advantage of the many “short” joins, but we also need a second search mechanism to provide solutions when the Local Search falls short. Based on our experimental data shown in Figure 4.1(b), with a TTL value of two hops, QoSMIC can successfully connect 50% of all receivers onto the existing multicast tree\(^2\). Therefore, a Local Search of

\(^2\)We designed the simulator such that it can count the number of successful joins from the Local
Figure 4.1: Measurement results using full blast msearch as a function TTL value: Waxman flat Euclidean graph (node degree 3.4).
4.2 Results

Figure 4.2: Overhead complexity using full blast msearch as a function TTL value: Waxman flat euclidean graph (node degree 3.4).
two hops is a reasonable choice for QoSMIC, given the specific network topology and
the multicast group size used in the experiment, while YAM would have to go up to
five hops.

In Figure 4.1(a), we observe that the setup time of QoSMIC is generally longer as
compared to YAM. This is due to the fact that the receivers whose path are longer
than the given TTL scope, have to switch to the Multicast Tree Search procedure to
get connected after the Local Search fails. However, as we mentioned in chapter 2.
that both the Local Search and the Multicast Tree Search procedures can be operated
in parallel to reduce the setup latency.

As mentioned in Section 1.1, the long setup latency of QoSMIC is well suited to
long-lived applications, such as video conference. In contrast, applications, such as
web-browsing and email checking, are not well suited to use QoSMIC as their routing
protocol, because of their short lived sessions.

In Figure 4.3, we show an approximate model to compare the set-up latency
between QoSMIC and PIM-SM. If the join latency of PIM-SM is $X$, then the latency
of QoSMIC is approximately $4X + C$, where $C$ is the timer interval for the Local Search
procedure, and we assume that the distance from the New router to the Manager.
from the Manager to the Candidate, and from the Candidate to the New router in
Figure 4.3(a) are all about comparable to the distance from the New router to the RP
in Figure 4.3(b). The measured latency of QoSMIC has a maximum value of slightly
larger than 2 seconds and an average value of 1 second, which is well within the the
acceptable limits for the long-lived applications that we discussed in the previous
paragraph. The scaling behavior is that the worst case latency grows linearly with
the diameter of the network topology, which holds the same for PIM-SM, though with
different constants.

Similar experiments were also performed in the MBone graph for increasing values
of TTL. Based on the experimental results, a reasonable choice of the Local Search
scope is 3 hops, where QoSMIC can successfully connect 58% of all receivers onto the
existing multicast tree through the Local Search procedure alone with a reasonable
4.2 Results

number of BID-REQ message.

![Diagram](image)

Figure 4.3: An approximate model to compare the set-up latency between QoSMIC and PIM-SM.

4.2.2 Multicast Tree Search with the Directivity Mechanism

We turn on the Multicast Tree Search procedure with the directivity mechanism for Candidate selection. We use a $TTL = 2$ to limit the scope of the Local Search procedure. We vary the value of distance threshold, used to discard the BID-ORDER message (see section 2.2.4), to observe its effect on the control overhead. In Figure 4.4(a) to Figure 4.7(a) we compare various metrics between the full blast msearch, $threshold = 5$, and $threshold = 0$ for the Waxman flat graph. The bars showing the performance of the directivity mechanism with $threshold = 5$ and $threshold = 0$ are labeled as $d(5)$ and $d(0)$ respectively. We did not compute the confident interval for these figures, because we are not interested in the absolute number, but the relative effect between different Candidate selection mechanisms. The results are not very sensitive to the value of the threshold chosen. We obtained the same results for threshold values from 10 to 3. Given a distance threshold of 0, the worse case number of BID-ORDER messages is reduced from 20 to 15. However, we are not impressed by the reduction in the worst case number of BID messages\(^3\). The quality of the tree is not degraded from the use of the directivity mechanism, even with $threshold = 0$.

\(^3\)Based on our experiment data, the worst number of BID message always comes from the Multicast Tree Search procedure.
4.2 Results

This can be seen from Figure 4.5(a), where the maximum path length of join remains the same as compared to the result obtained from the full blast msearch.

