QoS-sensitive routing for Real-time Multicasting

Shuqian Yan
University of Toronto

Michalis Faloutsos
UC, Riverside

Anindo Banerjea*
ISI/USC

Abstract

Multicast service with Quality of Service (QoS) guarantees is critical to commercial and real-time applications, such as telemedicine, video-conferencing, and virtual reality. The Internet is currently in the process of developing QoS architectures (e.g., Int-serv, Diff-serv) to support such applications. However, most multicast protocols, such as PIM and CBT, do not employ QoS-sensitive *routing: they do not consider the QoS requirements of the user in computing the route. In this paper, we discuss the importance of QoS-sensitive routing for the support of real-time multicast applications. 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. In addition, existing work shows that the QoS-sensitive routing for multicast can be implemented stably, efficiently and scalably. Therefore, we conclude that QoS-sensitive multicast routing should be considered in commercial networks.*

1 Introduction

The motivation for this paper 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 the 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. In this paper, we focus on real-time multicast applications, and ask the question: what are the costs and benefits of using the "traffic-aware" routing approach?

Multicasting is an essential building block of future networks. Only with multicasting can we support one-to-many and many-to-many applications, such as t.v. broadcasting, teleconferencing, and telemedicine efficiently. Such applications will become commercial commodities in integrated services networks in the future. At the same time, these applications have the potential to saturate the network resources. The need for scalable and efficient multicast protocols has triggered much research activity in the Internet community.

Real-time applications have stringent Quality of Service (QoS) requirements. These applications demand bounded end-to-end delay [14] [12], which is defined as the time it takes a packet to go from the source to the destination. Other, closely related, metrics are the end-to-end path-length, the packet loss ratio and the required bandwidth of a connection [16]. In this paper, we consider that users want to join a multicast session with some end-to-end delay bound. If the delay bound is not met, that particular user is blocked.

Multicast routing protocols can be classified into two categories with respect to QoS. If a protocol considers the QoS requirement of the application in its routing phase, we call it *QoS-sensitive*. Internet has been traditionally best effort in nature. Most current multicast Internet protocols, such as PIM, CBT and BGMP, are *non-QoS-sensitive*. They provide a single path per user, and if the path does not have enough resources, the connections fails. Note that QoS architectures like Int-serv [8] and Diff-serv [5] can not help in such a case, since they can only manage resources along the single path provided by the non-QoS-sensitive routing protocol. In contrast, a *QoS-sensitive* routing protocol would attempt to find a QoS-capable path. QoSMIC is currently the only QoS-sensitive *multicast* routing protocol that has appeared as an IETF draft [3].

The Internet community seems sharply divided on the issue of the need for QoS-sensitive routing. To some, the need for QoS-sensitive routing is self-evident. However, a significant section of the community doubts the importance of QoS-sensitive routing. Some of the objections are technical: a network with QoS-sensitive may be unstable; or it may not scale well enough to be applicable to the Internet. The technical objections can be countered by good design and proofs of properties [3].

*Contact author: banerjea@isi.edu, USC Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292-6695. Phone: (310) 822-1511 Fax: (310) 823-6714

However, the most fundamental objections relate to the very need for QoS-sensitive routing. In fact, the most common question that we were asked after presenting our initial work [11] was, how many more multicast sessions can you support with QoSMIC than with a simpler protocol?

This is the question that we set out to answer in this paper: How much more efficient is QoS-sensitive routing as compared to *non-QoS-sensitive* routing for real-time multicast? The answer, of course, is that it depends, mainly on the delay requirements of the application. However, we show that the difference can be substantial. Almost three times as many users can be supported for some delay requirements, and the difference is always positive. Our simulations compare the PIM and QoSMIC protocols, but the the study has more general interest: it is a comparison between two different routing approaches.

The rest of this paper is structured as follows. In section 2, we discuss previous work. We also present the QoSMIC protocol, which shows that QoS-sensitive routing can be achieved in a *scalable, and efficient manner*. In section 3, we show detailed simulations that quantify the gain in end-to-end QoS using QoS-sensitive routing. We conclude that QoS-sensitive routing protocols will be vital for commercial networks supporting real-time multicast applications.

2 Background and Related Work

In this section, we present previous work in multicast routing on the Internet. We discuss how QoS-sensitive routing is feasible through the QoSMIC protocol.

