QoS-Aware Multicast Routing for the Internet: The Design and Evaluation of QoSMIC

Shuqian Yan, and Michalis Faloutsos and Anindo Banerjea

Abstract

One of the main problems of the current Internet infrastructure is its inability to provide services at consistent QoS levels. At the same time, many emerging Internet applications, such as tele-education, and tele-conferencing, require multicast protocols that will provide the necessary Quality of Service (QoS). In this paper, we propose QoSMIC, a multicast routing protocol for the Internet, that provides QoS-sensitive paths in a scalable, resource-efficient and flexible way. QoSMIC differs from the previous protocols in that it identifies multiple paths and selects the one that can provide the required QoS. Two other key advantages of QoSMIC are its flexibility and adaptivity. First, the distribution tree does not have to be rooted at a pre-selected *core* router. Second, we can trade-off between efficiency metrics depending on our needs; for example we can trade-off routing efficiency for a reduction in the control messages. Extensive simulations show that our protocol improves the resources utilization and the end-to-end performance compared to the current protocols. Specifically, our protocol reduces the call blocking probability by a factor of six and reduces the end-to-end delay by as much as 90% compared to the PIM protocol.

Keywords: Multicast Routing, Quality of Service, Protocol Evaluation

1 Introduction

In this paper, we propose a protocol to support multicast or group communications at high quality levels for the Internet. This capability is an essential part of the Next Generation Internet and the services it is supposed to provide. Multicasting is a crucial component of the networking infrastructure. Several emerging applications, such as tele-conferencing, and tele-education, are based on group communications. With multicasting, the same information is distributed to many receivers at the same time. As a result, a single multicast group can replace several unicast streams, reducing the number of packets carried by the network. In addition, the network should be able to support and guarantee the Quality of Service (QoS) that a user receives. It is clear that the commercial value and social impact of a service depends heavily on its ability to maintain a consistent quality level.

The emerging multimedia applications have stringent Quality of Service (QoS) requirements. For example, real-time applications demand bounded end-to-end delay [39] [29]. Other related end-to-end QoS metrics are the packet loss ratio and the required bandwidth of a connection, while end-to-end path length is a routing metric strongly correlated to QoS [46]. In this paper, we use the end-to-end delay as a representative of the end-to-end QoS requirements. If the delay bound requested by a user is not met, then we will consider that this user rejects the service.

Most current multicast routing protocols were not designed to support QoS-sensitive applications. Most protocols do not use *QoS-aware* routing; they do not consider QoS information in the selection of paths. For example, protocols like PIM, CBT and BGMP adopt the best-effort mentality of the Internet. These protocols provide a single path per user, and if the path does not have enough resources, the user's requirements cannot be met. Note, that QoS architectures like Int-serv [15] and Diff-serv [11] can not help in such a case, since they can only manage resources along the single path provided by the protocol.

Our goal is to define an efficient QoS-aware multicast routing protocol. The notion of efficiency includes several properties in trade-off relationships. In our work, we require our protocol to a) support QoS, b) be scalable, and c) use resources efficiently. The scalability is crucial if we want our protocol to work at the inter-domain level. Therefore, our protocol should not require link-state information. In addition, the protocol should support multiple sources for a given group, which corresponds to the many-to-many communication paradigm.

In this paper, we propose QoSMIC, an efficient and scalable multicast protocol for the Internet that supports QoS applications. QoSMIC is based on the notion of multiple path and QoS-aware routing. The multicast participant joining a group is offered multiple paths and selects the one that best satisfies its QoS requirements. The idea behind QoSMIC can be best illustrated through the analogy of driving a car at rush hour; we need routing choices, as well as radio traffic reports, to avoid congestion. QoSMIC provides both the routing choice, and the up-to-date information for each choice. Note, that QoSMIC can operate in a distributed environment and does not depend on

Shuqian Yan is at ANDA Networks 2708 Orchard Parkway, San Jose, CA 95134 . Michalis Faloutsos is at U.C. Riverside, Dpt Computer Science, Riverside, 92521, CA, USA. Anindo Banerjea is at TBD Networks, 43138 Christy St #114, Fremont, CA 94538. This material is based upon work supported by the National Science Foundation under Grant No. 9985195, DARPA award N660001-00-1-8936, Tata Consulting Services Inc. and DIMI matching fund DIM00-10071

any external protocol for providing QoS-information[1]. At the same time, the paths are pinned after setup to avoid oscillatory behavior. In a series of simulations, we show the advantages of QoSMIC compared to previous protocols such as PIM. Earlier versions of our work have appeared in [22, 53].

The rest of this paper is organized as follows. In section 2, we provide an overview of previous work. In section 3, we describe the architecture of our protocol. In section 4, we examine the performance of QoSMIC through series of simulations. In section 5, we summarize our work.

2 Background and Previous Work

In this section, we present some background on QoS and multicasting in the Internet environment. We classify and review briefly a number of protocols and highlight the needs that motivate our work.

The Internet provides best effort service, and does not guarantee any QoS level. Recently, as the importance of QoS has become apparent, the Internet community is in the process of developing QoS architectures (e.g., Int-serv [15], Diff-serv [11]). However, these architectures do not include QoS-aware routing mechanisms. Most ongoing QoS-routing work in the Internet community focuses on unicast traffic [16, 15].

Multicast routing is intended to be a mechanism for many-to-many communications in the Internet. The core function of the protocol is to create a distribution tree, i.e., to install state on a set of routers forming a tree that spans all the members of each multicast group. The efficiency of a multicast protocol can be defined by a set of metrics [21].

All multicast protocols currently under development within the Internet Engineering Task Force (IETF) are not QoS-aware. Furthermore, the majority of the multicast protocols even outside IETF are not QoS-aware. Protocols such as Protocol Independent Multicasting (PIM) [18, 17], Core Based Trees (CBT) [9, 8], Border Gateway Multicast Protocol (BGMP) [30], Multicast extensions of OSPF (MOSPF) [32] Simple Multicast (SM) [37], Multicast Internet Protocol (MIP) [36], Ordered Core Based Trees (OCBT) [40] and Hierarchical Internet Protocol (HIP) [41] do not consider QoS requirements in their routing. All of the above protocols construct shortest path trees rooted at a single router, which is called the *core*. Typically, the multicast trees are *shared*: packets from all the sources travel along the same bidirectional distribution tree. PIM also uses source-specific trees for traffic intensive sources.[2] More recently, source-specific routing has attracted significant attention. The EXPRESS protocol of Holbrook and Cheriton [27] creates source-based trees targeting single-source applications. Source-specific multicast (SSM) is a more recent protocol in the IETF community introduced by the developers of EXPRESS, Holbrook et al. [26].

The previous protocols were not designed to support applications with demanding QoS requirements. Their QoS performance suffers from two issues: a) their routing provides a single path based on static information, and b) their performance depends on the initial core selection.

The first effort to overcome the single-path routing weakness was the YAM protocol [12]. The protocol relies on a search procedure involving scoped multicast to find the closest portion of the tree to join. Our protocol is a superset of YAM, and has additional mechanisms that alleviate the scalability concerns [22, 21]. More recently, Zappala proposed a scheme for alternate path routing in order to avoid bottleneck links [54].

