2002-10-01

Performance Evaluation of Path Searching Heuristics for Multicast QoS Routing

Daniel Zappala
daniel_zappala@byu.edu

Dayi Zhou

Original Publication Citation

Daniel Zappala and Dayi Zhou, "Performance Evaluation of Path Searching Heuristics for Multicast QoS Routing", IEEE 11th International Conference on Computer Communications and Networks (ICCCN), October 22.

Performance Evaluation of Path Searching Heuristics for Multicast QoS Routing

Daniel Zappala and Dayi Zhou
Department of Computer Science
1202 University of Oregon
Eugene OR 97403-1202, USA
zappala|dayizhou@cs.uoregon.edu

***Abstract*—Quality of Service routing for multicast enables a group member to find and install a branch of the multicast tree that can meet its QoS constraints. The most promising protocols in this area use receiver-oriented path searching heuristics to find feasible routes. In this paper, we examine the performance of the path searching heuristics used by these protocols, to determine which techniques are the most effective. We find that several low-overhead path searching heuristics are effective, and that generic path searching can work as well as targeted, QoS-specific searching.**

I. Introduction

There has been a long history of research to support multicast Quality of Service (QoS) routing in the Internet. Current multicast routing protocols compute paths based on hop count and policy, without regard to the capabilities of the path nor its current load. When a network supports QoS, an application may specify a QoS request to the network as some combination of delay, bandwidth, and loss characteristics. A QoS routing protocol then computes a path that has available resources and installs this path for the application.

Computing a QoS-capable path in a scalable manner has proven to be very difficult since network conditions may change rapidly. The most promising recent protocols use receiver-oriented mechanisms to find and install routes that can support a given receiver's QoS requirements. These protocols include include APR [1], QoSMIC [2], [3] and QMRP[4], [5]. While the overall performance of these protocols have been studied, primarily in isolation, little has been done to determine which techniques for path searching are most effective.

In this paper we isolate the path searching component of these multicast QoS routing protocols and study which types of path searching are most effective for QoS routing. Our goal is to find low-overhead techniques for discovering QoS-capable paths, so that QoS routing could be deployed in large-scale, wide-area networks.

In addition, we test the effectiveness of using generic path searching, based only on finding alternate paths, versus collecting QoS metrics during the search process. APR is an example of the former type of protocol – it finds alternate routes but does not test them for available bandwidth or other QoS metrics. In this case, availability of resources is determined only when the path is installed. This characteristic makes APR suitable for non-QoS applications that need an alternate path, but could potentially weaken it for QoS applications. To determine how important this capability is, we test APR alongside QAPR, which adds collection of QoS metrics to the APR path search heuristic.

We believe that both APR and QoSMIC hold promise because they use highly localized searches. APR uses a hop-scoped search to find feasible paths from a receiver's neighbors to the source. QoSMIC uses a similar hop-scoped search to find a path to the existing multicast tree. QMRP, on the other hand, provides a more targeted type of search, but it may involve additional overhead compared to a local search. With QMRP, a new receiver tries to follows its shortest path to the source, then backtracks to alternative routes when resources are not available. This approach requires global deployment of a standard search mechanism, whereas APR and QoSMIC are better suited to incremental deployment and application-specific searches.

Our performance study indicates that APR's generic path searching provides consistently good performance across different types of networks and for different group sizes. Our results also demonstrate that APR is well-suited for both unicast and multicast routing, whereas the hop-scoped searching used by QoSMIC is most effective when group membership is dense. Finally, we find that APR generally performs as well or better than both QMRP and QAPR, with lower overhead and only slightly higher join delay.

Taken together, our results indicate that generic path searching can work as well as targeted, QoS-specific searching. This means that a single searching heuristic may be utilized for a wide range of applications, including both alternate path routing and QoS routing, as well as both unicast and multicast routing.

II. Background

The goal of a multicast QoS routing protocol is to build multicast trees that can support an application's needs relative to bandwidth, delay, loss, and other performance characteristics. One of the obstacles to deploying QoS routing in the Internet has been scalability. For this reason, recent protocols use receiver-oriented path searching mechanisms, where each receiver is responsible for finding and installing its path to the source.

A good path search mechanism generally involves trading efficiency for overhead. While link-state QoS routing protocols maintain global knowledge of the network and all link states, receiver-oriented protocols obtain only partial information of the topology and status of the network. A common way to do this is through a hop-scoped or expanding-ring search [6].