We perform similar measurement by turning on the directivity mechanism for the MBone topology, using \(TTL = 3\) for the Local Search procedure, as per the conclusions of Section 4.2.1. The series of results are presented in Figure 4.4(b) to Figure 4.7(b). When the threshold value reaches 0, the worst case BID-ORDER message is reduced dramatically as compared to the result from the full blast msearch. The quality of the tree is not degraded from the use of the directivity mechanism.

We also performed some preliminary experiments with “dynamic directivity”, where instead of assigning a pre-determined distance threshold value in the protocol, we let the protocol itself to determine the value of distance threshold to be used as the multicast tree evolves overtime. However, in the network topologies that we tested, \(threshold = 0\) continued to perform as well or better than our first implementation of “dynamic directivity”. While, we believe that “dynamic directivity” is important, since \(threshold = 0\) may not work well for extremely large trees, we were not able to explore this issue sufficiently due to lack of time.

4.2.3 Multicast Tree Search with the Local Minima Mechanism

We turn on the Multicast Tree Search procedure with the local minima mechanism for Candidate selection. In Figure 4.4 to Figure 4.7, we compare the results between full msearch and local minima (labeled as “l”) on the Waxman flat graph and the MBone graph respectively.

Our experiment result show that the local minima mechanism has no impact in reducing the number of BID-ORDER message. This is because the local minima mechanisms only determines if a node that receives the BID-ORDER message should be a candidate or not, it does not kill the BID-ORDER message as the previous directivity mechanism sometimes does. However, local minima alone can reduce the worst case number of BID message better than directivity alone. For example, with
4.2 Results

the Waxman flat graph, the worst case number of BID message is reduced from 66 to 41, while it is reduced from 66 to only 50 if using the directivity mechanism alone. Note, that the actual number of bids received by the new router is smaller (the actual number of bids received is the same as the number of Candidates selected shown in Figure 4.4), since we count the overhead of BID messages by the number of hops they traverse.

Another observation is that the optimality of the multicast tree is not affected from using the local minima mechanism regardless of the topology experimented. The maximum path length of join remains the same as compared to just having the full msearch.

4.2.4 Multicast Tree Search with the Combination of Directivity and Local Minima Mechanisms

The previous experiments show that directivity alone with a threshold of 0 is able to reduce the worst case BID-ORDER message, while local minima alone is able to reduce the worst case BID message. In both cases, the optimality of the tree is not affected by the use of any of these mechanisms. We expect that the combination of these two orthogonal mechanisms could further create a greater impact on the reduction of the control overhead regardless of the topology.

In Figure 4.4 to Figure 4.7, we compare the results between full msearch and the combination of the directivity and local minima mechanisms (labeled as “d+1”) on the Waxman flat graph and the MBone graph respectively. The reduction in both the number of BID-ORDER message and the number of BID message is just as impressive as expected. The average number of Candidates is reduced to 1.767 as compared to 2.62 in full msearch, and the worst case number of Candidate is reduced to 8 as compared to 12 in full msearch. Furthermore, the optimality of the tree is not degraded.

Based on these experimental result, we conclude that the combination of the directivity (using threshold = 0) and the local minima mechanisms is very effective
in reducing the control overhead during the Multicast Tree Search procedure.

![Graphs showing number of candidates and BID messages](image)

(a) Waxman graph.  
(b) MBone graph.

Figure 4.4: Comparison on the Number of Candidates between different Candidate selection mechanism. This is also the number of BID messages received by the new router. (For this series of figures, we use the following representation on the x axis: “d(5)” refers to the directivity mechanism with a threshold value of 5, similarly for d(0); “l” refers to the Local Minima mechanism; “d+l” refers to the combination of directivity and the local minima mechanisms; and “d+l+f” refers to the combination of directivity, local minima and the fractional choice mechanisms.)