Other Related Work. There are a number of protocols that target environments where there is link-state information [23, 7, 29, 56, 32]. Among these protocols, MOSPF [32] is the most prominent one, and it has become an Internet standard [31]. The protocols in this group need link-state information and this reduces their applicability to single domains running link-state routing protocols, and we will not examine them further. A QoS-aware protocol was proposed that tries to identify a QoS-capable path starting from the receiver and going towards the core of the tree [14]. Despite its promise, the protocol needs to address a number of issues: a) it seems unsuitable for end-to-end metrics, b) the joining time may be long, and c) the communication complexity could be very large for dense networks. Recently, QoS extensions to CBT were proposed [45], but the work focus more on the determining whether the user QoS can be provided and not how we can alter the routing in order to provide the required QoS, which is the focus the work here. In an effort to facilitate the multicast deployment, Keshav and Paul proposed the Centralized Multicast protocol that centralizes control functions and decouples them from data forwarding functions [28]. In a different tangent, Aggarwal and Paul propose an algorithmic framework based on Steiner tree approximations for many-to-many communications [2, 3].

Our work in perspective. It should be clear by now that our work focuses on routing. The big picture of managing multicasting includes complementary aspects such as admission control, resource reservation, and costing or pricing. All aspects are equally important for managing a network effectively, and eventually they should be addressed in a unified framework. We find that addressing the routing aspect is a good starting point towards such a framework.

3 Protocol description

In this section, we describe the QoSMIC protocol. First,we provide an overview of its architecture in section 3.1. Then,

[1] QoSMIC needs only an underlying unicast protocol. We also assume that a packet can collect QoS information from the database of a router along its path.

[2] Source specific trees provide better end-to-end performance in general, but the routing state they require increases linearly to the number of sources. Routing state is considered by many researchers a crucial resource and a lot of research work attempts to reduce the amount of multicast state [10, 51, 44, 38, 2]. Small routing state can allow more multicast groups to be supported in lower end routers with less memory. Moreover, having reduced routing state that fits in the route cache can improve the performance significantly even in higher end routers. Wei and Estrin compare various aspects of performance of shared and source-based trees but not the relative routing state [48, 49].

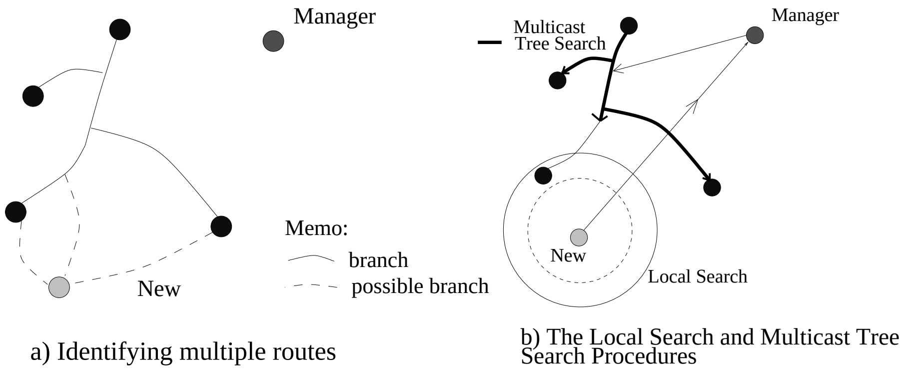


Figure 1: An overview of QoSMIC.

we describe in detail the messages and mechanisms used by QoSMIC in section 3.2. Finally, we outline several implementations issues regarding the deployment of the protocol at the intra-domain and the inter-domain levels in section 3.3.

3.1 Overview

The main innovation of QoSMIC is that it offers routing choices. More precisely, QoSMIC identifies multiple paths and collects QoS routing information for each of them. A new node joining a multicast tree can select a path that suits best its QoS needs according to the collected information. QoSMIC implements a greedy routing heuristic, which attempts to find routers in the tree near the new router as shown in figure 1(a). Another important novelty of QoSMIC is that it does not use a core router. This way, the tree is always "near" the active group members and, as a result, QoSMIC is more efficient than previous single-path core-based protocols; QoSMIC provides better end-to-end performance and accommodates more users.

In a nutshell, QoSMIC constructs a tree in a receiver-driven manner, which involves three phases [22]. In the *Search* phase, it identifies routers on the existing tree that are possible graft points, which we call Candidate routers. In the *Bidding* phase, the Candidates "advertise" themselves to the new router by sending *bid* messages. Note that a *bid* message from the Candidate to the new node collects on its way the current state of the path. In the *Select* phase, the joining router selects the most promising path according to its needs. In QoSMIC, a receiver can switch to a Source-Based Tree to improve its QoS or avoid a congested Shared Tree. The above three-phase approach applies for both Shared and Source-Based Trees.

The Search phase is carried out by two mechanisms: the Local Search and the Multicast Tree Search. The Local Search is identical to the search procedure of YAM; the joining router attempts to "reach" the tree via a limited-scope broadcast in its neighborhood (see figure 1(b)). The Multicast Tree Search is an innovative characteristic of QoSMIC which reduces the control overhead [22]; the in-tree routers run a distributed algorithm to select candidates with lower control overhead.

What does QoSMIC provide? QoSMIC provides QoS-aware and multiple path routing. First, the joining nodes have several paths to choose from. Second, it provides an estimate of the expected QoS performance for each such path. The QoS information depends on the information that the routers on and off the tree can provide. In the extreme case, where there is absolutely no performance information or cooperation from the routers, QoSMIC will resort to an efficient routing protocol based on hop count. Lastly, QoSMIC supports user-driven reconfiguration when QoS becomes unacceptable. Users can disconnect and reconnect to the tree. However, QoSMIC does NOT provide QoS guarantees or globally optimal paths. The greedy algorithm that QoSMIC implements does not guarantee globally optimal results, although it is known to provide close to optimal results both theoretically and in practice [20].

3.2 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 1 lists the different roles of the routers. Table 2 contains a list of the protocol messages.

Every router maintains multicast routing information, which enables it to forward each multicast packet to an appropriate number of links. Each link which is part of a distribution tree has an entry in its table for that tree. If an incoming packet "matches" an entry on a routing table of a link, it is forwarded along that link. A routing entry can be for a Shared Tree, in which case it matches packets from all the sources of that group. Similarly, a routing entry for a Source-Based Tree matches packets only from that source. In the literature, we typically denote an entry for a Source-Based Tree of source S of group G by (S,G) and for a Shared Tree by (*,G), where the symbol "*" represents "all sources". Most IETF multicast protocols have soft state, which means that the routing state expires. For this, adjacent nodes exchange messages periodically, and refresh the state of their common links.

Router Name	Role
Manager router	Supervises the group and facilitates the joining procedure; does not necessarily participate in the data distribution tree.
Candidate router	Considered as possible joining point for a new join request.
Designated router	Connected to a set of users (e.g. hosts on a LAN), receives request from users and initiates searches.
New router	Designated router of a LAN containing the new member.
Destination router	Designated router of a LAN that has active group members.
In-tree router	All routers that are part of the distribution tree.
Intermediate router	In-tree routers that is not a Destination or a Source.
Border router	Router that connects multiple domains.

Table 1: The different roles of the routers.

MESSAGE	Explanation
BID-REQ	New router searches locally for In-tree neighbors.
BID	Candidate identifies itself to the New router; Message collects QoS information along the path.
M-JOIN	New router contacts the Manager who initiates a Multicast Tree Search.
BID-ORDER	The Manager "orders" the In-tree nodes to send bids.
JOIN	The New router sends a message up its selected path to the tree to establish/refresh routing state.
PRUNE	Departing Destination tears down unneeded part of tree; can be for a source-specific or a group tree.

Table 2: An explanation of the messages in QoSMIC.

3.2.1 The Search for Candidates

The process of adding a new branch to the multicast tree starts when the New router receives a request to join a multicast group. If the New router is part of the group already, the connection is established locally. If the New router is not part of the group, the Local Search and the Multicast Tree Search procedures are employed (see figure 1(b)).

Local Search. The New router tries to identify neighboring In-tree routers.

