DIMACS Technical Report 2000-18
August 2000

A Scalable Distributed QoS Multicast Routing Protocol

by

Shigang Chen
Cisco Systems,
San Jose, CA, USA
sgchen@cisco.com

Yuval Shavitt*
Bell Labs
101 Crawfords Corner Rd.,
Holmdel, NJ 07733, USA
shavitt@ieee.org

*Permanent DIMACS member

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

Many Internet multicast applications such as teleconferencing and remote diagnosis have Quality-of-Service (QoS) requirements. Such requirements can be additive (end-to-end delay), multiplicative (loss rate) or with a bottleneck nature (bandwidth). For these applications, QoS multicast routing protocols are important in enabling new receivers to join a multicast group. However, current routing protocols are either too restrictive in their search for a feasible path between a new receiver and the multicast tree, or burden the network with excessive overhead.

In this paper we propose S-QMRP, a new Stateless Qos Multicast Routing Protocol that supports all three QoS requirement types. S-QMRP is scalable because it has very small communication overhead and requires no state outside the multicast tree; yet, it retains a high success probability. S-QMRP achieves the favorable tradeoff between routing performance and overhead by carefully selecting the network sub-graph in which it conducts the search for a path that can support the QoS requirement, and by auto-tuning the selection according to the current network conditions. S-QMRP does not require any global network state to be maintained and can operate on top of any unicast routing protocol. Our extensive simulation shows that S-QMRP performs better than the previously suggested protocols.

1 Introduction

Multicast is an efficient way to deliver content to a large group of receivers by using a tree structure along which all communication is carried out. Building a multicast tree, termed multicast routing, was studied extensively. Finding an optimal multicast tree, i.e., a tree that uses a minimal number of links, is known to be NP-hard. Thus heuristics were deployed to find trees that are "good enough". The simplest heuristic was to use the existing unicast routing paths and build a tree of shortest paths from the source (or some core node in case of several sources) to the destinations [BFC93, DEF+94]. This approach was adopted by the IETF in its multicast standardization effort, where shortest path trees are used as multicast trees: DVMRP [Dee89], PIM [EFH+98], MOSPF [Moy94].

The shortest-path trees work well for delivering information that is not QoS sensitive. However, when applications have non-trivial QoS requirements (delay bound, throughput, etc.), there is a high probability that the shortest-path trees would not have adequate characteristics [CNS00].

A *feasible tree* is a multicast tree that satisfies the QoS requirement. A *feasible tree branch* (*path*) is a path that connects a new group member to a multicast tree and has the resources to support the required QoS. The task of QoS multicast routing is to find feasible tree branches for new group members. Finding feasible tree branches is difficult in very large networks such as the Internet because it is impractical to maintain the global QoS state at any single node. A brute-force flooding algorithm that searches all possible paths in the network guarantees to find a feasible branch if one exists. However, the excessive overhead of full-scale flooding deems to be impractical for all but small networks. Thus, for applications that require QoS guarantees, recent research focuses on distributed multicast routing algorithms that search a selected subset of the network to find feasible tree branches for new group members [Sha96, CC97, CRS97, FBP98, CN99, CNS00].

A good QoS routing protocol achieves a favorable tradeoff between the routing overhead and the ability of finding a feasible path (often quantified as *success ratio* or *success probability*). In addition, a good protocol exhibits or optimizes other characteristics, such as minimizing the extra state information the protocol maintains in the network; decentralizing the routing operations to spread the workload; and adapting the routing activity according to the current network condition and avoiding the area where traffic congestion occurs. QMRP [CNS00] is a protocol that exhibits a very good tradeoff between the routing overhead and the success probability. In addition, QMRP has many of the good merits mentioned above. However, QMRP suffers from two drawbacks. First, QMRP deposits temporary state in the network routers for each join request. This hurts its scalability. The second drawback is that QMRP is suitable only for applications with non-additive QoS requirements such as bandwidth and buffer space, and cannot be used for additive/multiplicative requirements such as delay or packet loss. While spanning join [CC97] and QoSMIC [FBP98] do not have the above problems, they have much higher overhead and lower success probability [CNS00].

This paper suggests a new stateless QoS multicast routing protocol, S-QMRP, that shares some adaptive path-searching techniques of QMRP, but uses a completely novel design and

new routing mechanisms. It inherits the good routing merits of QMRP, but on the other hand solves the problems of QMRP. S-QMRP does not maintain any routing state other than the multicast tree itself at the routers, while QMRP needs to maintain a temporary state machine at intermediate routers for each new member. Another key difference is that S-QMRP searches for a feasible path starting from the current tree towards the new member. It has several advantages over QMRP that searches in the reverse direction from the joining new member to the tree. First, one can dynamically aggregate several join requests and maintain a single state per multicast group. This reduces the state in the routers especially in peak hours when many join requests arrive simultaneously. It also reduces the signaling load. Second, the search in the direction of the actual data flow enables us to handle additive QoS requirements by incorporating heuristics that rely on the fact that the length of the path to the joining new member can be obtained from the unicast routing protocol.

S-QMRP exhibits the following properties:

Scalability: The overhead of the protocol is very low, comparing with previous suggested protocols [CC97, FBP98]. This makes it more scalable in large networks.

Efficiency: The protocol achieves high success probability of finding feasible paths. It can potentially detect multiple feasible tree branches for a new member. A distributed process selects the best branch connecting the new member onto the tree.

Effectiveness: The protocol can support additive and non-additive QoS metrics.

State Freedom: Besides the multicast tree itself, the protocol does not deposit any temporary state information at routers during the routing process.

Robustness: The routing process is fully decentralized, which helps spreading the workload and reduces the chance of single-point failure.

Operability: Like PIM, the protocol can operate on top of any existing unicast routing algorithm. It does not require any extra global network state to be maintained in the network.

Loop freedom The protocol always constructs loop-free multicast trees.

The rest of the paper is organized as follows. Section 2 presents our network model. Section 3 presents the related work. Section 4 describes the routing protocol. Simulation results are provided in Section 5. Section 6 draws the conclusion.