0-7803-7533-X/02/$17.00 (C) 2002 IEEE

A. APR

The APR protocol always begins by trying to use the shortest-path route from a receiver to the source of the multicast group. If the given QoS cannot be met on the shortest-path, then APR marks the link(s) that do not have enough resources and uses an expanding-ring search to collect feasible paths to the source of a multicast group.

Starting with a hop count of one, the receiver floods a search message to all of its direct neighbors. Each neighbor then returns to the receiver the shortest path from itself to the source of the multicast group. The receiver uses these paths to build a partial map of the network and determines whether any feasible path exists. A feasible path is any path that avoids links that are marked as not having sufficient resources. Any feasible paths that are found are tried. If the one-hop search fails to successfully find a path for the receiver, then the search is expanded to two-hops. This continues up to a configured maximum number of hops. If the receiver fails to find a path, it may also ask the sender to perform its own local search and return a feasible path.

Several features make APR easy to deploy. First, during an APR local search, neighbor routers do not need to keep any state concerning the search nor knowledge of the current multicast tree. Second, returning a path to the source in question is a relatively simple operation. If routers use a unicast path vector routing protocol, then neighbors can use their local routing table to immediately return the desired path. Otherwise, the neighbor can use *traceroute* to discover the path and return it.

APR is also suitable for unicast QoS routing since it finds routes to a source and not to a multicast tree. In fact, the path searching function of APR is completely independent of multicast routing and can be used for any number of uses. Once a route is found, a separate path installation protocol, called APM, is used to install the route in the multicast tree [1].

B. QoSMIC

The QoSMIC protocol, which is based on YAM [7], is designed specifically for multicast QoS routing. QoSMIC uses a hop-scoped search, similar to APR, but tries to find a route to the existing multicast tree, rather than to the source.

To join a group, a QoSMIC receiver floods a bid request message within a limited scope. If a router receives this message and it is on the multicast tree for the given group, then it responds with a bid, which is a path annotated with the QoS it can support. The bid is unicast to the initiator of the search and collects resource information along the way. The receiver collects the bids and chooses one according to some QoS metrics, then installs this path.

The QoSMIC local search is designed to be efficient when there is a nearby receiver. In this case, connecting to a nearby part of the multicast tree may consume fewer resources than a path to the source. However, the scope of the search must naturally be limited, so it is not effective when group membership is sparsely distributed.

Thus to augment local search, QoSMIC also includes a *manager-assisted search* mechanism to receive additional bid messages is a more scalable manner. In this case, a receiver contacts a group manager, which has knowledge of the entire tree. The group manager selects some nodes on the multicast tree to provide bid messages to the receiver.

Note that QoSMIC is not appropriate for unicast QoS routing, as finding a route depends on having a multicast tree to connect to. Similarly, when there are very few members, QoSMIC will have a hard time finding feasible paths. This makes bootstrapping the first few members rather difficult. We demonstrate this drawback in a later section.

C. QMRP

The QMRP protocol uses a more targeted search than both APR and QoSMIC. Like APR, QMRP begins by searching the shortest-path route to the source of the multicast group. A new member sends a REQUEST along its shortest-path route to the source, checking each link to see if it has available resources. If the desired QoS cannot be met at a given link, then QMRP backtracks to the previous hop. This previous hop copies the REQUEST message and *detours* it to all adjacent nodes (not counting the two nodes that lie forward and backward along the shortest path). These nodes then try to continue following their shortest-path route to the source of the multicast group, and continue as long as each link has available resources. Any time a reservation cannot be made, backtracking occurs again. Any message that successfully reaches the source or the existing multicast tree has found a feasible path. Where two feasible paths meet, a node must decide which path to use.

In its basic form, QMRP can explore an unlimited number of paths, which can lead to a high success rate but also very high overhead. Thus, the authors recommend limiting the search overhead by restricting the number of nodes that can detour a REQUEST message. This restricted form of QMRP is designated QMRP-n, meaning n nodes between the new member and source can be actively detouring a REQUEST message.

One of the problems with QMRP is that its design makes it difficult for a REQUEST message to travel far away from the shortest-path route. When a node detours a request message, sending it to neighboring nodes, these nodes will continue sending the REQUEST message on their shortest path to the source. However, in many cases this path may lead back to the detouring node, preventing QMRP from exploring alternate routes. We refer to this problem as a *false detour*. While the same thing can happen with APR, it can send messages two or three hops away in order to avoid this problem and find alternate paths.