### 4.2.5 Multicast Tree Search with the Combination of Directivity and Local Minima and Fractional Choice mechanisms

We turn on the Multicast Tree Search procedure with all three orthogonal mechanisms: directivity, local minima and fractional choice. In the fractional choice mechanism, the number of Candidates is reduced to an arbitrary percentage of the original (i.e., before turning on fractional choice), by coin-tossing. We implement a dynamic version of fractional choice, where the parameter used in the coin-tossing is determined by the mechanism based on the number of Candidates seen in the previous JOIN attempt. We call this the dynamic fractional choice. Initially, the coin-tossing
4.2 Results

Figure 4.5: Comparison on the Number of hops per successful join between different Candidate selection mechanism.

Figure 4.6: Comparison on the Number of BID-ORDER messages between different Candidate selection mechanism.
4.2 Results

Figure 4.7: Comparison on the Number of BID messages between different Candidate selection mechanism.

Parameter is selected so that no candidates are lost. As the tree grows very large, the parameter is adjusted to put an upper bound on the number of candidates selected.

In Figure 4.4 to Figure 4.7, we show the comparisons between the full msearch and the combination of all three orthogonal mechanisms (labeled as “d+1+f”) on the Waxman flat graph and the MBone graph respectively.

Compared to the results on using the combination of directivity and local minima, adding fractional choice does not improve the overhead significantly. The results also show that fractional choice has no effect on the optimality of the tree. Actually, in this topology, because the number of Candidates remain small, the fractional choice mechanism is rarely turned on, leading to very little impact on either the overhead or the tree optimality. However, we want to point out to the readers that without fractional choice, the overhead could grow arbitrarily large when the size of the tree grows large. We suggest that fractional choice should be implemented in the protocol, as a means of putting an upper bound on the overhead in large topologies, without affecting the performance in smaller networks.

Comparing the overhead cost between QoSMIC and PIM-SM is difficult, because PIM-SM’s overhead is a periodic refresh process. QoSMIC has an equivalent refresh process that is roughly the same in terms of PIM-SM’s overhead. QoSMIC has an
4.3 Summary

additional search overhead, which is a one-time cost per receiver, during the session setup phase. From our experimental results, the amount of this additional overhead is roughly the same as the refresh overhead that can be attributed to a single receiver over 8 refresh intervals for the Waxman graph, and 6.4 refresh intervals for the MBone graph. This gives us a rough method of comparing the additional overhead introduced by QoSMIC to that already existing in PIM-SM. Clearly, for long lived sessions, this additional overhead is negligible.

4.3 Summary

In this chapter, we examine the overhead complexity of QoSMIC. We show that the scope of the Local Search procedure can be limited to a small TTL value, such as 2 hops for the waxman flat graph with an average node degree of 3.4, and 3 hops for the partial MBone topology used in our experiment. This way, the overhead due to the Local Search procedure can be reduced significantly, while we still enjoy the benefit of many "short" joins.

We present an approximate model to compare the join latency between QoSMIC and PIM-SM. We show that the join latency of QoSMIC is larger than PIM-SM. There is an extra constant additive component due to the Local Search procedure and a multiplicative constant of 4 because of the four way communication. We can eliminate the additive constant by having both the Local Search procedure and the Multicast Tree Search procedure operated in parallel, though at the expense of higher control overhead. However, for the long-lived multicast applications, our measured join latency is well within the acceptable limits.

In Multicast Tree Search procedure, we assume the Manager works in a distributed way to select the Candidates. First, we examine the overhead complexity for the full blast msearch. Then we turn on different Candidate selection mechanisms or a combination of them to see the effect on the overhead.

Directivity alone with a threshold = 0 is able to reduce the worst case number of BID-ORDER messages, while local minima alone is able to reduce the worst case
number of BID messages. The combination of these two works the best in reducing both the number of BID-ORDER message and the number of BID message. Moreover, even though we worked with networks as large as NS could support (MBone graph N=255, L=266; Waxman graph N=100, L=170), the combination of directivity and local minima was sufficient to reduce the number of BIDs to the point where dynamic fractional choice had no further effect.