The Internet was not designed to support QoS applications; it is best-effort in nature. The routing protocols consider hop-count as the only metric in constructing routing tables. All multicast protocols currently under development within the IETF are non-QoS-sensitive. Protocols such as PIM [9], CBT [2] [1], BGMP [15], and SM [13], do not consider QoS requirements in their routing. These protocols construct multicast trees using shortest path trees rooted at a single node, called the *core* or rendez-vous point. The multicast trees are *shared*: packets from all the sources use the same bidirectional link. Source specific trees (as created by protocols such as DVMRP, or available as an option in PIM) provide much better end-to-end performance, but they do not scale well, and are not considered a viable solution for multicasting at large [4].

The previous protocols were not designed to support real-time applications. Their end-to-end performance suffers from two issues: b) their performance depends on an initial core selection, and a) their routing uses hop-count as the only metric. First, core-based trees are *statically* constrained by the selection of the core. Thus, their performance is heavily dependent on the selection of the core, and does not respond well to changes in group locality [11] [10]. Physically close routers may communicate through a distant core, leading to an unnecessary increase in the end-to-end delay. Second, the protocols provide a single path for each participant, based on hop-count as the only criterion. If this single path does not meet the QoS requirement, that participant is blocked.

2.1 QoS-sensitive protocols

QoS-sensitive routing relies on the ability to choose between multiple available paths based on QoS criteria. In the context of multicast routing in the Internet, the Yet Another Multicast (YAM) protocol was the first multiple-path protocol [6], but the scalability of its control messages was a possible weakness. More recently, the QoSMIC protocol [11] [3] appears as a superset of YAM; it has a number of additional mechanisms that enhance its scalability and its adaptability, but it constructs similar multicast trees. For this reason, we focus on QoSMIC in this paper. In brief, QoSMIC identifies multiple paths for each new member of the group, which chooses the path that suits its QoS requirements best. For each path, the control messages gather up-to-date routing information. Recall the driving analogy: QoSMIC provides both the option and the knowledge to make a good routing choice.

In addition, QoSMIC does not rely on a core router; the tree follows dynamically the evolution of the group. As a result, the end-to-end path-lengths are shorter than for core-based tree, resulting in smaller end-to-end delays. Furthermore, QoSMIC is a flexible protocol that can adapt to the needs of different applications and users in a receiver-driven way (for example, by using different path selection criteria for different receivers).

The above advantages of QoSMIC have no serious disadvantages to outweigh them. The protocol causes a reasonable and controllable increase in the number of control messages [11]. Because the route computation is done at JOIN time, and not for every packet, it has the same stability properties as PIM or CBT. More importantly, QoSMIC does not cause any increase in the amount of state in the network; the state requirements of the routing table of QoSMIC are *identical* to PIM or CBT. Further details can be found in Appendix A and [11].

Preliminary simulations have shown that QoSMIC and YAM create more efficient trees compared to the core-based protocols, and QoSMIC exhibits better scaling behavior than YAM [11] [10].

3 Simulations

In this section, we quantify the benefit of doing QoS-sensitive routing through detailed simulations. We use the Network Simulator (NS) version 2.1 b5 by UCB/LBNL/VINT. We implement the QoSMIC protocol in NS to compare its performance with the existing implementations of PIM and DVMRP. We focus on the tree setup portion of the protocol, since the data forwarding mechanisms are identical for all the protocols compared. As we saw, QoSMIC can attempt to optimize user-defined criteria. For our experiments, we look at the effect of using two different criteria, hop-count and end-to-end delay[1]. 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 the sources, and then we select the path with the minimum such delay[2].

PIM. We use the simple shared tree implementation of PIM that is currently available in NS.

DVMRP. We use the existing NS implementation of DVMRP. The protocol creates *source specific* min-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.

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 for real-time applications.

We use real and artificial graphs in our experiments. First, we use a partial Mbone topology with 255 nodes and 266 duplex links available in the NS package. Second, we use flat-random (Waxman) graphs, which are often used to model the inter-domain topology [17] [18]. In order to see the effect of changing the node degree on the difference between QoSMIC(delay) and QoSMIC(hop), we use two Waxman topologies with node degrees of 3.78 (Flat 1) and 4.32 (Flat 2) respectively.

The delay of each link is randomly chosen. We experimented with various delay distributions, such as exponential and uniform with various parameters, as well as combinations. 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 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 observed that, for the distributions that we examined, the choice of the parameters had no significant impact on the relative performance of the protocols. Hence, we do not investigate the impact of the delay distribution on the performance further in this paper. The graphs presented are based on a uniform delay distribution from [10,110] ms.