In addition, the approach taken by QMRP requires substantially more deployment effort for the protocol to be successful. With APR, paths can be collected individually from any node that runs APR. Similarly, with QoSMIC, the only routers needed to support path collection are those in the immediate vicinity of a new member. With QMRP, on the other hand, each node along the path from the new member to the source, plus nearby nodes, must support its path searching protocol. In addition, each active node must maintain a state machine to track the status of the search. In contrast, APR and QoSMIC require no state be

kept, and with APR search strategies and route selection can be customized for each receiver or application.

III. Modeling the Path Search Function

The goal of our work is to model the path searching function of a QoS routing protocol. Accordingly, we have isolated this part of the protocols we study from other aspects, such as path installation. This allows us to fairly compare the path searching algorithms using the same QoS routing framework.

A. General Model for Receiver-Oriented, Multicast QoS Routing

We have developed a general model for receiver-oriented QoS routing that uses two distinct components: the path searching protocol and the path installation protocol. The receiver uses the path searching protocol to find a feasible route, then asks the path installation protocol to install this route into the multicast tree.

Although some QoS routing protocols, such as QMRP, integrate path installation with path searching, we have separated the path searching component for the purposes of our study. This enables us to give an accurate and fair comparison of their path search capability.

We use APM as the path installation protocol [1]; this is the only generic, multicast path installation protocol that has been developed.

Our model consists of the following steps:

1) The receiver first tries to use the shortest-path route from itself to the source of the multicast group. The receiver invokes APM to install the shortest-path route and waits for a response. If the QoS of the path is sufficient, APM will return a CONFIRM message and the receiver is finished. Otherwise, it will return FAILURE.
2) If the QoS of the shortest-path route is not sufficient, the receiver invokes a path searching protocol to find a feasible alternative route. This protocol will either return a path or indicate that no path is available. If no path is available, the receiver leaves the group.
3) If a path is available, the receiver will try to install this path with APM. If the receiver gets a CONFIRM message the receiver is finished. If it receives a FAILURE message, it loops back to step 2.

Our model always tries to use the shortest-path route before trying any alternative routes. This same technique is used by both APR and QMRP, so that when congestion is not present the same routes are used as with a non-QoS protocol. While QoSMIC does not incorporate this technique, our model allows us to more fairly compare QoSMIC path searching with the other protocols.

Note that the model is phrased in terms of a source-specific multicast tree, but can be applied equally well to a shared, core-based tree.

B. Basic Path Searching Algorithms

We simulate the path searching components of APR, QoSMIC, and QMRP. Because our preference is for localized, receiver-oriented (and thus scalable) searches, we eliminate some parts of the APR and QoSMIC protocols. For APR, we omit the search mechanism whereby a receiver can contact the source to find a route. This occurs only when the receiver cannot find a path on its own and is particularly effective in hierarchical networks [1]. For QoSMIC, we omit the manager-assisted search, which is likewise used for cases when the receiver cannot find a feasible route on its own. Manager-assisted search is useful for cases when the multicast tree is sparsely-distributed. Naturally, we would expect the performance of both APR and QoSMIC to improve if these mechanisms were included in our study.

The algorithms we implement are described below. If QoS metrics are collected, they are for the path from the source (or tree) to the receiver. Some algorithms utilize an expanding ring search to help decrease overhead. This means that an algorithm is first run using a hop count of 1, then 2, up to a maximum of n.

B.1 APR-n

APR is a simple local search for alternate paths to the source of a multicast tree:

- The new member uses expanding ring search to send a probe to all neighbors within n hops. The probe contains only a source identifier.
- Each neighbor returns the path from the member to the neighbor, plus the path from the neighbor to the sender.
- The new member collects all paths and chooses the shortest one for installation with APM. If this path is not available (APM returns a FAILURE), the member iterates through the remaining paths, until all paths have been exhausted.

The overhead for this search can be very low; if routers use a path-vector protocol or a distance-vector protocol with source tracing [8], [9], then a neighbor already has its path to the source stored in its routing tables and can immediately respond to the query without further probing.