The combination of directivity (with a threshold of 0) and local minima works well enough to reduce the overhead from the Multicast Tree Search procedure. This additional overhead is comparable to 6-8 times the ongoing per refresh interval overhead already found in PIM-SM, for the topologies studied, and is negligible for long lived sessions. However, this conclusion is valid within the scope of our experiment. For large graphs, we believe it is important to have both the “dynamic directivity” and the “dynamic fractional choice” to provide an upper limit on the amount of overhead introduced in the Multicast Tree Search procedure. However, due to the limitations of our simulator, we did not experiment with extremely large graphs.
Chapter 5

Measurement of Routing Efficiency

In this chapter, we evaluate the effectiveness of QoSMIC in utilizing network resources. The effectiveness is measured in terms of the percentage of receivers prevented from joining the multicast session due to insufficient resources. We show that QoSMIC can always accommodate more incoming calls as compared to the existing Shared Tree protocol, such as PIM-SM. We experiment with different link load distributions to observe its effect on the sensitivity of our results. In section 5.1, we provide an overview of the simulation model. In section 5.2, we discuss our simulation results. In section 5.3, we summarize our work in this chapter.

5.1 Simulation Model

In this section, we provide an overview of the simulation model.

For our experiment, we evaluate the routing efficiency of QoSMIC and PIM-SM. In QoSMIC, we use the *Widest-Shortest* criteria for path selection. In other words, among the paths with sufficient resources to support the desired QoS, QoSMIC will select the one with the minimum hop count from the existing tree to the joining node. If there is a tie, then QoSMIC selects the path with the maximum bottleneck bandwidth. In order to focus on resource usage, we assume the end-to-end delay bound is arbitrarily large, such that no call is blocked due to violation of the end-to-end delay bound. We augment the implementation of PIM-SM with a simple
admission control function, in order to detect receivers that cannot find adequate resources on the paths selected by PIM-SM.

We measure the percentage of receivers prevented from joining the multicast session due to insufficient network resources. We believe the above performance metric will provide a good indication on how efficiently that QoSMIC utilizes the network resources as compared to other Shared Tree protocols, such as PIM-SM.

We use real and artificial graphs in our experiment. First, we use the same partial MBone graph as of previous set of experiment, which is composed of 255 nodes and 266 duplex links. Second, we experiment with the same series of Waxman flat graph that appeared in Table 3.1.

Since there is no exact measurement available for the current Internet traffic load pattern, we experiment with several different link load distribution functions: uniform distribution, bi-modal distribution, and truncated heavy-tailed distribution. We assign a bandwidth utilization factor to each link from these distributions to capture the effect of current network load conditions. We then simulate a single session per experiment instance and measure the number of calls admitted under the current network state.

The link delay is assigned to an arbitrary value of 10ms across all links in the topology, since its value has no effect on the performance metric we intend to measure in the experiments.

5.2 Results

In this section, we present our simulation results, corresponding to the three different link load distribution functions. We use the same experiment setup as was describe in Section 3.2.

5.2.1 Uniform Load Distribution

We perform seven experiments with the uniform load distribution. these corresponds to one MBone graph and six Waxman flat graphs of different average degree of nodes.
5.2 Results

The link utilization factor is uniformly distributed in the range of $[0, 1]$.

Figure 5.1 compares the percentage of blocked receivers of QoSMIC between PIM-SM as a function of average degree of nodes on the Waxman flat graph. The error bar indicates the 95% confidence intervals. Table 5.1 shows the percentage of blocked receivers for the MBone graph.

We observe that QoSMIC always yields more incoming calls as compared to PIM-SM regardless of the topology used. For the series of Waxman flat graphs, there is a decreasing trend in the percentage of blocked receivers for QoSMIC as the graph becomes more dense. However, for PIM-SM, after the initial drop in the percentage of blocked receivers from node degree 2.5 to node degree 3.75, there is no further drop. We believe this is because PIM-SM is not able to make use of the larger number of alternate paths available.

![Figure 5.1: Percentage of blocked receivers as a function of average degree of nodes using the uniform load distribution: Waxman flat euclidean graph.](image)

5.2.2 Bi-modal Load Distribution

For the reasons mentioned in Section 3.2.2, our intuition suggests that the bi-modal load distribution function is a reasonable model to use for our experiments.
### 5.2 Results

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Metric</th>
<th>PIM</th>
<th>QoSMIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBone</td>
<td>Avg</td>
<td>28.7</td>
<td>20.9</td>
</tr>
</tbody>
</table>

Table 5.1: Percentage of blocked receivers using the uniform load distribution: MBone graph.