1. The New router "floods" a BID-REQ message in its neighborhood. Reverse path multicasting with scope controlled by the Time To Live (TTL) field. 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 this paper using simulation results.
2. Every In-tree router that receives a BID-REQ message, becomes a Candidate router, and unicast a BID message back 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.

Multicast Tree Search. The New router contacts the Manager and the Manager causes some of the In-tree nodes to propose themselves as Candidate routers.The selection of Candidates is an important aspect of QoSMIC and it is discussed in section 3.2.4. 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 become Candidates.
3. The Candidates unicast BIDs to the New router. The BIDs are identical to the BIDs in the Local Search.

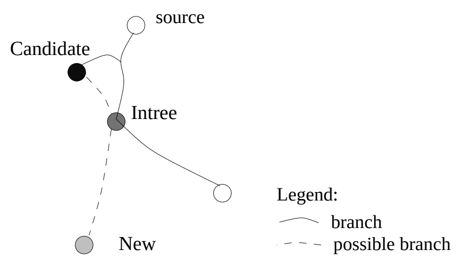


Figure 2: The Take-Over procedure: In-tree routers do not allow the creation of cycles.

Loop-free routing. For both the Local Search and the Multicast Tree Search, if an In-tree 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), and it guarantees that the tree does not have loops.

3.2.2 Connection Establishment

We will examine how the new path is established after the bidding phase has been completed.

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 In-tree 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.

3.2.3 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[3] 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 In-tree router for that tree.

3.2.4 Candidate Selection in the Multicast Tree Search

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

Centralized or Manager Selection. In this mechanism, the manager of the group chooses the candidates. We consider two possible cases where this mechanism may be used. First, if the Manager has global topological information, the candidate selection can be done efficiently with minimal overhead. This can happen if the domain is running a link state protocol, or if the Manager has access to a centralized routing information database [19]. Second, we can make QoSMIC behave like a core-based tree protocol by forcing the tree to be routed at a single chosen candidate. This mechanism would be useful in high-load scenarios, where minimizing the control overhead is more important than finding the most promising path regarding QoS.

This mechanism shows that a) QoSMIC is a flexible routing framework b) the control overhead can be minimized and made comparable to other protocols, if we are willing to sacrifice the quality of the distribution tree.

Distributed Selection. In the absence of centralized information, each In-tree router has to decide whether to become a Candidate in a distributed way. The following orthogonal mechanisms can be used separately or in combination to select the appropriate number of Candidates

1. *Directivity.* The distant routers are 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.
2. *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.
3. *Fractional Choice.* In this mechanism, we can choose as Candidates a representative fraction $(1/n)$ of either all In-tree routers or the ones that meet the other two criteria.

Choice of Mechanisms. The choice of different mechanisms will depend on the network topology and the traffic behavior. Manager Selection offers the lowest control overhead if topology information for the entire domain is available. 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.

3.3 Multiple Managers and Multi-Domain Employment Issues

In this section, we highlight several implementation and inter-operation issues. For a more detailed discussion see [20].

User and protocol QoS interface. The user or the application needs to communicate to the protocol the QoS parameters of interest and the related values and bounds. This is an issue that is not yet resolved or standardized even for unicast QoS routing protocols. It would make sense to develop a common "QoS language". At this point, we envision the QoS communication to be handled by suitable extensions of the Internet Group Management Protocol (IGMP).

Manager Address Distribution. For the Multicast Tree Search, the New router needs to know the address of the Manager and of the group, and both addresses can be made

[3] The membership change can be the result of a request from a host, a PRUNE message from a neighboring router, or an expiration of the soft state if an update is not received for some time.

known through the same mechanism (e.g. Session Directory tool [25]). Alternatively, the address of a local Manager within a domain can be broadcasted via the Domain Wide Reporting [24] or can be provided on demand (by the Border routers or an administrative database).

Manager Change. A Manager change can be done in run time. The change does not cause any data loss or any change in the distribution tree. For a smooth transition, the old Manager can "resign", after the new Manager has been around for a sufficiently long time (depending on the size of the network).

Multi-Domain Multi-Level Operation. QoSMIC can be used at both the intra-domain and the inter-domain level. If the domain already has active group members, the search for Candidate router is carried out within the domain. If not, the connection has an intra-domain and inter-domain part. Inside the domain, the New router contacts one or more Border routers. The Border routers find their best path to the group at the inter-domain level. If the inter-domain protocol is QoSMIC, then the Border routers initiate searches using the procedures of QoSMIC. Each Border router that finds a path to the tree proposes itself to the New router (BID message). The New router chooses the "best" Border router. The selected Border router becomes the Designated Border router for this group. Finally, if one of the levels is running a different protocol, an interface similar to that in [30] will allow inter-operation.

Selecting Manager Routers. In practice, there should be multiple Managers that will cooperate for efficient and scalable solutions with reduced set-up time. In QoSMIC, the following scheme is used: there is at least one Manager per domain, and Border routers are preferred to be the Managers. Hence, the Designated Border router of the group is selected to be the Manager in a domain. In the creation of a session, the Border router nearest to the session initiator becomes the Designated router (and also the Manager) for the group in this domain. For a domain that joins the distribution tree, the selection of the Designated router was described in the previous paragraph. For a Source-Based Tree, the source is selected as the Manager in its own domain to simplify the administration of the tree.

4 Simulations

In this section, we conduct simulations to evaluate QoSMIC. We implement QoSMIC using the Network Simulator (NS) and compare its performance with the existing implementations of PIM and source-based trees (DVMRP). We only show an indicative subset of our results [52] due to space limitations. Table 3 summarizes the main assumptions and values of parameters that we examined in our work.

We have three main goals for our simulations. First, we quantify the difference between source-based trees (SBTs) and shared trees. Second, we compare the shared trees of QoSMIC and PIM. Third, we examine the effects of the Candidate Selection mechanisms for reducing the overhead complexity.

Our simulations lead to the following observations:

- The shared trees of QoSMIC match the end-to-end performance of source-based trees.
- The shared trees of QoSMIC outperform the shared trees of PIM in terms of routing efficiency and end-to-end performance.
- The overhead of QoSMIC is reasonable and controllable.
- Our Candidate-selection mechanisms can reduce the overhead of the search phase significantly. In particular, the directivity mechanism reduces the message complexity by approximately 50% in our scenarios.

In more detail, 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[4].

Source Based Trees (SBTs). We use the existing NS implementation of DVMRP to calculate the minimum-hop source based trees.

PIM-SM. We use the shared tree implementation 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 Path Forwarding routing, such as PIM-SM (Shared Tree only), CBT, and BGMP.

We use real and artificial graphs in our experiments. First, we use an instance of MBone topology with 255 nodes and 266 duplex links measured in 1996. The MBone graph is a real instance of a sparse graph, which provides a very difficult routing environment for a multi-path routing protocol like QoSMIC, since options are absent. Second, we use a number of flat-random Waxman graphs with 100 nodes and different number of links, which are often used to model the inter-domain topology[5] [55, 47]. We model the topology by undirected graphs. In our past work [22] [20], asymmetric links were shown to favor QoSMIC over PIM, with the difference increasing with increasing asymmetry.

For each topology, we repeat the experiment for a total of 100 iterations. For each run of the experiment: one mul-

[4] 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.

[5] Note, that the Waxman topology is relatively richer in terms of alternate paths, while the MBone graph has limited routing choice. Recent measurements of the Internet topology suggest that the network becomes denser and richer in routes [43, 4, 35]. In this case, the experiments on the Waxman graphs may offer useful insights on the actual relative performance of the protocols. In any case, our interest here is not to predict the topology, but more to try to cover a variety of topologies.