B.2 QAPR-n

QAPR is an extension of APR to collect QoS metrics for the paths it searches:

- The new member uses expanding ring search to send a probe to all neighbors within n hops. The probe contains only a source identifier.
- When traversing the path from the new member to a given neighbor, a probe checks each link for its available resources. If some link cannot support the desired QoS, the probe stops and returns a failure indication.
- Once a neighbor is reached, a probe traverses the path from the neighbor to the sender. Again, if some link is reached where the QoS is insufficient, the probe stops and returns a failure indication. Otherwise, if the sender is reached, the probe returns the path from the member to the neighbor, plus the neighbor to the sender, along with the minimum available QoS for the path.
- The new member collects all paths and chooses the shortest path with available QoS for installation with APM. If this

path is not available (APM returns a FAILURE), the member iterates through the remaining paths, until all paths have been exhausted.

B.3 QoSMIC-n

QoSMIC searches for paths to the current multicast tree:

- The new member floods a probe to all nodes within n hops to try to find nodes that are part of the multicast tree for the given group. The probe contains a source identifier and group identifier. At each hop, the probe checks to see if current node is already part of the given multicast tree.
- If a node on the tree is found, the probe returns the path from the new member to the node on the tree, along with the minimum available QoS for the path.
- The new member collects all paths and chooses the shortest path with the largest available capacity for installation with APM. If this path is not available (APM returns a FAILURE), the member tries the next shortest path until all paths have been exhausted.

B.4 QMRP-n

QMRP uses backtracking to find a path to the source of a multicast tree:

- The new member sends a probe along the shortest path to the source, containing a source identifier, group identifier, and the limit n. At each hop, the probe checks to see if current node is already part of the given multicast tree.
- If the probe finds a node that is already on the multicast tree, then it returns the path it has accumulated so far, along with the available QoS for the path.
- If the probe encounters a link with zero capacity, it backtracks to the previous hop. This hop decrements the limit n by 1; if n is zero then the probe stops and returns a failure indication. Otherwise, this hop copies the probe, with the new limit $n-1$, and sends it to all adjacent nodes, not counting the nodes that lie backward and forward along the shortest path.
- Subsequent nodes send the probe on their shortest path to the source, backtracking and copying the probe as necessary.
- Any probe that successfully reaches the source or a node on the multicast tree returns its accumulated path, along with the minimum available QoS for the path.
- The new member collects all paths and chooses the shortest path with the largest available capacity for installation with APM. If this path is not available (APM returns a FAILURE), the member iterates through the remaining paths, until all paths have been exhausted.

B.5 Path Selection

For QAPR, QoSMIC, and QMRP, a new member may sometimes find multiple feasible paths from itself to the source or the multicast tree. In this case, the new member must use some algorithm to choose which path it will try first. For the results shown in this paper, APR and QAPR use the shortest path and QoSMIC and QMRP use the shortest path with the largest available capacity. We have experimented with several other algorithms and found the differences between them are small.

IV. Performance Evaluation

To evaluate the performance of the path searching algorithms, we use a *dynamic load model* with bandwidth as the QoS metric. For a given network, we assign each link an integer value k, representing its capacity. We then create a sequence of groups, each of size g, with group members chosen randomly from among all nodes in the network. When group members use APM to install a route, they reserve 1 unit of capacity on each link. By increasing the number of groups in the network, we increase load and the chance that some link will run out of available capacity. For each of the networks we study, we increase the number of groups in the network until the best algorithm has a 90% success rate. This covers what we consider to be an acceptable operating range for the network.

More complex models, using various combinations of bandwidth, delay, jitter, and loss have also been used in the literature. For our purposes, a simple model is sufficient to compare the performance of the path searching algorithms. Bandwidth was likewise used as the relevant metric in [4].

Our dynamic load model does, however, represent a somewhat more realistic approximation of network load than has been used in past multicast QoS routing studies. Previous studies use a *congested links model* in which links are randomly chosen as either congested (having no available capacity) or uncongested [4], [1]. With our model, if a particular router initiates many connections, then the links near that router will likely become congested, while other areas of the network may remain relatively unused.

We use two types of randomly-generated topologies in our study – flat and hierarchical. Our previous work in this area has shown that these topologies represent extremes in terms of routing performance [1], [10] and are thus good candidates for this study. Topology set F100 contains ten 100-node, flat topologies, with each link having capacity of 20. While this capacity is small – only 20 groups can be supported per link – it enables us to quickly reach loads where links become congested. Topology set T100 contains 10 100-node, transit-stub topologies. Links within stub networks have a capacity of 20, transit-stub links have a capacity of 50, and backbone links have a capacity of 150.