Thus, we present three experiments in this paper, corresponding to the three topologies: Mbone, Flat 1, and Flat 2. 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 random model is as good as any, and does not favour any protocol. Note that the join-only membership behavior has been shown to perform similarly to the join-leave behavior [10], hence we do not investigate the join-leave behaviour here. For PIM, its core router is also randomly selected, which corresponds closely to its method of core selection based on hashing the multicast address [9].

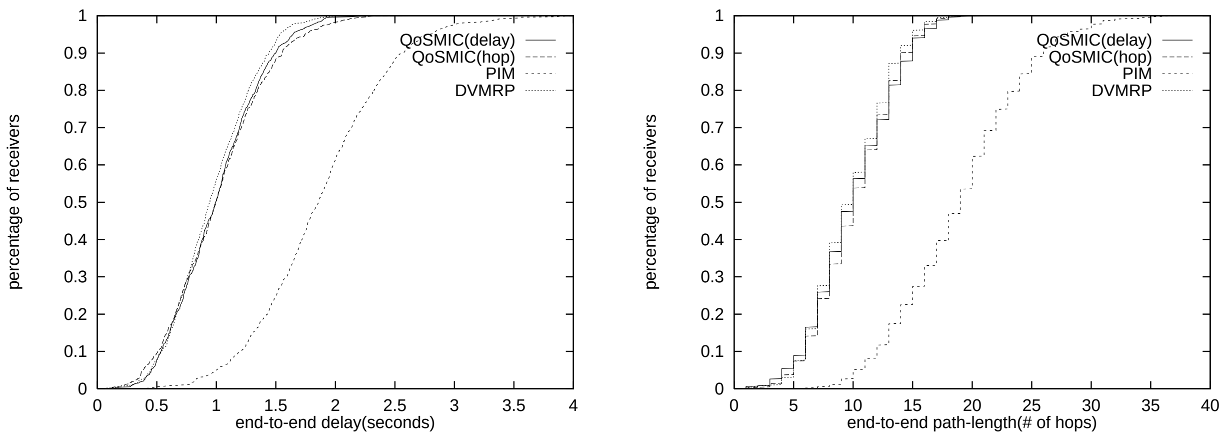
Figure 1 compares the performance of QoSMIC, PIM and DVMRP on the Mbone topology. Graph *a)* shows the fraction of receivers that obtain end-to-end delays lower than or equal to a certain value, expressed as a function of the delay value on the X-axis. Graph b) shows the end-to-end path-length in the same way. Ta-

[1] An in depth comparison of the effect of using different metrics can be found in [7].

[2] Typically in QoS routing, delay is used a constraint, while other criteria, such as hop-count or remaining bandwidth, are used as the optimization criteria. However, by using delay as an optimization criteria, we can produce a single graph like Figure 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.

Topology	Metric	PIM	DVMRP	QoSMIC(hop)	QoSMIC(delay)
MBone	Max	2.257	1.341	1.664	1.350
	Avg	1.183	0.622	0.653	0.628
Flat 1 (deg. 3.78)	Max	0.878	0.521	0.810	0.764
	Avg	0.478	0.249	0.325	0.305
Flat 2 (deg. 4.71)	Max	0.853	0.465	0.696	0.629
	Avg	0.441	0.228	0.319	0.277

Table 1. Maximum (over sources) end-to-end delay observed by receiver in seconds.



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

Figure 1. Fraction of nodes below an end-to-end performance value: MBone graph.