Parameter	Values - Assumptions
Protocols	Source Based Trees (SSM), PIM-SM (shared), QoSMIC (shared)
QoSMIC variations	hop metric, delay metric
Performance Metrics	End-to-end delay, routing efficiency, control overhead, set-up latency
Network and Load Scenario	
Graph model	Real (MBone) and Waxman models
Graph average degree	1.04 (MBone), 2.5 to 5.94 (Waxman)
Load Distribution	Uniform, bimodal, truncated heavy tail
Heavy loaded links %	10-90, with a step of 10
Group Member Choice	Uniformly Distributed
Protocol Parameters	
Core router placement	Uniformly Distributed
Source Selection	Uniformly distributed among multicast members
Local Tree Search TTL (hops)	1 to 5
Multicast Tree Search	directivity, local minima, fractional
Directivity threshold (hops)	0, 5

Table 3: Modeling assumptions and parameters in our simulations.

ticast 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. To the best of our knowledge, this selection does not favor any protocol. Note, that the join-only membership behavior has been shown to perform similar to the join-leave behavior [20], hence we do not investigate the join-leave behavior here. Note also, that the models used in [20] have been based on real membership behavior collected by Almeroth and Ammar [6, 5]. For PIM, the core router is randomly selected, which corresponds to its method of core selection based on hashing the multicast address [18, 50] without prior knowledge of the location of the group members.

Set-up latency. The set-up latency of QoSMIC can vary depending on which of the search procedure succeeds. In comparing worst cases for each protocol, we can say that QoSMIC may delay approximately three times more than a core-based single-path protocol such as PIM or CBT [52]. However, the latency of QoSMIC can be better than that of PIM, if the Local Search succeeds and connects to a close neighbor. Due to space limitations, we do not discuss the set-up latency here. For an extended discussion, please refers to [52].

4.1 End-to-end Performance

In this section, we measure the end-to-end delay for each source-receiver pair.

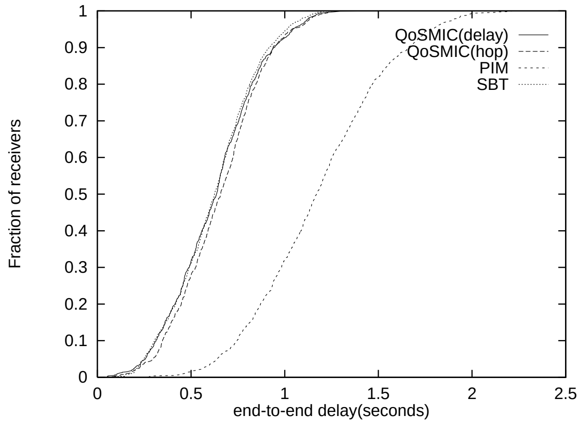
We experiment with various delay distributions: uniform distribution, bi-modal distribution[6], and heavy-tail distribution [33]. The link delay distribution is intended to capture the effect of several sources of delay as congestion, switching delays, etc. Note, that we are mainly interested in comparing 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.

We compare the end-to-end performance of QoSMIC(hop), QoSMIC(delay), PIM and SBTs degree of 4.32 using a uniform link-delay distribution. We plot the fraction of receivers that obtain end-to-end delays lower than or equal to a certain value versus that delay value. In other words, for a given end-to-end delay bound (x-value), the plot shows how many nodes (y-value) would be within that bound.

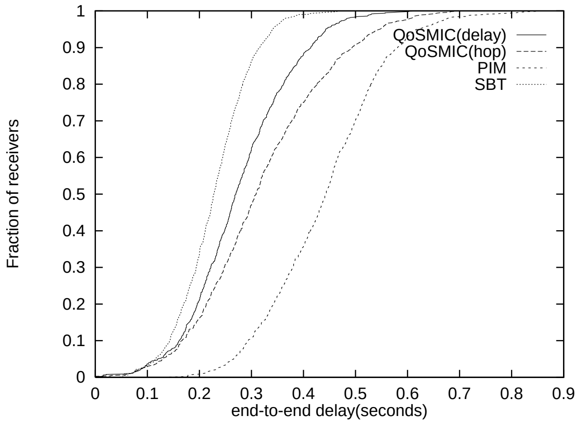
The gain from core-less multicast routing: QoSMIC improves the end-to-end delay by 50%. In figure 3(a) (Mbone topology), we observe that QoSMIC provides end-to-end delays close to the performance of the SBTs, which exhibit approximately half the delay of PIM's shared trees. We remind the reader that 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 SBTs requires more state by a factor of s, the number of sources. However, there is very little difference between QoSMIC(delay) and QoSMIC(hop) due to the smaller number of alternate paths available in the MBone topology. In figure 3(b) we observe that QoSMIC(hop) improves the blocking probability considerably compared to PIM. For example, for a 0.4 second delay bound, the blocking is reduced from 70% blocked receivers (for PIM-SM) to 28% blocked receivers. An alternative way of viewing the same data is that the average delay is reduced by about 30%. In QoSMIC, the multicast tree is "closer" to the position of the receivers.

The gain from QoS-aware multicast routing: QoSMIC admits 3 times more users. QoSMIC can improve things further by routing based on delay. QoSMIC(delay) reduces the blocking probability further for the same delay

[6] 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 distributed in the range of [0ms,10ms].

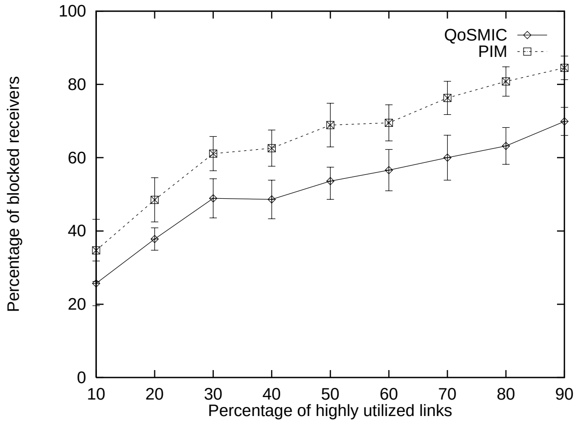


(a) End-to-end delay: MBone graph

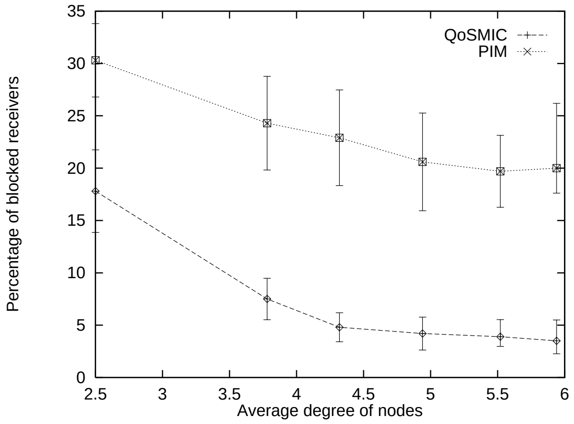


(b) End-to-end delay: Waxman flat graph (deg. 4.32).

Figure 3: Fraction of nodes below an end-to-end performance value using the uniform delay distributions.



(a) Blocked Receivers on the MBone graph



(b) Blocked Receivers on the Waxman graph

Figure 4: A: Percentage of blocked receivers vs. percentage of highly loaded links using the bi-modal load distribution. B: Percentage of blocked receivers as a function of average degree of nodes with 20% highly loaded links: Waxman flat Euclidean graph.

bound compared to QoSMIC(hop) and PIM. For example, for 0.4 seconds delay bound, the blocking probability is reduced to 20%. This represents a 2.7 fold increase in the number of satisfied participants over PIM-SM.