In addition to these networks, our study includes a 4000-node map of the MBone collected by researchers at ISI in 1999 [11]. Because the dynamic load model is computationally expensive, we use the basic congested links model to simulate path searching algorithms on this map.

A. Join success rate

We first examine success rate for sparse and dense groups to determine which path searching algorithms work well for these situations. We define the success ratio as s/j, where s is the total number of successful joins over the lifetime of the simulation and j is the total number of join attempts over that time.

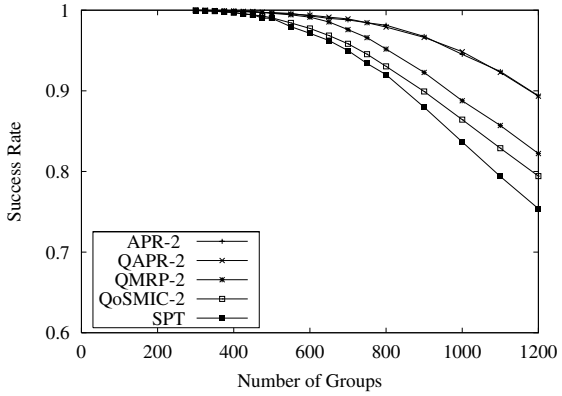

Fig. 1. Success Rate: F100, 2-member groups

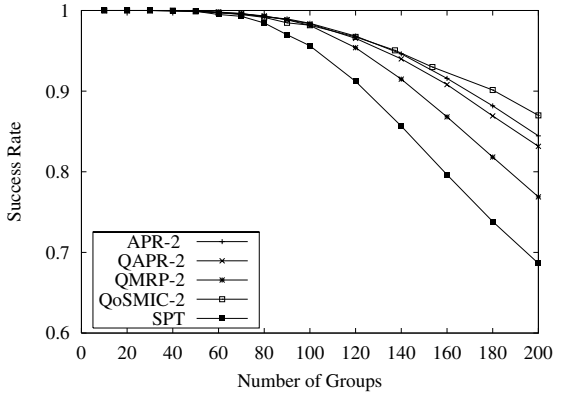

Fig. 2. Success Rate: F100, 20-member groups

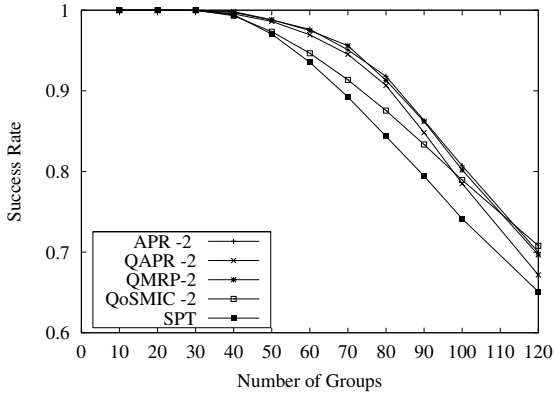

Fig. 3. Success Rate: T100, 10-member groups

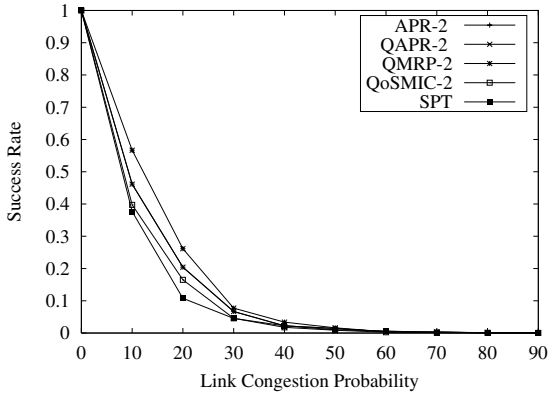

Fig. 4. Success Rate: MBone, 100-member groups

For small groups, APR and QAPR perform the best. Figure 1 shows the success rate for 2-member groups on topology set F100 as load increases. The performance of QAPR and APR is very close, as expected; their primary difference should be in terms of setup delay, since QAPR collects information on available capacity as it searches.