2 Network model

We make the following assumptions about the network.

1. There exists an underlying unicast routing protocol which can deliver a message between any two connected nodes in the network. A node knows the length (number of hops) of the unicast routing path to any destination. Many widely used unicast routing protocols such as RIP, OSPF, and BGP provide this information.

2. Every node maintains its up-to-date local state, such as the delay of each outgoing link, which includes the processing time, buffering delay, and link propagation delay. Assume that once resources are committed, such delay can be assured during the lifetime of data communication. How to make resource reservation [FV90, ZDE+93] and what packet scheduling algorithms are used [ZK91, Zha95] are out of the scope of this paper.

A node is not required to maintain any global state information or the local state of any other node.

In such an environment, we study how to construct a QoS-constrained multicast trees. We use the *delay* constraint as example to illustrate the protocol, while other additive metrics such as *cost* can be supported similarly. Multiplicative metrics such as *loss rate* can also be supported by using the logarithm of the functions to convert them into additive metrics. Bottleneck metrics such as *bandwidth* are easier to handle: the links having the resources have zero weights; the links not having the resources have infinite weights; feasible paths are those whose weights are zero.

There are two types of multicast trees: *sender-based trees* (e.g., radio transmission) and *shared trees* (e.g., a conference call). In a delay-constrained sender-based tree, the delay from the sender to any other node in the tree has to be bounded by a delay requirement, D, while in a delay-constrained shared tree, the delay bound is required between any two tree nodes. The latter can be transformed to the former by choosing a core node and use QoS routing to this node with half of the delay requirement. Our solution is thus suitable for both trees, though we will use sender-based trees to illustrate the protocol in the rest of the paper.

We assume that any new member is able to map a multicast group address to the root node of the tree on demand possibly by a query/response Session Directory [HJ96].

We define the notations in the following. Let k and i be two on-tree nodes. The path in the multicast tree connecting them is called the *in-tree path*, denoted by $P_{k,i}$. The guaranteed delay bound of this path is called the *in-tree delay*, denoted by $delay(P_{k,i})$. Let T be the set of on-tree nodes and r be the root of the tree. A delay-constrained multicast tree requires that

$$\forall i \in T, delay(P_{r,i}) \leq D$$

We require each on-tree node i to know $delay(P_{r,i})$. In fact, as we will see later in the protocol description, our protocol makes sure that any node joining the tree will have this value.

We assume that each link (i, j) can ensure a certain delay bound (which might be infinity) for the Class of Service (CoS) with which the multicast group is associated. We denote this delay bound by $delay(i, j)$.

3 Related Work

We study the relate work in this section.

Single-path routing: Traditional multicast routing protocols [BFC93, DEF+94] connect new members to the multicast tree along unicast routing paths from the new members to the tree root (core). These protocols can be extended to work with QoS constraints [THWC99]. A simple strategy is to accumulate the QoS property of the new tree branch as it is constructed from the new member toward the core. At any intermediate node, if the accumulated QoS property violates the QoS constraint the routing fails. If the construction reaches an on-tree node and the QoS constraint is not violated, the routing succeeds. Such a strategy works well in an environment where there are usually sufficient resources and the QoS constraints can be easily met. However, when resource contention arises, multi-path routing increases the chance of success.

Spanning Joins [CC97]: In order to increase the success probability, the spanning-joins protocol proposed by Carlberg and Crowcroft relies on flooding-based multipath routing. A new member broadcasts join-request messages in its neighborhood to find on-tree nodes. Whenever an on-tree node receives the message, it sends a reply message back to the new member. The path of the reply message, determined by the unicast routing protocol, is a candidate path. The new member may receive multiple reply messages corresponding to multiple candidate paths. Each reply message collects the QoS property of the path it traverses. The new member selects the best candidate path based on the information received in the reply messages. Consecutive broadcasts are necessary to search increasingly larger neighborhood until on-tree nodes are found, and this process can increase the overhead significantly. In addition, we showed in [CNS00] that such flooding-based approach is not optimal as in many cases broadcast hits one or a few on-tree nodes and thus a small number of paths are searched.

QoSMIC [FBP98]: The QoSMIC protocol proposed by Faloutsos et al. avoids large-scale flooding by using two parallel routing procedures: *local search* and *tree search*. The local search is equivalent to the spanning-joins protocol, except that only a small neighborhood is searched. The tree search handles the case when there is no on-tree node in the neighborhood checked by the local search. In the tree search, a new member sends an M-JOIN message to a designated Manager node for the group. Upon receipt of the message, the Manager multicasts a BID-ORDER message in the tree to select a subset of on-tree nodes. The selected nodes send BID messages to the new member. The paths of the BID messages, determined by the underlying unicast routing protocols, are also candidate paths. The tree search allows QoSMIC to restrict its *flooding* local search in a small neighborhood. On the other hand, the tree search "floods" in the multicast tree, which may still reach most of the network for large, dense multicast groups.

QMRP [CNS00]: The QMRP protocol takes an adaptive approach. It starts with single-path routing. When the single-path routing fails, it switches to multi-path routing by incrementally adding new searching paths. A REQUEST message is first sent from the new member to the root of the multicast tree along the unicast routing path. As it travels,

it checks the resource availability of every intermediate node and proceeds only when the node has the required resources. If every node has the resources, QMRP becomes a PIM-like protocol and finds a feasible tree branch by traversing only a single path. If an intermediate node does not have the required resources, the REQUEST message is replicated and sent towards directions other than the unicast path. After such branching, the paths searched by all REQUEST messages form a searching tree. The size of the searching tree is determined by how many intermediate nodes do not have the required resources. Simulations show that QMRP has higher success ratio and lower overhead than the spanning-joins protocol and QoSMIC [CNS00]. However, it requires extra memory space at the intermediate router to maintain the temporary searching trees. In addition, it only works for non-additive QoS metrics such as bandwidth and buffer space.

In this paper, we propose a new protocol that share some ideas used in the design of QMRP [CNS00], but requires no routing state inside the network and can handle additive QoS metrics. The protocol achieves better success ratio and lower overhead than other protocols supporting addtive metrics such as spanning joins and QoSMIC.