The effect of the network load: QoSMIC provides better end-to-end delay for variable network load. In figure 5(a), we observe that, in a lightly utilized network with few high delay links, the difference in the average delay between QoSMIC(delay) and PIM-SM is approximately 50%. The difference increases as more links exhibit high delay until half of the links exhibit high-delay, where the difference in the end-to-end delay becomes approximately 90%! It appears interesting that 20% of high delay links is a sweet spot for QoSMIC, since that matches the empirical data available for load distributions on the Internet [34].

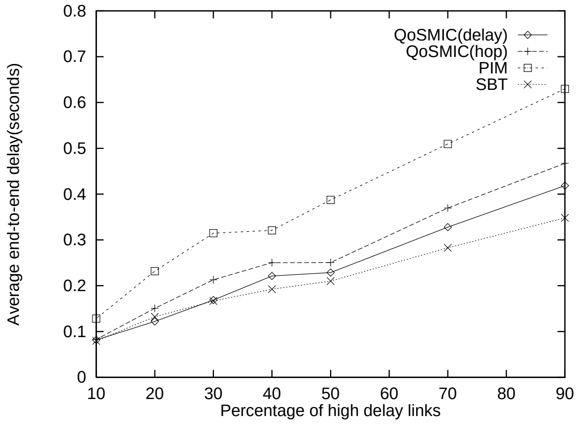
The effect of the network density: QoSMIC can take advantage of the richness of the topologies better than PIM. In figure 5(b), we observe that regardless of the topology, QoSMIC always does better in the end-to-end performance as compared to the other shared tree protocols. Note, that QoSMIC outperforms SBTs in some cases, since SBTs are based on QoS-unaware minimum-hop routing.

4.2 Routing Efficiency

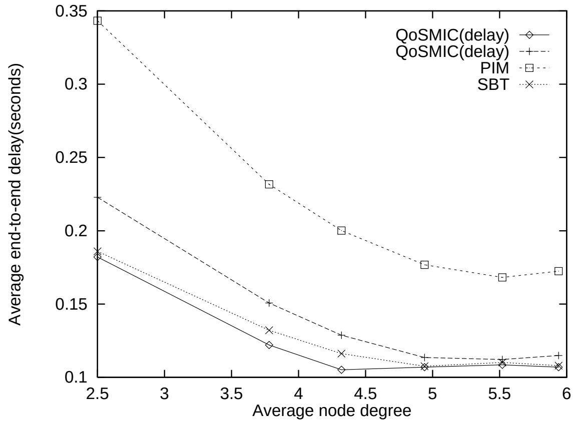
In this section, we evaluate the routing efficiency of QoSMIC and PIM-SM. Note that we focus on Shared Trees, since the comparison of Source Based Trees is less interesting. We measure the percentage of receivers prevented from joining a multicast session due to insufficient network resources. This is a good metric of the routing efficiency a protocol.

We assign a bandwidth utilization factor to each link to represent the effect of background network load. We then simulate a single session per experiment instance and measure the number of calls admitted under the current network state. We experiment with several link-load distribution functions, such as uniform distribution and bimodal distributions. Due to space limitations, we focus on the bi-modal distributions.

QoSMIC is resource efficient. In figure 4(a), we plot



(a) Delay vs. percentage of high delay links



(b) Delay vs. average node degree.

Figure 5: Average end-to-end delay in Waxman flat Euclidean graphs. Graph (a) shows the average delay as a function of the percentage of high delay links using bi-modal delay distributions keeping the node degree fixed at 3.78. Graph (b) shows average delay as a function of the average node degree keeping the distribution fixed at bi-modal(20).

the percentage of blocked receivers as a function of the percentage of highly utilized links for the MBone graph. We observe that QoSMIC exhibits lower blocking probability compared to PIM-SM.

QoSMIC: the higher the connectivity, the greater the gain. We examine the relative performance of the protocols as the connectivity of the graphs changes. In figure 4.(b), we observe that QoSMIC has make better use of a denser topology. The error bars represent the 95% confidence intervals. We also notice that the blocking probability for QoSMIC reduces faster than that of PIM.

4.3 Overhead Complexity

In this section, we examine the overhead complexity of QoSMIC. We conclude that the search-phase overhead of QoSMIC is a small fraction of the total control overhead of a multicast group. We also evaluate the different search mechanisms we proposed earlier and provide insight on their performance. Intuitively, we can distinguish two steps: a) QoSMIC introduces the Multicast Tree Search to reduce the overhead of the Local Search, and then, b) we introduce mechanisms to control the overhead of the Multicast Tree Search.

First, we examine the Candidate search procedures and study the advantage of having two such procedures. In our experiments, the Local Search is used before turning on the Multicast Tree Search procedure. The Multicast Tree Search is invoked only if the Local Search falls short. In addition, we vary the TTL value during the local search procedure to limit the scope of the Local 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 Section 3.2.4) to reduce the number of Candidates and the number of BID messages.

We also use the full blast Multicast Tree Search procedure[7] without turning on any of the specific Candidate selection mechanism mentioned. This will give us an estimate of the worst case overhead resulted from the Multicast Tree Search procedure.

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.

4.3.1 Full Blast Multicast Tree Search

We show the effect of TTL values on the overhead complexity of the Local Search procedure. We did not compute the confidence interval in these figures, because we are interested in the relative effect between different TTL values.

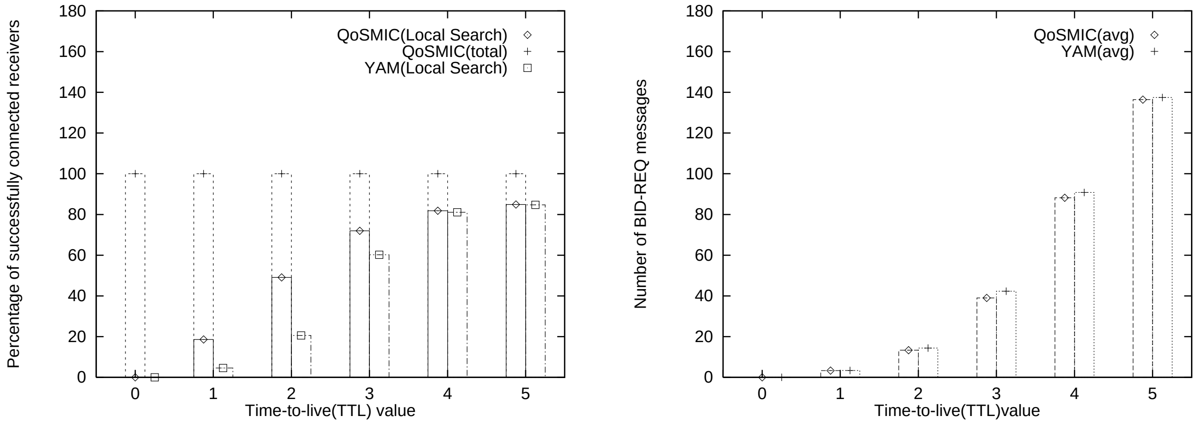
In figure 6(a), we observe that the success ratio of YAM depends solely on the TTL value, while QoSMIC can obtain 100% success with the help of multicast tree search. However, as the TTL value gradually increases, YAM succeeds in certain percentage of connections[8], and it achieves almost 90% of the connections when the TTL value is 5.

In figure 6(b), we observe the drastic increase in the number of BID-REQ message as the value of TTL increases.