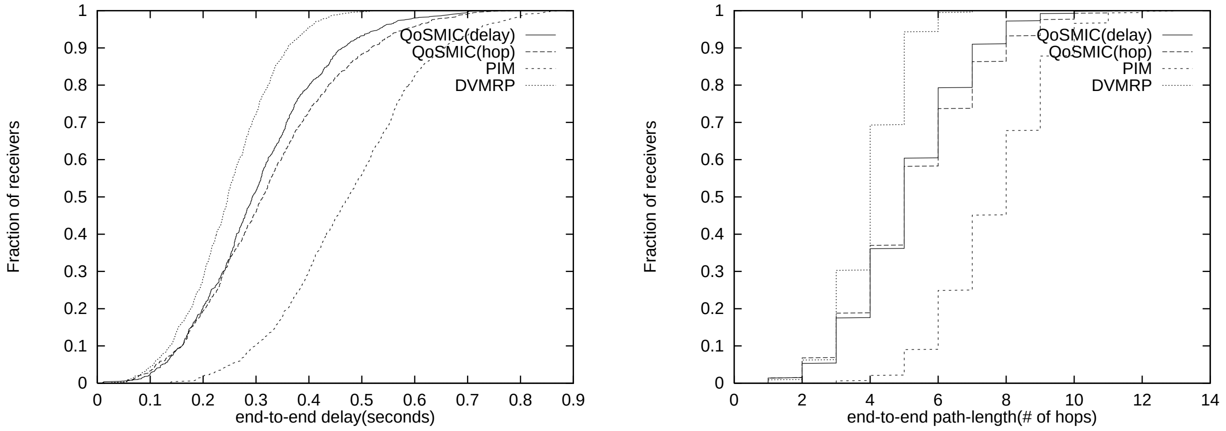
ble 3 reports the maximum, minimum and average end-to-end delay in our experiments. The delays are computed by simply adding the link delays from the source to the destinations, and computing maxima or averages as appropriate.

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 small 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.

In Figure 2, we show the performance on the flat-random Flat 1 (avg. degree 3.72). Compared to PIM, QoSMIC(hop) already improves the blocking probability for delay sensitive applications considerably. For example, for a .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%.

When we look at QoSMIC(delay), the blocking for a .5 second delay requirement further reduces from 28% to 20%. This represents a 2.7 fold increase in the number of satisfied participants over PIM, from 30% of the receivers to 80%. Further, 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. As already mentioned, QoSMIC can also construct source based trees, to improve the end-to-end delay to the DVMRP line, but we do not explore this aspect in this paper due to the scaling problems of source-specific trees.

In Figure 3, we compare the end-to-end performance



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

Figure 2. Fraction of nodes below an end-to-end performance value: Waxman's flat euclidean graphs (node degree 3.78).

of the protocols on the flat-random Flat 2 (avg. degree 4.32). The two QoSMIC versions strike the balance between PIM and DVMRP as in the previous experiment. The improvement in blocked calls is just as dramatic. However, here the gap between QoSMIC(delay) and QoSMIC(hop) increases. This indicates that denser topologies, with more alternate paths, benefit more from QoS metric based routing such as QoSMIC can provide. This not only confirms our intuition, but indicates the importance of QoS routing in future networks with more dense topologies. However, more extensive experiments over a wider range of node degrees are needed to confirm this.

4 Conclusions

In this paper, we examined the importance of QoS-sensitive routing for real-time multicast applications. We show that QoS-sensitive protocols can reduce call blocking for applications with strict QoS requirements significantly, in some cases increasing the number of satisfied participants by a factor of 2.7. If we imagine a commercial scenario, with paying customers, it is easy to see the importance of QoS-sensitive routing protocols.

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, we can improve the routing further by selecting the path that meets our 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 conclusion, QoS-sensitive routing promises gain in the efficiency of the network to support real-time multicast applications, and QoSMIC is a promising step in the right direction.

In the future, we want to examine the effect of other routing criteria, such as bandwidth available on a link, on the performance of the protocols. We also want to compare the protocols on other metrics, such as setup latency and control overhead, and examine the trade-offs of these metrics with the optimality of the tree. We plan to examine more scenarios from traces of real multicast events and more real topologies.