4 QoS Multicast Routing Protocol

In this section, we discuss our design objectives, describe the new routing protocol, and present the pseudo code of the protocol.

4.1 Design objectives

We design our QoS multicast routing protocol based on the following objectives.

Favorable Tradeoff: Like many other network functions, QoS routing has multiple performance metrics, typically, *success ratio* and *overhead*, which represent the average probability of finding a feasible path and the average consumption of network resources (e.g., bandwidth, memory, and CPU), respectively. These performance metrics often conflict with each other — it normally takes more overhead to achieve better success ratio. There are two extremes in designing a QoS routing protocol: (1) searching only one path to optimize the overhead while sacrificing the success ratio, and (2) searching all paths to optimize the success ratio at the cost of heavy overhead. Many existing protocols [CC97, FBP98, CNS00] chose a tradeoff between the two extremes: searching multiple but not all paths. The rational behind these protocols is that sub-optimal success ratio can be achieved by searching a set of carefully selected paths. The key to the performance is how to make the path selection. A good path-selection strategy achieves *favorable tradeoff* — improving success probability significantly at an insignificant overhead increase. As we will see in Section 5, many existing routing protocols do not always achieve favorable tradeoff. Our new protocol achieves favorable tradeoff (good success ratio and low overhead) in all network conditions.

Adaptivity: The design of many QoS routing protocols is geared towards *congestion* conditions where the total resource requirement by all competing QoS traffic exceeds the resource quota for reservation. These protocols employ extra operations (overhead) in order

to find a good way to lay out the routing paths so that more connections can be accommodated. However, both spanning join and QoSMIC perform in the same way for any network conditions. The often-large extra overhead for congestion condition becomes a waste under normal traffic conditions, where networks are designed to operate for most of the time.[1]

Therefore, an adaptive routing algorithm is desired. It should be able to detect the network congestion condition and introduce extra overhead only when necessary. In addition, the amount of extra overhead should also be adaptive according to the traffic condition in the area where the routing takes place.

Scalability The spanning-joins protocol has large overhead for small multicast groups and QoSMIC has large overhead for large multicast groups [CNS00]. Though QMRP achieves lower communication overhead than spanning joins and QoSMIC, it requires extra memory space at the routers to keep the temporary search trees. That is per-group per-member information[2]. For large networks, when there are many simultaneous join requests, the transient memory demand on the routers may be high, which will degrade the performance of the routers. Recall that, historically, less memory consumption is one of the major advantages of core-based multicast trees against sender-based multicast trees.

Our new protocol does not construct any search tree. It does not require the routers to maintain any information other than the multicast tree itself. It thus uses less memory and achieve better simplicity.

4.2 A new Qos multicast routing protocol (S-QMRP)

We describe the basic idea of our new QoS multicast routing protocol (S-QMRP) in this section. The pseudo code of the protocol is given in Section 4.3. Optimization of the basic protocol is discussed in Section 4.4, which is crucial in reducing the overhead of S-QMRP.

4.2.1 Protocol overview

S-QMRP consists of two phases. The first phase is similar to SPR, in which a JOIN message is sent from the new member t to the root r along the unicast routing path. The JOIN message accumulates the path it traverses. It also accumulates the delay of the path in the reverse direction. When the JOIN message reaches an on-tree node i, if the accumulated delay plus the in-tree delay from r to i does not violate the delay requirement, a feasible tree branch is detected, which is the traversed unicast path. A CONSTRUCTION message is sent back along this path to construct a tree branch connecting the new member. The routing is successful and the second phase will not be activated.

On the other hand, if the delay requirement is violated at i, the JOIN message continues travelling to the root r. When the root receives the message, it starts the second phase, which employs multi-path routing. The root sends GROW messages to its neighbors. These

[1] Even when QoS traffic is light, best-effort traffic may be heavy [CN98]. In order to improve the performance of best-effort traffic, the excess overhead by the QoS routing protocol is always undesired.

[2] The routing for *each new member* of *a multicast group* forms a temporary search tree

GROW messages will then travel along the unicast routing paths towards the new member. As they travel, they try to construct new tree branches hop by hop along the way. Each GROW message carries the delay requirement D. It also accumulates the delay of the constructed tree branch. Hence, when an intermediate node i receives GROW, it knows the in-tree delay from r to i, $delay(P_{r,i})$.

Now, let us describe the actions that i will take after receiving GROW. First, i does an EW (Early Warning) test to see how likely the proceeding unicast path will satisfy the delay requirement D. Let j be the next hop on the unicast path. The EW test accepts four parameters, D, $delay(P_{r,i})$, $delay(i,j)$, and l, which is the length of the unicast path from i to the new member t. If the EW test passes, routing continues along the unicast path towards t; Otherwise, i attempts multi-path routing which may result in multiple downstream paths to be constructed. In this paper, the EW test is defined as follows.

if $delay(i,j) > (D - delay(P_{r,i}))/l$
then warning
else pass

$D - delay(P_{r,i})$ is the remaining slack of the delay requirement that further tree construction is allowed to have. $(D - delay(P_{r,i}))/l$ is the "fair share" of this slack for each link on the path from i to t. The above EW test states that if the delay of the link is larger than the fair share, a warning should be triggered; otherwise, the test is passed. More sophisticated EW test can be used, but are not considered here. In our simulation, the above EW test worked well.

If the EW test is passed, which means that the proceeding path is likely to satisfy the QoS requirement, i adds link (i,j) into the multicast tree and forwards the GROW message to the next hop j. If every intermediate node passes the EW test, a feasible branch is established for the new member.

However, if the EW test warns that the proceeding path may violate the QoS requirement, extra effort needs to be taken. Searching multiple downstream paths will increase the chance of success. Namely, the tree construction needs to *branch out*. We call i a *branching point*. GROW messages are sent to a subset of adjacent nodes x that satisfy the following *QoS test*:

if $delay(i,x) > D - delay(P_{r,i})$
then fail
else pass

Apparently, x can be the node j that just failed the EW test, but x should not be the adjacent node from which the GROW was previously sent to i. For the purpose of overhead reduction, we also might select only some of the nodes that pass the QoS test (see section 4.4).