For our experiments, we generate different distributions of bi-modal random numbers in the range of $[0,1]$ to represent the link utilization factor for each link. For example, a bi-modal(20) indicates there are 20% of links in the topology have a link utilization factor between $[0,0.1]$, while the rest 80% of links have a utilization factor between $[0.9,1]$. Figure 5.2 presents the percentage of blocked receivers as a function of the percentage of highly utilized links for the Waxman graph and the MBone graph respectively. Figure 5.3 shows the percentage of blocked receivers as a function of average degree of nodes for different Waxman flat graphs with 20% highly loaded links. The error bars represent the 95% confidence intervals.

We observe that, regardless of topology and different network load conditions, QoSMIC always admits more calls as compared to PIM-SM. However, in the MBone graph, the difference in the number of blocked calls between QoSMIC and PIM-SM is not as great as compared to the results from the Waxman flat graph. This is due to the smaller number of alternate paths available in the MBone topology. In Figure 5.3, as the graph becomes dense, the percentage of blocked receivers is reduced for both QoSMIC and PIM-SM. This is exactly what we expect, since in a dense graph with more alternate path, the probability to block an incoming call is also likely to be reduced. However, the reduction for PIM-SM is from 30.3% to 20% (a 33% reduction) while the reduction for QoSMIC is from 17.8% to 3.5% (80% reduction), indicating that QoSMIC has a better ability to make use of the alternate paths.

#### 5.2.3 Truncated Heavy-tail Load Distribution

For this set of experiment, we generate heavy-tail distributed random numbers with different characteristic exponent $\alpha$ and symmetry parameter $\beta$. We truncate those
5.2 Results

Figure 5.2: Percentage of blocked receivers vs. percentage of highly loaded links using the bi-modal load distribution.

Figure 5.3: Percentage of blocked receivers as a function of average degree of nodes with 20% highly loaded links: Waxman flat euclidean graph.
5.3 Summary

extremely large numbers from the set of heavy-tailed data in order to get a set of number in the range of [0,1], and we assign these numbers to each link in the topology to represent the link utilization factor. We mainly want to see the effect of high variance load distribution on our measurement results.

Table 5.2 and Table 5.3 report the average percentage of blocked receivers for the MBone graph and the Waxman flat graph respectively. Not surprisingly, regardless of topology, QoSMIC can admit more calls for different combinations of $\alpha$ and $\beta$.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Metric</th>
<th>PIM</th>
<th>QoSMIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 0.5, \beta = 0$</td>
<td>Avg</td>
<td>51.7</td>
<td>42.1</td>
</tr>
<tr>
<td>$\alpha = 1.5, \beta = 0$</td>
<td>Avg</td>
<td>1.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\alpha = 0.5, \beta = -1$</td>
<td>Avg</td>
<td>59.5</td>
<td>44.1</td>
</tr>
<tr>
<td>$\alpha = 0.5, \beta = 1$</td>
<td>Avg</td>
<td>46.7</td>
<td>41.2</td>
</tr>
</tbody>
</table>

Table 5.2: Percentage of blocked receivers using the heavy-tail load distribution: MBone graph.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Metric</th>
<th>PIM</th>
<th>QoSMIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha = 0.5, \beta = 0$</td>
<td>Avg</td>
<td>26.4</td>
<td>9.5</td>
</tr>
<tr>
<td>$\alpha = 1.5, \beta = 0$</td>
<td>Avg</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>$\alpha = 0.5, \beta = -1$</td>
<td>Avg</td>
<td>23.8</td>
<td>12.6</td>
</tr>
<tr>
<td>$\alpha = 0.5, \beta = 1$</td>
<td>Avg</td>
<td>24.8</td>
<td>8.8</td>
</tr>
</tbody>
</table>

Table 5.3: Percentage of blocked receivers using the heavy-tail load distribution: Waxman flat Euclidean graph (node degree 3.78).