Neither QMRP nor QoSMIC perform as well for small groups. For QMRP, the primary reason is likely the problem of *false detours*, as described in section II-C. False detours are a particular problem for sparse or small groups, because the multicast tree may not be located nearby a detouring node. As the group grows denser, there is a better chance that even a false detour will reach a nearby node of the multicast tree. The success rate of QoSMIC is naturally limited for small groups due to its reliance on a hop-scoped search and our elimination of manager-assisted search. For very small or sparse groups, a hop-scoped search will not find nearby nodes on the multicast tree, limiting group members to only their shortest-path route.

As the group size increases, APR and QAPR continue to perform well, and QoSMIC's performance increases substantially. Figures 2 shows the success rate for each of the protocols using topology set F100 with 20-member groups. Clearly, QoSMIC's style of local search is very sensitive to the distribution of group members and does well only when membership is dense.

For both the transit-stub networks in set T100 and the MBone map, QMRP performs slightly better than APR. Figure 3 shows the results for T100 and Figure 4 shows the results for the MBone map. APR, QAPR, and QoSMIC are limited in hierarchical networks because they can only search near the receiver, whereas QMRP can search near the sender if this is where congestion occurs. APR addresses this problem by allowing the receiver to ask the sender to perform its own local search. If this option is included, APR performs as well or better than QMRP on these hierarchical networks.

B. Search Overhead

Search overhead is an important component of algorithm performance. Clearly the best algorithm, in terms of success rate, is a link-state protocol that knows the entire topology, the available capacity of every link, and the location of every multicast tree [1]. However, this performance comes at the price of very high overhead to distribute link-state advertisements every time capacity or group membership changes.

We measure the overhead of path searching algorithms by the number of nodes contacted in order to perform the search. This indicates the fraction of the topology seen by the new member as it finds a path.

In our experiments flat random networks, QAPR has the highest overhead, as shown in Figure 5. This graph shows the number of messages per Join for 20-member groups on topology set F100. While the overhead of QAPR is much smaller than a protocol that floods the entire network, it is about 3 times higher

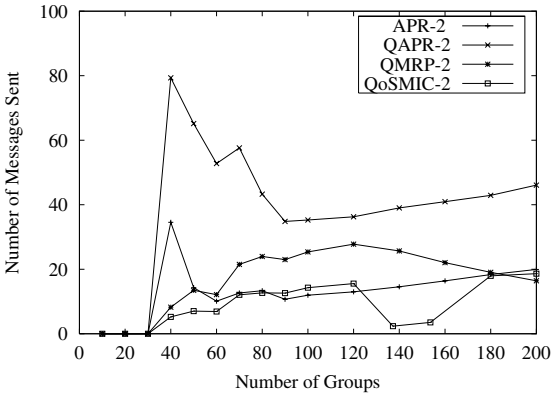

Fig. 5. Search Overhead: F100, 20-member groups

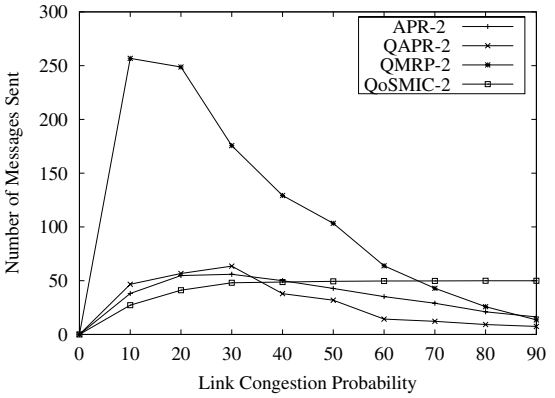

Fig. 6. Search Overhead: MBone, 100-member groups

than the other algorithms when load increases. This overhead is due to QAPR checking the available capacity on all the paths from its neighbors to the source. In contrast, QMRP uses a more focused search, checking capacity only along the shortest path and then branching off from there as needed. Hence QMRP, as long as it limits the number of detouring routers, keeps its overhead in check. APR does not send many messages in our simulations because we assume each router knows the entire path from itself to all other nodes – thus the only messages sent are to nodes within 2 hops. The overhead of QoSMIC is also limited due to its 2-hop search. For 2-member and 10-member groups, our results show the same relative performance.