If the QoS test is failed for every adjacent node, a BREAK message is sent back to trim the partially constructed tree branch. When a node k receives a BREAK message from a node i, it first deletes link (k,i) from the multicast tree, and then if k becomes a leaf node

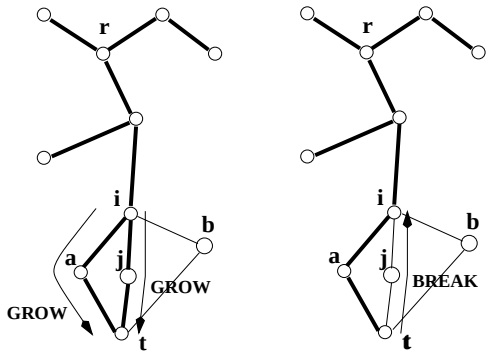

Figure 1: GROW and BREAK

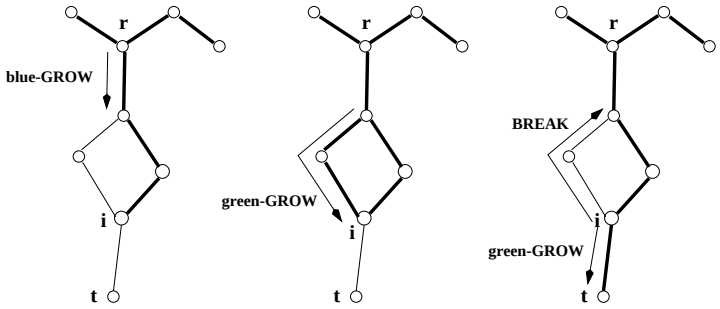

Figure 2: Break loops

and is not a member of the multicast group, it will delete itself from the multicast tree and propagate the BREAK message to its parent node. As BREAK travels back to r, the new tree branch is deleted.

There is a difference between the EW test and the QoS test. The EW test tries to make early guess on whether the proceeding path is likely to satisfy the delay constraint. If it sees signs of trouble, it generates warnings and triggers more branches to be constructed to improve the chance of success. The QoS test is to check if the delay constraint has already been violated. If it is, no further construction will proceed towards this direction.

At the beginning of the second routing phase, our protocol requires the root to be a mandatory branching point (an EW test is not necessary). Our simulation consistently shows that S-QMRP performs better in this way. The reason is that an early branching like this widens the search range and gives the subsequent tree construction more flexibility. Note, however, that the second phase commences only if the first phase fails.

Whenever a GROW message reaches t, a feasible tree branch is successfully established. However, t may receive multiple GROW messages from different branches. Fig. 1 gives an example. The multicast tree is shown by bold lines. i is an branching point, from which two branches reach t. In this case, t needs to send back BREAK messages to tear down all but the best branch. t can use the information (delay, bottleneck bandwidth, etc.) collected by the received GROW messages to determine which branch should be kept.

4.2.2 Breaking loops

As tree branches are constructed towards the new member, loops may form in the multicast tree. Fig. 1 is one example. Before we give a general solution to the looping problem, we need to study GROW messages more closely. Consider a GROW message that constructs a tree branch along a unicast path P to the new member. Some of the links on P may be already in the multicast tree while the others are not. When a GROW message travels along a link that is already in the multicast tree, we assign a color of *blue* to the GROW message. When a GROW message travels along a link that is not in the multicast tree, we assign *green* to the message.[3] Only green-GROW messages may form loops, because green-GROW messages join new links to the tree while blue-GROW messages follow existing tree links.

Using the coloring scheme, loop detection is easy. When an on-tree node receives a green-GROW message, a loop is formed. Fig. 2 gives an example. When the on-tree node i receives a green-GROW message, a loop is detected. A BREAK message is sent back to break the loop, while the GROW message continue constructing a tree branch towards the new member.

The sender of a GROW message can determined the color of the message as follows. When a node i sends a GROW to an adjacent node j, if j is the parent or a child of i in the multicast tree, i marks the GROW to be blue; otherwise, it marks the GROW to be green.

Since GROW messages can travel backward away from the new member at branching points, it is possible that a GROW message may travel along a loop in the network forever. This problem is solved in Section 4.4, where the number of branching points a GROW message may encounter is limited (MBL) and a GROW message is allowed to travel backward only once (Directivity).

4.2.3 Concurrent joins

S-QMRP can efficiently support concurrent join requests. Fig. 3 <1> gives an example, in which each GROW message is identified by the group address G and the new member id. Two concurrent new members, t and y, join the tree at the same time. Two GROW messages independently construct two new branches, which are partially overlapped. The messages contain different new member ids, and thus the branches grow towards different directions.

Fig. 3 <2> gives an abnormal case. Suppose the green-GROW message for y arrives at j first. It finds that link (j, y) will violate the delay requirement. It sends a BREAK message back to i to tear down the partially constructed tree branch. While the BREAK message is on the way to i, i sends a blue-GROW message for t down the same link. When j receives the blue-GROW, link (i, j) no longer belongs to the multicast tree and thus j should not construct a tree branch further towards t. Therefore, when a node (j) receives a blue-GROW message and finds that it is no longer an on-tree node, it should discard the blue-GROW message without further constructing tree branch.

[3]The green-GROW message will join this link to the multicast tree.

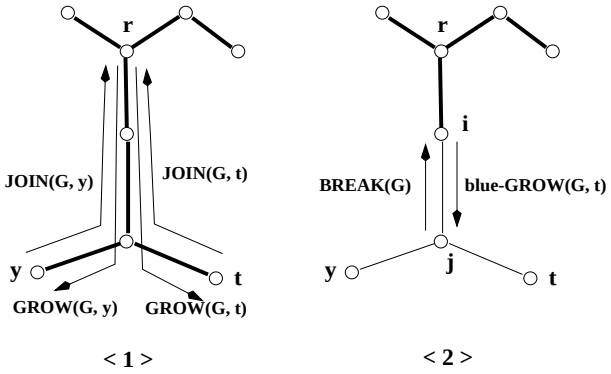

Figure 3: Concurrent joins

4.3 Pseudo code

The first phase of S-QMRP is quite straight forward thus we present only the pseduo code of the second phase of S-QMRP. Two types of routing messages are of concern.