Thus, we want to limit the scope and communication overhead of the local search procedure. But, we also want to use the Local Search procedure to connect the many "short" joins efficiently and quickly. The scope of the Local Search depends on the network size and topology. In our experiments, we observe that QoSMIC can successfully connect 50% of all receivers using a Local Search with TTL=2. Therefore, a Local Search of two hops is a rea-

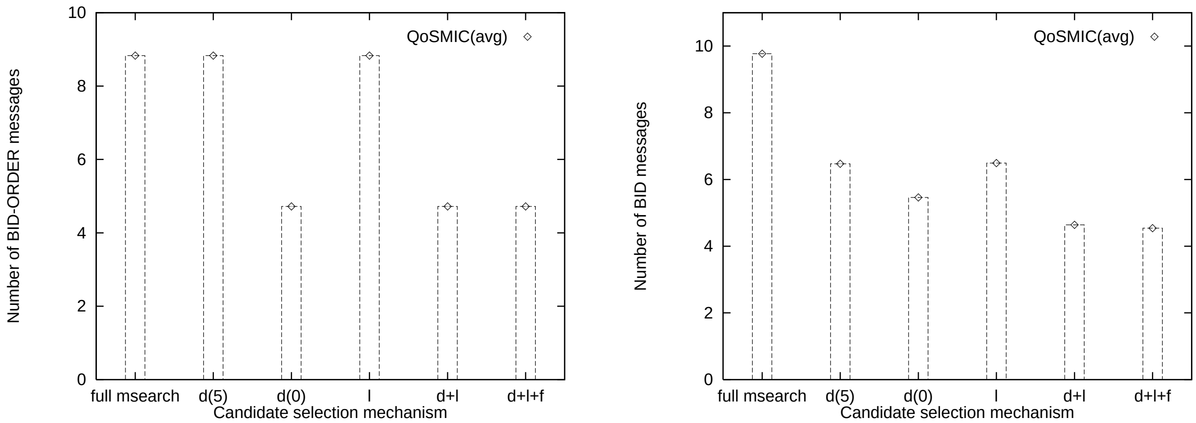
[7]In the full blast Multicast Tree Search procedure, the BID-ORDER message is sent to every node in the tree and every In-tree node sends a BID message to the new router.

[8]Note, that the Local Search of QoSMIC has a higher success rate. The two Local Search procedures are identical, but as YAM fails to connect some nodes, the tree is "smaller" compared to the tree of QoSMIC, whose Multicast Tree Search will connect all nodes. Therefore, subsequent nodes fail to connect to the smaller tree of YAM.



(a) Percentage of connected receivers. (b) Number of BID-REQ messages.

Figure 6: The message overhead using full blast Multicast Tree Search as a function of TTL value: Waxman flat Euclidean graph (node degree 3.4).



(a) Number of BID-ORDER messages. (b) Number of BID messages.

Figure 7: Number of BID-ORDER messages and Number of BID messages resulted using different Candidate selection mechanism: Waxman flat Euclidean graph (node degree 3.4). (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.)

sonable choice for QoSMIC, for the overhead of the Local Search is small, and at the same time, we can take advantage of many "short" joins.

In QoSMIC, we can limit the message complexity of the Local Search arbitrarily. This advantage comes from the Multicast Tree Search procedure that will provide a path even if the Local Search falls short. In the next series of simulations, we show how we can further reduce the total message complexity by minimizing the overhead from Multicast Tree Search.

4.3.2 Multicast Tree Search With Various Selection Mechanisms

We examine the reduction of the control overhead by employing the Candidate Selection mechanisms discussed in section 3.2.1. In particular, we want to limit the number of BID-ORDER messages (because of the unconstrained searching of the multicast tree) and the number of BID messages (because of the large number of candidates selected).

Our simulations suggest that the Candidate-selection mechanisms can reduce the overhead significantly.

- The directivity mechanism can reduce the number of BID-ORDER and BID messages by approximately 50%.
- The local minima mechanism can reduce the number of BID messages by approximately 40%.

In the experiment, we turn on each Candidate selection mechanism: directivity, local minima, and fractional choice separately, or use a combination of them. We use a $TTL = 2$ for the scope of the Local Search procedure.

In figure 7, 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 observe that 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, where to a point, fractional choice has no further effect.

The overhead of QoSMIC in perspective: *The search-phase overhead of QoSMIC is a small fraction of the total control overhead of a multicast group.* To put things in perspective, we compare the control overhead of a new join to the overhead incurred over the life of the multicast group. First of all, both QoSMIC and PIM-SM and most other protocols[9] use a periodic refresh mechanism. QoSMIC has a 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. For long lived sessions, this additional one-time overhead becomes negligible.

5 Conclusions

In this paper, we propose and evaluate QoSMIC, a QoS-aware multicast routing protocol for the Internet. QoSMIC is a promising protocol offering several advantages both in terms of architectural design and in terms of improved performance and efficiency. QoSMIC is a scalable and efficient multicast protocol that can operate both at the intra-domain and the inter-domain level. The fundamental innovation of QoSMIC is that it offers QoS-aware receiver-driven routing; it provides routing choice and up-to-date information, and the user/application can tailor the routing decisions to their QoS needs.

The main innovations and features of QoSMIC are listed below.

1. *QoS support.* QoSMIC implement QoS-aware routing by providing multiple paths and QoS-information. This way, the new router joins to the multicast tree with the path that seems able to provide the best QoS performance. QoSMIC supports user-driven reconfiguration when QoS becomes unacceptable; users can disconnect and reconnect to the tree hoping that the new set of paths will provide better solutions.
2. *Flexibility.* One of the main advantages of QoSMIC is its flexibility in handling applications and users. Applications are able to specify their requirements and create a tree according to their needs. Furthermore, within each group, receivers can request and obtain the resources that appeal to their needs. Finally, QoSMIC can create source-based trees (like PIM but unlike CBT), which is another aspect of flexibility.
3. *Dynamic path information.* Control messages of the protocol collect dynamic routing information on each path and therefore, QoSMIC needs only interact with the local routing database at each router.
4. *Adaptive core-less routing.* QoSMIC constructs a tree in a dynamic way that adapts to the location of the group members, instead of pinning the tree at a predetermined core.
5. *Scalability.* The protocol can scale to large networks and receivers. The two procedures for the route identification overcome the problems that, YAM, the predecessor of QoSMIC, had.

[9] The MIP protocol proposes the use of hard-state [36], which incurs less overhead but is more vulnerable to maintaining obsolete state.

We conducted simulations to study the performance of QoSMIC. Our results can be summarized in the following observations.

Improved end-to-end performance. QoSMIC can reduce the end-to-end delay by up to 90% in high loads. Alternatively, this translates into a reduced call blocking for applications with strict QoS requirements. For some cases, the number of satisfied participants with QoSMIC is increased approximately by a factor of three as compared to PIM.

Improved resource utilization. QoSMIC uses resources more efficiently compared to other protocols such as PIM. The efficient use of network resource translates into a reduced call blocking probability. Furthermore, the advantage of QoSMIC increases as graphs become denser.

Controllable and reasonable overhead. First, we saw that the overhead of QoSMIC is a small fraction of the total overhead of a multicast group. Second, QoSMIC can be tuned to strike the balance between a high-overhead routing-optimization mode to a low-overhead basic routing mode. This tuning can take place in run time according to the network conditions within the lifetime of a multicast group.

Future work. The QoSMIC protocol could be seen as a collection of routing procedures that could be of independent interest. For example, the Local and Multicast Tree Search can be used by other multicast protocols. Finally, we have already developed a publicly available implementation of QoSMIC for the NS-2 simulator [1], and we intend to integrate it eventually.

References

[1] The QoSMIC Implementation web-Site. www.cs.ucr.edu/~michalis/qosmic.html.

[2] S. Aggrawal and S. Paul. A flexible protocol architecture for multi-party conferencing. *Proceedings of the fifth International Conference on Computer Communications and Networking*, 1996.

[3] S. Aggrawal and S. Paul. A flexible multicast routing protocol for group communication. *Computer Networks*, 32(1), 2000.

[4] R. Albert, H. Jeong, and A. Barabasi. Attack and error tolerance of complex networks. *Nature*, 406, July 2000.