The search overhead for QMRP is much higher on the MBone map than on the random networks, as shown in Figure 6. This behavior is likely due to the fact that when QMRP backtracks it detours its REQUEST message to all adjacent nodes. In a hierarchical topology, such as the MBone, there are some transit nodes that have many attached stub networks and thus a high degree of connectivity. When QMRP detours into these stub networks, it can incur high overhead. This behavior can be particularly pronounced in real networks, since degree distributions in the Internet have been shown to have a power law distribution [12]. To avoid this consequence, QMRP should include a limit on the number of detours it takes.

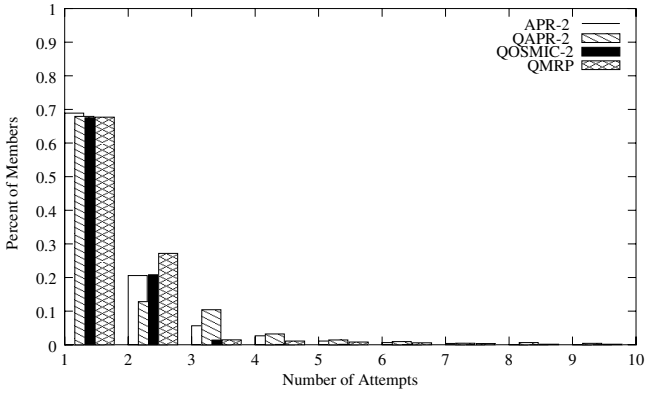

Fig. 7. Join Delay Distribution: F100, 150 20-member groups

C. Join delay

Join delay is the time taken for a new member to join the multicast tree successfully. Because path searching occurs before path installation, it is possible that an installation may fail and a different route tried. This is especially true of APR, which does not attempt to learn about available capacity when it searches for feasible routes.

We measure join delay by the number of join attempts per new member, regardless of whether that member was successful. Our results are very similar if we only consider successful joins. Note that there is often a trade-off between success rate and join delay. Trying more paths will likely lead to a higher success rate but also a higher join delay.

Not surprisingly, APR has the highest number of attempts per join, since it does not check available capacity as it searches. However, the number of join attempts for all protocols is generally below 2, and is much smaller at reasonably small loads.

Examining the distribution of join delay at high load, we see that there is little difference among the protocols, with most succeeding after 1 or 2 attempts. This result is shown in Figure 7, which illustrates the join delay distribution for 150 20-member groups using F100. In this figure, all protocols except QoSMIC have long tails. We could reduce the overhead of these protocols by limiting the number of join attempts to 4, without significantly altering the overall success rate

Our results for 2-member and 10-member groups show the same relative performance among the protocols. Performance is likewise the same for topology set T100 and for the MBone map.

D. Path Length

Our final measure of QoS routing performance is path length. In order to maximize the number of groups that can be supported – and hence maximize revenue – it is best to utilize short paths. We normalize path length across various members by measuring the number of *extra hops* incurred by a QoS route, relative to the shortest path route. Extra hops is defined by $l_{rt} - l_{spt}$, where l_{rt} is the length of the route from the source to the member and l_{spt} is the length of the shortest path route from the source to the member.

Even at high load, the overwhelming majority of group members tend to use their shortest path or a QoS route of equiva-

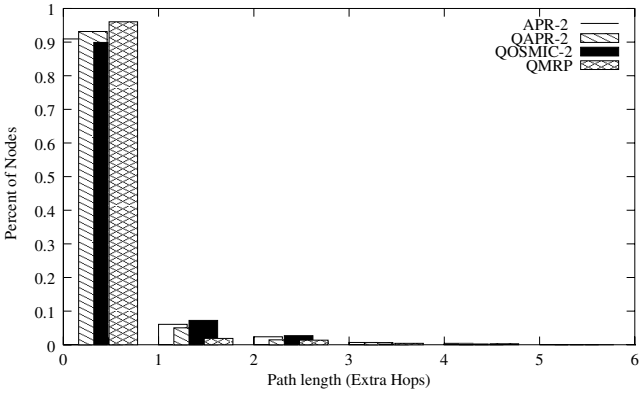

Fig. 8. Path Length Distribution: F100, 150 20-member groups

lent length. Figure 8 shows the distribution of the path length across all group members for 150 20-member groups using F100. About 90% to 95% of all members use a shortest-path route, and about 5-10% use 1-hop routes. QMRP uses more shortest paths and fewer 1-Hop paths than the other protocols. We could easily limit all protocols to 2 or 3 hops beyond the shortest-path length and not significantly affect the success rate. Our results are similar for topology set T100 and the MBone map.