1. A GROW message grows a new tree branch to connect the new member.
2. A BREAK message tears down a tree branch in order to break a loop in the multicast tree.

Both messages carry the multicast group address. GROW messages also carry the new member id, the delay bound D, the in-tree delay from the root to the current node, and the message color.

Each node in the multicast tree keeps a *multicast entry*, which is denoted as $M\{G, in, out\}$, where $M.G$ is the address of the multicast group, $M.in$ is the parent node in the multicast tree, and $M.out$ is the set of child nodes. The next hop on the unicast routing path from node i to node r (t) is denoted as $N_{i\to r}$ ($N_{i\to t}$). Given any node i, our protocol is implemented by the following pseudo code. Suppose i received a control message from k.

Node i:

switch (the received message)
case GROW(G, t, ...):
  if (it is a green-GROW)
    if (i is an on-tree node)
      /* break loop in the tree */
      send BREAK(G) back to k
    else
      /* add i into the tree */
      create a multicast entry $M(G, out, in)$
      $M.in = k$
      $M.out = \emptyset$

```
    if (i = t)
        routing is successful
    else
        /* further grow tree branch to next hop */
        if (N_{i→t} ∈ M.out or N_{i→t} = M.in)
            send blue-GROW(G, t, ...) to N_{i→t}
        else if (the EW test is passed)
            M.out := M.out + {N_{i→t}}
            send green-GROW(G, t, ...) to N_{i→t}
        else
            for every adjacent node j, j ≠ k
                if (j ∈ M.out or j = M.in)
                    send blue-GROW(G, t, ...) to j
                else if (j passes the QoS test)
                    M.out := M.out + {j}
                    send green-GROW(G, t, ...) to j
            if (no GROW was sent by the for loop,
                M.out = ∅, and i is not a group member)
                /* tear down partially constructed branch */
                remove the multicast entry M(G, out, in)
                send BREAK(G) to k
case BREAK(G):
    M.out := M.out − {k}
    if (M.out = ∅ and i is not a group member)
        /* remove i from the multicast tree */
        remove the multicast entry M(G, out, in)
        send BREAK(G) to M.in
```

Unlike QMRP [CNS00], our protocol does not require any node to store intermediate state information (search trees) about the routing process of every new member. A node needs only to keep the multicast entries of the multicast groups it is currently participating in.

4.4 Optimization

Whenever the EW test generates a warning at an intermediate node, the node becomes a branching point and multiple tree branches may grow out from this node towards the new member.[4] The number of branching points, if not restricted, can potentially be large, which will result in large routing overhead. We define two protocol parameters that are used to restrict the number of constructed tree branches.

[4]The BREAK messages will cut all but one branch. Hence, there will be only one tree branch connecting the new member eventually.

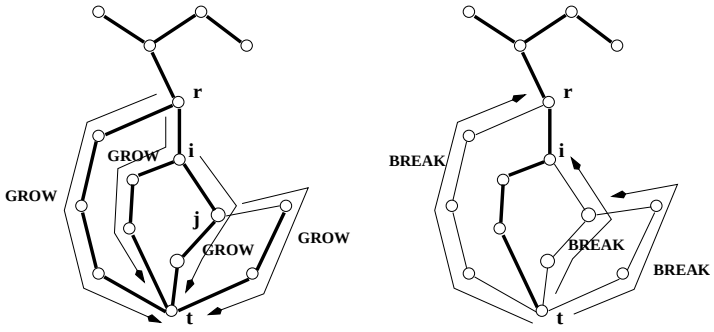

Figure 4: an example of S-QMRP-3

Maximum Branching Level (MBL): An easy way to control the number of branching points is to maintain an assertion: between the root and any on-tree node, there are at most m branching points. In other words, the maximum number of branching points is bounded by $\sum_{i=0}^{m-1}(d-1)^{i-1} = \frac{(d-1)^m-1}{d-2}$, where d is the maximum degree of a node. Such a restricted version of S-QMRP is denoted as S-QMRP-m. This number m is called the *maximum branching level*. An illustration of S-QMRP-3 is given in Fig. 4. Our simulations show that (1) S-QMRP-3 scales well because of its limited overhead, and (2) it achieves better routing performance than previous protocols such as QoSMIC and spanning joins. S-QMRP-m can be easily implemented by augmenting the GROW messages with a counter.

Directivity can be implemented to discourage tree branches growing away from the new member t. When a GROW is sent from i to j, if the distance from j to t is not shorter than the distance from i to t, the counter for MBL is set to zero, indicating that there is no branching point allowed for this GROW message.

Maximum Branching Degree (MBD): A branching point may have a large number of adjacent links, which can also cause excessive overhead. S-DMRP-m can be further argumented with an additional parameter, *maximum branching degree*, which specifies the maximum number of GROW messages that are allowed to be sent by a branching point. If the maximum branching degree x is smaller than the node degree minus one,[5] the node selects x outgoing links (based on distance to the new member or randomly) from which GROW has not been received, and sends GROW messages out along these links. We found in our simulations that a modest maximum branching degree (MBD) is sufficient. A too-large MBD does not add much to the routing performance but may add much overhead. An MBD of 5 worked very well in our simulation.

We suggest both MDL and MBD to be implemented. With MDL = m and MBD = x, the maximum number of branching points is $\Sigma_{i=0}^{m-1}x^i = \frac{x^m-1}{x-1}$. Therefore, the overhead can be controlled by these two parameters.