[5] K. Almeroth. A long-term analysis of growth and usage patterns in the multicast backbone (mbone. *Proc. IEEE INFOCOM*, Tel Aviv, Israel 2000.

[6] K. Almeroth and M. Ammar. Collection and modeling of the join/leave behavior of multicast group members in the mbone. *IEEE Communications*, June 1997.

[7] S. Angelopoulos. Efficient online algorithms for multicasting with bandwidth and delay guarantees. Master's thesis, University of Toronto, Dept of. Computer Science, 1999.

[8] A. Ballardie. Core Based Trees (CBT: An architecture for scalable inter-domain multicast routing. *ACM SIGCOMM*, 1993.

[9] A. Ballardie. Core Based Trees (CBT version 2) multicast routing. IETF RFC 2201, 1997.

[10] T. Billhartz, J. B. Cain, E. Farey-Goudreau, D. Fieg, and S. G. Batsell. Performance and resource cost comparisons for CBT and PIM multicast routing protocols. *IEEE Journal of Selected Areas in Communications*, 15(3):304–315, April 1997.

[11] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. Internet RFC 2475, December 1998.

[12] K. Carlberg and J. Crowcroft. Building shared trees using a one-to-many joining mechanism. *ACM Computer Communication Review*, pages 5–11, January 1997.

[13] S. Casner. Major MBONE routers and links. Available from ftp.isi.edu:mbone/mbone-topology.ps, 1994.

[14] S. Chen, K. Nahrstedt, and Y. Shavitt. A QoS-aware multicast routing protocol. *Proc. IEEE INFOCOM*, 2000.

[15] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. *ACM SIGCOMM*, pages 14–26, August 1992.

[16] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A framework for QoS-based routing in the Internet. IETF RFC 2386, 1998.

[17] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, F. Sharma, and L. Wei. The PIM architecture for wide area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2):153–161, April 1997.

[18] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, F. Sharma, and L. Wei. Protocol independent multicast-sparse mode (PIM-SM): Protocol specification. Internet Draft RFC 2362, 1998.

[19] D. Estrin, J. Postel, and Y. Rekhter. Routing arbiter architecture. *ConneXions*, 8(8), 1994.

[20] M. Faloutsos. *The Greedy, the Naive, and the Optimal Multicast Routing: from Theory to Internet Protocols.* PhD thesis, University of Toronto, Dept of. Computer Science, 1998.

[21] M. Faloutsos, A. Banerjea, and R. Pankaj. Designing QoSMIC: a QoS Multicast Internet protoCol. Technical Report CSRI 372, University of Toronto, Technical Report CSRI, 1998.

[22] M. Faloutsos, A. Banerjea, and R. Pankaj. QoSMIC: a QoS Multicast Internet protoCol. *ACM SIGCOMM*, Sep 2-4, Vancouver BC 1998.

[23] A. Fei and M. Gerla. Receiver-initiated multicasting with multiple QoS constraints. *Proc. IEEE INFOCOM*, 2000.

[24] W. Fenner. Domain wide multicast group membership reports. Distributed in the IDMR mailing list. To appear as a Working Draft, 1997.

[25] M. Handley and V. Jacobson. SDP: Session description protocol. Internet Draft. Work in progress., 1995.

[26] H. Holbrook and B. Cain. Source-specific multicast for ip. Internet-Draft: draft-holbrook-ssm-00.txt available from ftp://ftp.ietf.org/internet-drafts/, 2000.

[27] H. Holbrook and D. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. *ACM SIGCOMM*, pages 65–78, Sep 1-3, Cambridge MA, 1999.

[28] S. Keshav and S. Paul. Centralised multicasting. *International Conference on Network Protocols*, 1999.

[29] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transaction on Networking*, pages 286–292, June 1993.

[30] S. Kumar, P. Radoslavov, D. Thaler, C. Alletinoglu, , D. Estrin, and M. Handley. The MASC/BGMP architecture for interdomain multicast routing. *ACM SIGCOMM*, pages 93–104, Sep 2-4, Vancouver BC 1998.

[31] J. Moy. Multicast extensions to OSPF. IETF RFC 1584, 1994.

[32] J. Moy. Multicast routing extensions for OSPF. *ACM Connunications*, 37(8):61–66, 1994.

[33] C. L. Nikias and M. Shao. *Signal Processing with Alpha-Stable Distributions and Applications.* John Wiley and Sons, Inc., 1995.

[34] A. M. Odlyzko. Data networks are mostly empty and for good reason. *IT Professional*, pages 2, 67–69, 1999. Available at http://www.research.att.com/ amo.

[35] C. Palmer, G. Siganos, M. Faloutsos, C. Faloutsos, and P. Gibbons. The connectivity and fault-tolerance of the Internet topology. *Workshop on Network-Related Data Management (NRDM 2001), In cooperation with ACM SIGMOD/PODS, Santa Barbara*, 2001.

[36] M. Parsa and J. J. Garcia-Luna-Aceves. A protocol for scalable loop-free multicast routing. *IEEE Journal of Selected Areas in Communications*, 15(13):316–331, April 1997.

[37] R. Perlman, C-Y Lee, A. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, and M. Green. Simple Multicast: A design for simple, low-overhead multicast. Internet-Draft: draft-perlman-simple-multicast-02.txt available from ftp://ftp.ietf.org/internet-drafts/, 1999.

[38] P. Radoslavov, D. Estrin, and R. Govindan. Exploiting the bandwidth-memory tradeoff in multicast state aggregation, 1999.

[39] H. F. Salama, D. S. Reeves, and Y. Viniotis. Evaluation of multicast routing algorithms for real-time communication on high-speed networks. *IEEE Journal of Selected Areas in Communications*, 15(13):332–345, April 1997.

[40] C. Sheilds and J. J. Garcia-Luna-Aceves. The Ordered Core Based Tree protocol. *Proc. IEEE INFOCOM*, Kobe, Japan, April 7-11 1997.

[41] C. Sheilds and J. J. Garcia-Luna-Aceves. The HIP protocol for hierachical multicast routing. *Proc. ACM Symposium on Principles of Distributed Computing(PODC)*, Puerto Vallarta, Mexico, June 28-July 2 1998.

[42] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

[43] S.L. Tauro and M. Faloutsos. Fault-tolerance and robustness of the internet topology. *PRDC 2000, (abstract), Los Angeles*, 2000.

[44] David Thaler and Mark Handley. On the aggregatability of multicast forwarding state. *Proc. IEEE INFOCOM*, Tel-Aviv 2000.

[45] H.-Y. Tyan, J. Hou, and B. Wang. QoS-enhanced core-based multicast routing. *IEEE 20th Real-Time System Symposium (RTSS'99)*, December 1999.

[46] S. Verma, R. K. Pankaj, and A. Leon-Garcia. Performance of QoS based multicast routing algorithms for real-time communications. *Proc. of SPIE symposium on voice, video and data communications*, Nov 1997.

[47] B. M. Waxman. Routing of multipoint connections. *IEEE Journal of Selected Areas in Communications*, pages 1617–1622, 1988.

[48] L. Wei and D. Estrin. The trade-offs of multicast trees and algorithms. *International Conference on Computer Communications and Networks*, 1994.

[49] L. Wei and D. Estrin. Multicast routing in dense and sparse modes: Simulation study of tradeoffs and dynamics. *ICCCN*, pages 150–157, 1995.

[50] B. Williamson. *Developing IP Mulitcast Networks.* CISCO Press, 2000.

[51] T. Wong and R. Katz. An analysis of multicast forwarding state scalability. *International Conference on Network Protocols*, 2000.

[52] S. Yan. Implementation and evaluation of qosmic - a new internet multicast routing protocol. Master's thesis, University of Toronto, Dept of. Electrical Engineering, 1999.