5.3 Summary

In this chapter, we evaluate the routing efficiency of QoSMIC and compare it with PIM-SM under different network load conditions. We show that QoS-sensitive routing
can reduce the percentage of blocked receivers significantly, allowing more calls to be admitted into the network. This means more revenue for the network provider.

The gain in routing efficiency of QoS-sensitive routing protocol, such as QoSMIC, comes from two aspects. First, QoSMIC identifies several paths and constructs the multicast tree in a dynamic way that adapts to the location of the group members. QoSMIC searches for paths between the new router and the neighboring routers of the tree, instead of joining a pre-determined core. Second, given multiple paths between the tree and the new router, QoSMIC collects dynamic routing information along each path, and selects the one that best meets the application’s requirement.
Chapter 6

Conclusions

In our work, we implement the QoSMIC protocol in a detailed simulation environment and investigate its performance under realistic network conditions. We perform an in-depth study of QoSMIC in three different areas. First, we look at the end-to-end performance of QoSMIC across a range of network topologies and under different assumptions of the network link delay distribution functions. Second, we examine the overhead complexity of QoSMIC to show that it can be implemented scalably. Finally, we evaluate the efficiency of QoSMIC in utilizing network resources under different network load conditions.

6.1 Contributions

The simulation result from our work lead to the following conclusions and observations.

1. *End-to-end performance*. We investigate the end-to-end performance of QoSMIC and compare it with existing protocols, such as PIM-SM and DVMRP. We experiment with both real and artificial graphs and use different assumptions of the link delay distribution functions. Our results show that QoS-sensitive routing protocols can reduce the call blocking significantly for applications with strict QoS requirements, in some cases, almost tripling the number of satisfied
participants for a given delay requirement as compared to PIM-SM. Furthermore, QoSMIC always exhibits better end-to-end performance than core-based tree protocols, such as PIM-SM, regardless of the network topologies and different assumptions on the link delay distributions. From our experimental data, we also show that QoSMIC performs the best among all the other routing protocols in question in a network with around 10%-30% highly congested links. Based on these experimental results, we believe that delay sensitive applications will benefit from QoS-sensitive routing, such as QoSMIC can provide. If we imagine a commercial scenario, with paying customers, it is easy to see the importance of QoS-sensitive routing protocols.

2. Overhead Complexity. We examine the overhead complexity of QoSMIC. We show that the Local Search procedure of QoSMIC can be limited to a small scope. In this way, the overhead due to the Local Search can be reduced significantly, while at the same time we can still enjoy the benefit of many "short" joins. In the Multicast Tree Search procedure, we look at the effect of different Candidate selection mechanisms. We assume that the Candidate selection is done in a distributed way by the Manager router. Our experimental results show that directivity alone with a threshold = 0 is effective to reduce the worst case number of BID-ORDER messages, while local minima alone is effective in reducing the worst case number of BID messages. The combination of directivity and local minima mechanisms works the best to reduce both the number of BID-ORDER messages and the number of BID messages. Our results also show that the fractional choice does not add any further improvement when used in combination with the above two mechanisms for the set of graphs we used. Here we want to point out to the reader that these results are limited to the size of graphs that we used in our simulations. For large networks, we believe that both the "dynamic directivity" and the "dynamic fractional choice" may be useful to provide an upper limit on the amount of overhead introduced in the Multicast Tree Search procedure.
3. **Routing efficiency.** We evaluate the effectiveness of QoSMIC and PIM-SM in utilizing network resources across a range of network topologies and under different assumptions of network load conditions. Our experimental results show that QoS-sensitive routing can reduce the percentage of blocked receivers significantly, allowing more calls to be admitted to the network. The reduction is achieved without increasing the amount of resources per receiver used in the network. This means more revenue for the network provider.

### 6.2 Future Work

We want to extend the capability of our simulator to make it feasible for large graphs with thousands of nodes. For the Candidate selection mechanism, we want to evaluate the dynamic directivity and dynamic fractional choice in more depth using large graphs. With the experience we gained through this work, we intend to implement a prototype in the near future.
Bibliography