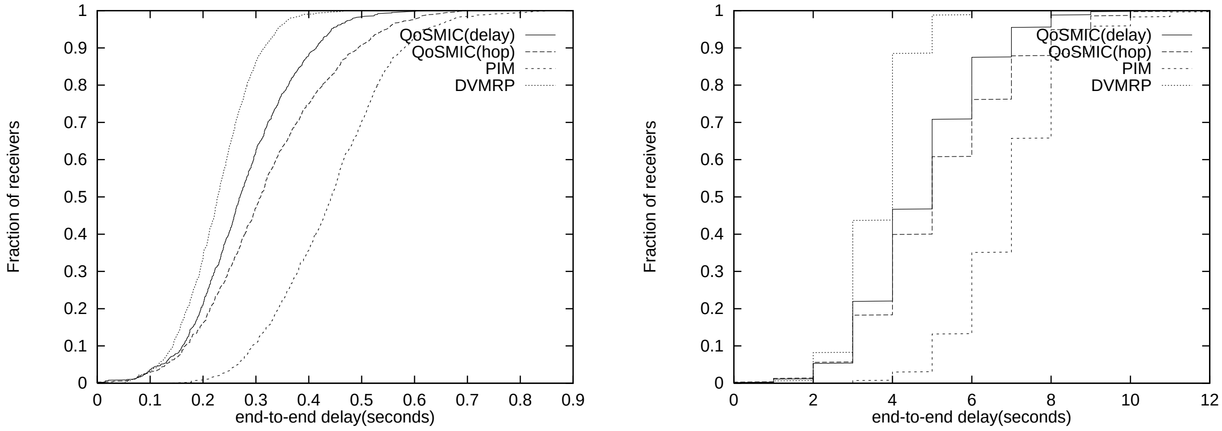
References

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

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

[3] A. Banerjea, M. Faloutsos, and R. Pankaj. Designing QoSMIC: a QoS Multicast Internet protoCol. Internet Draft: draft-banerjea-qosmic-00.[ps,txt], available from ftp://ftp.ietf.org/internet-drafts/, WORK IN PROGRESS, 1998.

[4] 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.



(a) End-to-end delay.

(b) End-to-end path.

Figure 3. Fraction of nodes below an end-to-end performance value: Waxman's flat euclidean graphs (node degree 4.32).

IEEE Journal of Selected Areas in Communications, 15(3):304–315, April 1997.

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

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

[7] K. Carlberg and J. Crowcroft. A comparison of shared trees constructed with different metrics. *Work in progress.*, 1999.

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

[9] 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. IETF RFC 2117, available from ftp://ftp.ietf.org/internet-drafts/, 1997.

[10] 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.

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

[12] 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.

[13] R. Perlman *et al.* 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/, WORK IN PROGRESS, 1999.

[14] 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.

[15] D. Thaler, D. Estrin, and D. Meyer. Border gateway multicast protocol (BGMP): Protocol specification. Internet-Draft: draft-ietf-idmr-gum-02.txt, WORK IN PROGRESS, 1998.

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

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

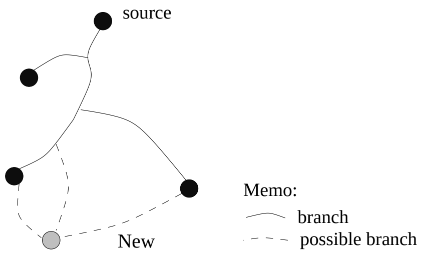
[18] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for Internet networks. *IEEE/ACM Transactions on Networking*, 5(6):770–783, December 1997. http://www.cc.gatech.edu/projects/gtitm/.

A Overview of QoSMIC

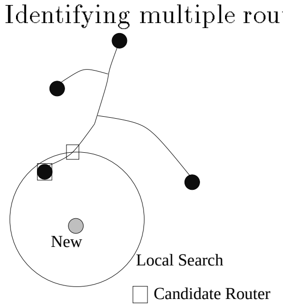
Here, we present an overview of the QoSMIC multicast protocol. In brief, QoSMIC constructs a shared tree in a receiver-driven manner. QoSMIC can also create Source-Based Trees, which can improve further the end-to-end QoS [11], at the cost of source specific state in the network, but we do not examine this case here. For each joining node it identifies a number of candidate paths. The protocol then selects a path that can provide the desired QoS service, and optimizes its other selection criteria.

The creation of the multicast tree can be decomposed in three phases. 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. The Search phase is carried out by two mechanisms: the Local Search and the Multicast Tree Search[11]. 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 Fig.4.(b)). The Multicast Tree Search is an innovative characteristic of QoSMIC which reduces the control overhead [11]; the in-tree routers run a distributed algorithm to select candidates with lower control overhead. In the experiments presented in this paper, the search procedure used is the Local Search procedure, allowing us to generalize the results of the experiments to YAM as well. However, this does raise the issue of the trade-off of control message overhead vs. the optimality of the computed tree, which we shall explore in future work.

An advantage of QoSMIC, is that the routes it computes are provably loop-free [11]. Furthermore, as in PIM and CBT, the routes are pinned for the duration of the multicast session, so that issues of stability do not arise. Finally, the routing table state requirements of QoSMIC are identical to PIM or CBT.



(a) Identifying multiple routes.



(b) The Local Search procedure.

Figure 4. An overview of QoSMIC.