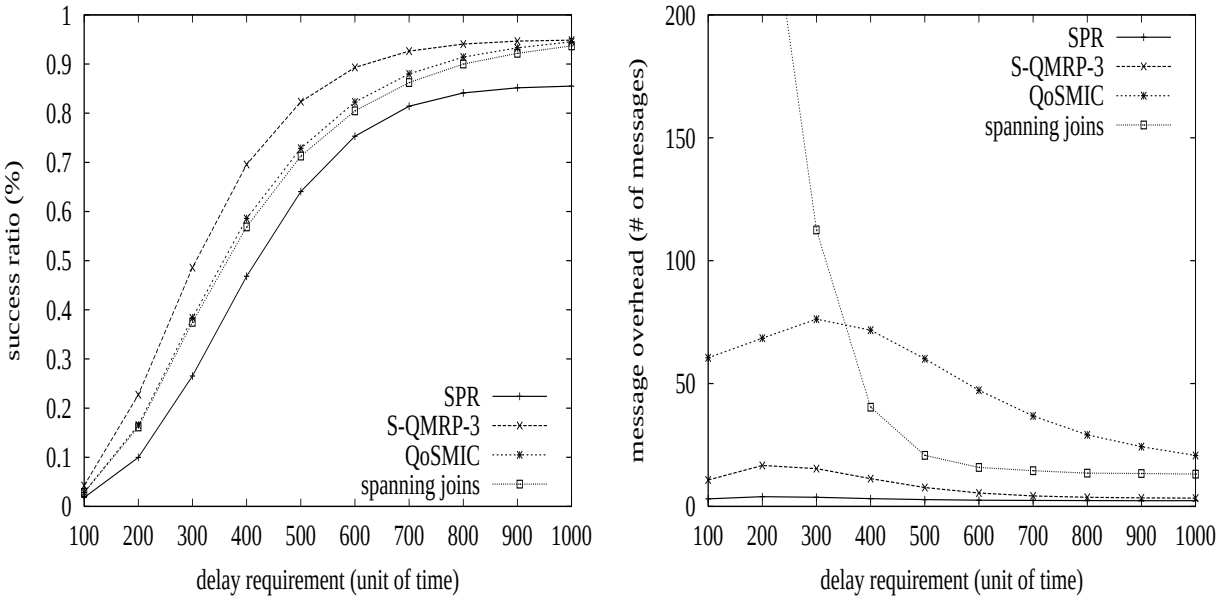

Figure 5: Power-Law topology, 600 nodes, 5% links saturated

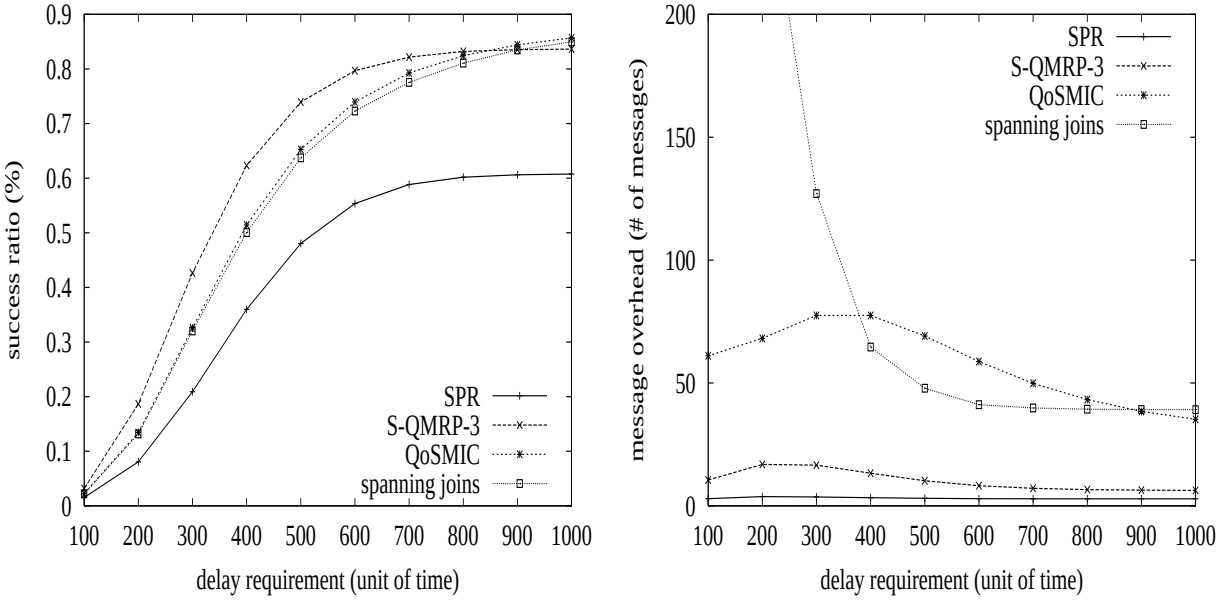

Figure 6: Power-Law topology, 600 nodes, 15% links saturated

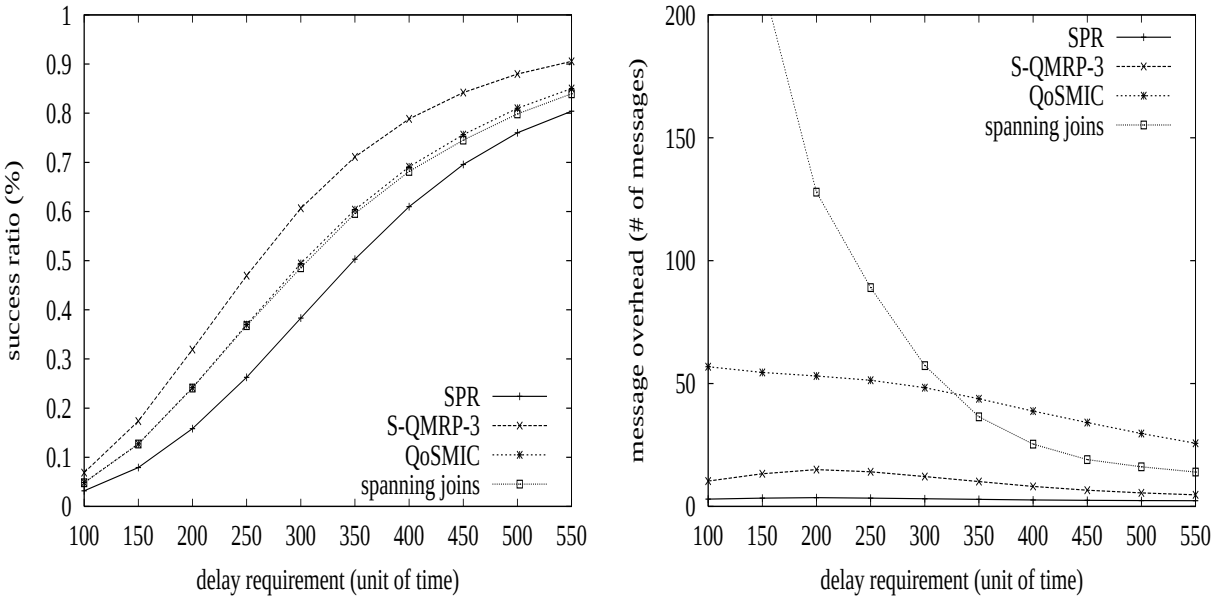

Figure 7: Power-Law topology, 300 nodes, 5% links saturated

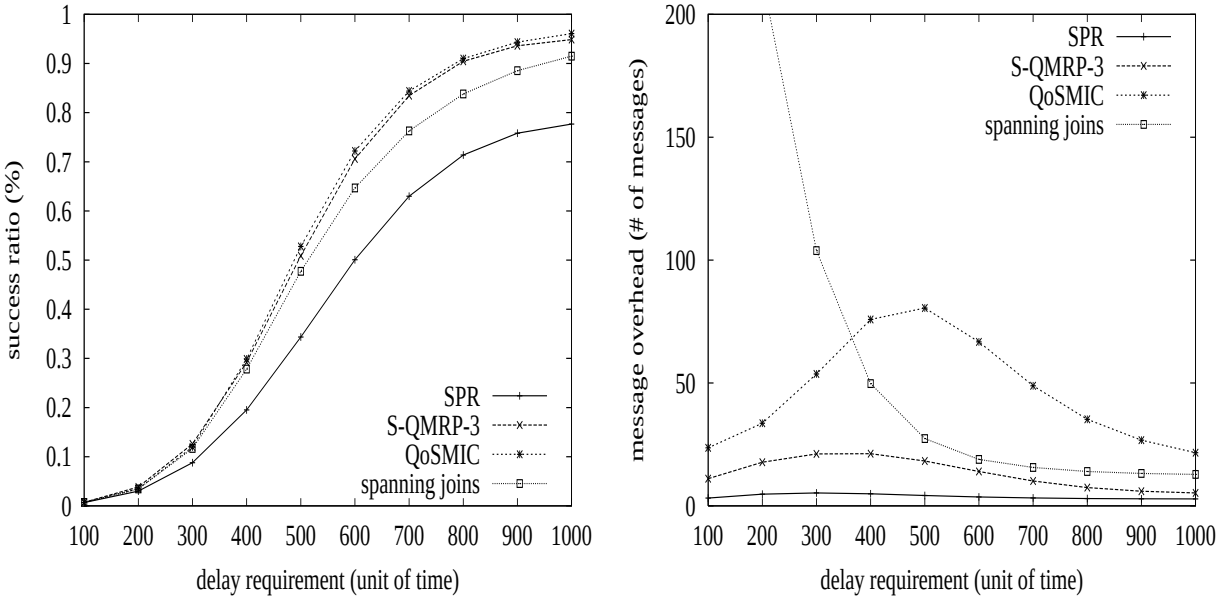

Figure 8: Waxman topology, 600 nodes, 5% links saturated

5 Simulation

Extensive simulations were conducted to study the performance of S-QMRP. Two performance metrics, *success ratio* and *average message overhead*, are defined as follows.

$$\text{success ratio} = \frac{\text{number of new members accepted}}{\text{total number of join requests}}$$

$$\text{avg. msg. overhead} = \frac{\text{total number of messages sent}}{\text{total number of join requests}}$$

When the message overhead is calculated, sending a message over a path of l hops is counted as l messages.

Four multicast routing protocols were simulated: *SPR*, *S-QMRP-3*, *QoSMIC*, and *spanning-joins*. The maximum branching degree of S-QMRP-3 is 5, i.e., a branching point can send at most 5 GROW messages to its neighbors. For QoSMIC, the local search and the tree search are implemented as sequential procedures; the tree search is executed only when the local search fails. Comparing with the parallel implementation of these two search procedures, the sequential implmentation minimizes the overhead, but may introduce additional delay. *Directivity*, *local minima*, and *fractional choice* [FBP98] were also implemented. For spanning joins, we implemented its directed flooding version, called *directed spanning joins* [CC97]. We assume a unicast routing protocol providing the shortest path in term of hops between each pair of nodes.

We shall not compare S-QMRP with QMRP [CNS00] because S-QMRP is mainly designed for additive metrics such as delay whereas QMRP does not support additive metrics. In addition, the advantage of S-QMRP over QMRP is neither success ratio nor message overhead but that it does not maintain any additional state other than the multicast tree itself, while QMRP needs to maintain the temporary per-member search trees.

Our simulations were conducted on Power-Law network topologies [FFF99] and Waxman network topologies [Wax88]. The Power-Law topologies are based on the results reported in [FFF99], which showed that the node degrees in the Internet obey a power log law: most nodes have small degrees and a small number of nodes have large degrees; as the degree increases, the number of nodes with that degree decreases exponentially. We used a topology generator described in [JJJ+00]. For the Waxman based topologies we used the Waxman method [Wax88] that spreads nodes randomly on a grid and adds links randomly, such that the probability of a link to be included decreases exponentially with its length.

In the simulation, a small percentage of links in the topology are randomly selected as *saturated links*, which refuse to accept more QoS traffic due to the lack of resources. The delays of these links are thus considered to be infinite. The delays of the rest links are uniformly distributed in the range of [0, 200] units of time.