E. Combined Path Searching Algorithms

Based on the results of our experiments, we implemented one path searching algorithm that combines QoSMIC and APR and another that combines QoSMIC and QAPR. The combined algorithms give APR and QAPR the ability to find paths to any node on the existing tree, rather than just the source of the tree. However, we did not find much of a performance gain for these combined algorithms.

V. Discussion

Our overall results indicate the feasibility of scalable, receiver-oriented multicast QoS routing. APR, QAPR, and QoSMIC each perform well, and both APR and QoSMIC do this with low overhead. All of the algorithms we tested performed well with respect to join delay and path length.

Considering its simplicity, APR performs rather well in comparison to the other path searching algorithms. It performs consistently across both flat and transit-stub topologies and over a range of group sizes. Its overhead is quite low, and it is easy to deploy. Moreover, APR can perform as both a multicast and unicast path searching algorithm, since its performance with very small groups is as good as with large groups. Combined with our previous work, these results indicate that APR has the potential to serve as a general-purpose path searching protocol, suitable for both alternate path routing and QoS routing.

Of the QoS-specific searches, QoSMIC clearly has the best performance, except when group membership is sparse. Moreover, QoSMIC's overhead is low because it uses a simple hop-scoped search. QMRP did not perform as well, and we suspect this is due to the problem of false detours. It is likely that QMRP could be re-designed to overcome this problem; however, this lends more weight to our arguments in favor of end-system searching. For an end-system protocol, it is a much simpler matter to re-design and then deploy a new protocol. Finally, our results show that QAPR has good performance, but very high overhead since it must probe each neighbor's path for the current QoS metrics. From our study, it appears this overhead does not buy a significant performance improvement.

VI. Future work

As part of our ongoing work, we are continuing to refine our simulation models and verify our results on additional types of networks. This includes:

- Implementing trunk reservation so that shortest paths are preferred under high load. This technique has proven useful in telecommunication networks to improve network utilization. We can do this by placing restrictions on each link so that a given percentage of flows must use shortest paths when load is high.
- Using a more realistic model of group lifetimes and network load. Currently, groups are only added to the network as a rough estimation of network load. We would like to instead have groups form and disband, plus have individual members join and leave. We would also like to generate some dynamic background traffic to help vary the load at each link.

References

[1] D. Zappala, "Alternate Path Routing for Multicast," in *IEEE INFOCOM*, 2000.
[2] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of Service Multicast Internet protoCol," in *ACM SIGCOMM*, August 1998.
[3] S. Yan, M. Faloutsos, and A. Banerjea, "QoS-Aware Multicast Routing for the Internet: The Design and Evaluation of QoSMIC," *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, February 2002.
[4] Shigang Chen, Klara Nahrstedt, and Yuval Shavitt, "A QoS-Aware Multicast Routing Protocol," in *IEEE INFOCOM*, 2000.
[5] Shigang Chen, Klara Nahrstedt, and Yuval Shavitt, "A QoS-Aware Multicast Routing Protocol," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 12, December 2000.
[6] Stephen Deering, "Multicast Routing in Internetworks and Extended LANs," in *ACM SIGCOMM*, August 1988.
[7] K. Carlberg and J. Crowcroft, "Building Shared Trees Using a One-To-Many Joining Mechanism," in *ACM Computer Communication Review*, January 1997.
[8] C. Cheng, R. Riley, S.Kumar, and J.J. Garcia-Luna-Aceves, "A Loop-Free Extended Bellman-Ford Routing Protocol without Bouncing Effect," in *ACM SIGCOMM*, September 1989.
[9] B. Rajagopalan and M. Faiman, "A New Responsive Distributed Shortest-Path Routing Algorithm," in *ACM SIGCOMM*, September 1989.
[10] Daniel Zappala, Deborah Estrin, and Scott Shenker, "Alternate Path Routing and Pinning for Interdomain Multicast Routing," Tech. Rep. USC-CS-97-655, Computer Science Department, University of Southern California, June 1997.
[11] ISI Scan Project, "Mbone Map from 23 February 1999," http://www.isi.edu/scan/mbone.html.
[12] C. Faloutsos M. Faloutsos, P. Faloutsos, "On Power-Law Relationships of the Internet Topology," in *ACM SIGCOMM*, August 1999.