[53] S. Yan, M. Faloutsos, and A. Banerjea. QoS-sensitive routing for real-time multicasting. *Real-Time Internet Applications Workshop (RTAW'99)*, pages 36–42, Vancouver, June 1, 1999.

[54] D. Zappala. Alternate path routing for multicast. *Proc. IEEE INFOCOM*, Tel-Aviv, Israel 2000.

[55] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for internetworks. *IEEE/ACM Transactions on Networking*, 5(6):770–783, December 1997. http://www.cc.gatech.edu/projects/gtitm/.

[56] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves. A source-based algorithms for delay constrained minimum cost multicasting. *Proc. IEEE INFOCOM*, pages 377–385, 1995.

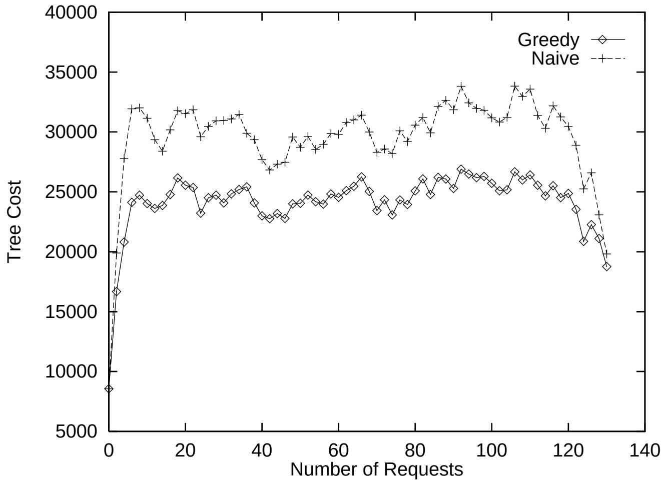


Figure 8: Tree Cost versus number of requests in the MBone map for the join-leave problem with group-size 8 and asymmetry 10.

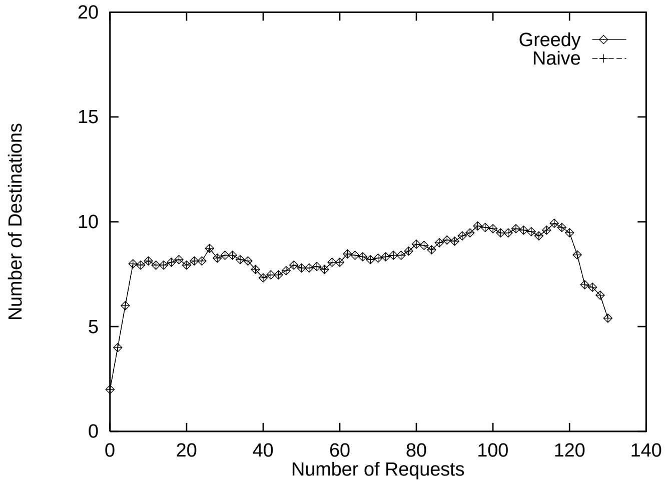


Figure 10: Actual Group-Size versus number of requests on the MBone map for the join-leave problem with group-size 8 and with asymmetry 10.

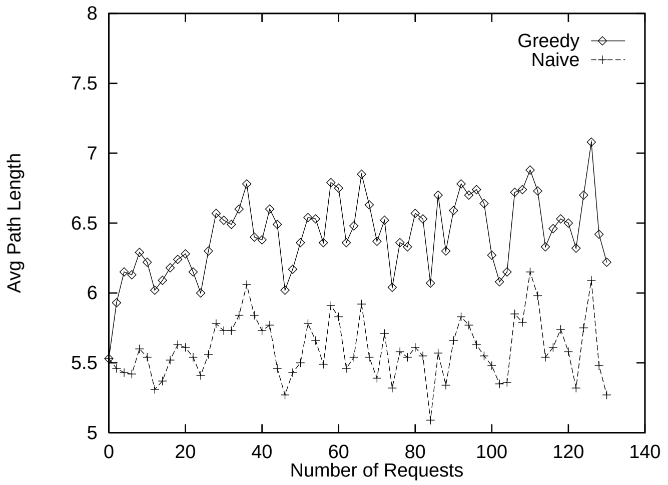


Figure 9: Path length versus number of requests on the MBone map for the join-leave problem with group-size 8 and with asymmetry 10.

A The Effect of Join-Leave Membership

In our work, we focused on the join-only membership behavior. Here, we show that the join-leave behavior does not affect the reported results. More specifically, extensive simulations show that [20] the routing efficiency of QoSMIC over a protocol like PIM is likely to exist even if nodes join and leave.

We simulate and compare the two routing approaches that the routing for the shared-tree that QoSMIC and PIM (along with the majority of other protocols) implement: Greedy and Naive or Shortest Paths The Greedy algorithm is based on an efficient heuristic for the Steiner tree problem [42]. The main idea is that at we connect each destination with the minimum cost path to the existing multicast tree. The Naive algorithm takes the union of the shortest paths between the source and each destination. Observe that the tree that Naive creates is independent of the order with which nodes join or leave [20]. Therefore, we do not expect the tree quality of Naive to degrade as nodes join and leave.

Membership Scenarios. We consider two scenarios of dynamic behavior. Also note that we consider a single source.

- In the *join-only case*, destination appear one by one in a random order and stay until the end of the session.
- For the *join-leave case*, we have three phases: a) build-up, b) fluctuations, and c) tear-down of the session. This behavior was observed in real multicasts over the Internet by Almeroth and Ammar [6]. The build-up is like a join-only session. In our simulations, once the group reaches the specified size, we add and remove nodes randomly: first, we select randomly with equal probability between a join and a leave, and then we select an eligible node to associate with the request. The number of fluctuations is set approximately three times the size of the graph, i.e., every node in the graph joins and leaves at least twice on the average. In the tear-down of the session, we remove randomly the active destinations one by one.

Performance Metrics. We monitor the performance of the algorithms with the use of the following metrics. The *tree cost* is the sum of the routing costs of all its edges, which quantifies the routing efficiency as we have defined earlier. The *end-to-end average path length* is the number of edges of the path along the tree from the source to a destination. We take the average end-to-end length over all destinations of the tree.

We use the term "relative performance" to denote how the performances of Naive and Greedy compare. For the tree cost, we consider the ratio of the cost of the tree of Naive over that of the Greedy: T_{Naive}/T_{Greedy}. For the average path length, we refer to the absolute difference in edges: $P_{Greedy} - P_{Naive}$, where P_x is the average path length of the tree of algorithm x.

The Graph Models. We use two types of graphs: a) a real MBone instance, and b) artificial topologies based on the Waxman model. Casner provides an actual topology of the multicast-capable Internet routers in 1994 [13]. We experimented with a large number of Waxman graphs that are not shown here. The cost of the edges are uniformly distributed.

The Effect of Join-Leave. The relative performance of Greedy does not degrade as the number of requests increases. In addition, the difference in the average path length of the two algorithms does not seem to increase over time either. We plot the tree cost and end-to-end path length versus the number of requests over time in figures 8, 9 respectively. Greedy creates less expensive trees for a small increase in the average path length. We observe that the cost gain of the Greedy can be as much as 25%, while the average path length difference is never more than one. In figure 10, we plot the number of destinations over time.

Given that the tree of Naive is oblivious to the dynamic membership, we can conclude the following:

- The performance of the Greedy does not seem sensitive to the join-leave group behavior in practice.
- We observe that the routing efficiency of Greedy is consistantly better than that of Naive.

We obtained similar results in all experiments with alarge number of scenarios with different topologies, membership behavior and edge-cost distribution [20].