In each simulation run, the link delays are first randomly generated, the root of the tree is randomly selected, and a delay requirement for the multicast tree is set. Then, the nodes

[5]The node should not send GROW to a link from which a GROW message has been received previously.

in the network start to join the tree in a random order; each node attempts once. Upon completion, the next simulation run starts. Two hundred simulation runs are conducted on each of six randomly generated topologies. The average result (success ratio/message overhead) of all simulation runs yields one data point in the figure. The standard deviation is less than 4% of the value of the data point.

Fig. 5 compare the success ratio and the message overhead of the four routing protocols. The horizontal axis represents different delay requirements of the multicast trees. Power-Law topologies with 600 nodes are used. Five percent of links are saturated links. The figure shows that the success ratio of S-QMRP-3 is better than those of QoSMIC and spanning joins. Remarkably, S-QMRP-3 achieves better success ratio at much lower message overhead, as shown in the right plot. When the delay requirement is small (i.e., 100), the spanning joins protocol has very large overhead (more than 600 messages per join request). That is because the multicast tree is always small and most join requests result in large scale flooding. Although the overhead of S-QMRP-3 is higher than that of SPR, it worth mentioning that for join requests SPR is able to find feasible paths, S-QMRP-3 behaves just like SPR and thus has the same overhead. Only for join requests SPR is unable to find feasible paths, S-QMRP-3 sends more control messages.

The reasons for S-QMRP-3 achieving low overhead and high success ratio are as follows: (1) it becomes SPR if the shortest path succeeds, (2) it branches out to search more paths only when needed, (3) it branches at the "right" places where the EW test fails, and (4) it allows only limited number of branching points.

We repeated the above simulation with different percentage of saturated links (Fig. 6), different topology size (Fig. 7), and different type of topology (Fig. 8). Similar results were always observed. In Fig. 6, 600-nodes Power-Law topologies are used with 15% of saturated links. In Fig. 7, 300-nodes Power-Law topologies are used with 5% of saturated links. In Fig. 8, 600-nodes Waxman topologies are used with an average node degree of 3.5. In all our simulations, the success ratio of S-QMRP-3 is better than or comparable to those of QoSMIC and spanning joins, and the overhead of S-QMRP-3 is lower. For small or medium delay requirements, the overhead of S-QMRP-3 is often significantly lower than those of QoSMIC and spanning joins.

6 Conclusion

We presented S-QMRP, a new QoS multicast routing protocol that has a favorable tradeoff between the communication overhead and the success probability. It was shown that the protocol overhead is much lower then previously suggested protocols, spanning join and QoSMIC, while its success probability is higher in most cases than other protocols (In some cases QoSMIC has comparable success probability but with significantly higher overhead). The protocol maintains no state in the network and works with both additive and non-additive QoS requirements.

Though the protocol is demonstrated to perform very good, we believe that it can be farther improved. In particular, on needs to study better EW functions, and better selection

procedures for next hop neighbors when the EW test throws a flag. In particular, there is no reason to use the entire 'branching budget' if the next hop to the neighbor seems too expensive.

References

[BFC93] T. Ballardie, P. Francis, and J. Crowcroft. An Architecture for Scalable Inter-domain Multicast Routing. *ACM SIGCOMM*, pages 85–95, September 1993.

[CC97] K. Carlberg and J. Crowcroft. Building Shared Trees Using a One-to-Many Joining Mechanism. *Computer Communication Review*, pages 5–11, January 1997.

[CN98] S. Chen and K. Nahrstedt. An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions. *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, Nov./Dec. 1998.

[CN99] S. Chen and K. Nahrstedt. Distributed QoS Routing in Ad-Hoc Networks. *IEEE JSAC, special issue on Ad-Hoc Networks*, Aug. 1999.

[CNS00] S. Chen, K. Nahrstedt, and Y. Shavitt. A QoS-Aware Multicast Routing Protocol. *IEEE INFOCOM 2000*, May 2000. Accepted to JSAC special issue on QoS in the Internet.

[CRS97] Israel Cidon, Raphael Rom, and Yuval Shavitt. Multi-Path Routing Combined with Resource Reservation. *IEEE INFOCOM'97*, pages 92 – 100, April 1997.

[Dee89] S. Deering. Host extensions for IP multicasting, August 1989. Internet RFC 1112.

[DEF+94] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. An Architecture for Wide-Area Multicast Routing. *ACM SIGCOMM*, pages 126–135, August 1994.

[EFH+98] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast-sparse mode (PIM-SM), June 1998. Internet RFC 2362.

[FBP98] M. Faloutsos, A. Banerjea, and R. Pankaj. QoSMIC: Quality of Service sensitive Multicast Internet protoCol. *SIGCOMM'98*, September 1998.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On Power-Law Relationships of the Internet Topology. *ACM SIGCOMM 1999*, Aug. 1999.

[FV90] D. Ferrari and D. C. Verma. A Scheme for Real-Time Channel Establishment in Wide-Area Networks. *IEEE JSAC*, 8(3):368–379, April 1990.

[HJ96] Handley and V. Jacobson. SDP: Session Directory Protocol (draft 2.1). *Internet Draft — Work in Progress*, February 1996.

[JJJ+00] Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, , and Lixia Zhang. On placement of internet instrumentation. In *IEEE INFOCOM 2000*, pages 295 – 304, Tel-Aviv, Israel, March 2000.

[Moy94] John Moy. Multicast extensions to OSPF, March 1994. Internet RFC 1584.

[Sha96] Yuval Shavitt. *Burst Control in High-Speed Networks.* PhD thesis, Technion – Israel Institute of Technology, Electrical Engineering Dept., Technion City, Haifa 32000, Israel, June 1996.

[THWC99] H.-Y. Tyan, J. Hou, B. Wang, and Y.-M. Chen. QoS Extension to the Core Based Tree Protocol. *NOSSDAV'99*, 1999.

[Wax88] B. M. Waxman. Routing of Multipoint Connections. *IEEE Journal of Selected Area in Communications*, pages 1617–1622, December 1988.

[ZDE+93] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, September 1993.

[Zha95] H. Zhang. Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10), October 1995.

[ZK91] H. Zhang and S. Keshav. Comparison of Rate-Based Service Disciplines. *ACM SIGCOMM'91*, 1991.