A Scalable Distributed QoS Multicast Routing Protocol

Shigang Chen, Yuval Shavitt

Abstract— **Many Internet multicast applications such as teleconferencing and remote diagnosis have Quality-of-Service (QoS) requirements. The requirements can be additive (end-to-end delay), multiplicative (loss rate) or with a bottleneck nature (bandwidth). With such diverse requirements, it is a challenging task to build QoS constrained multicast trees with high performance, high success ratio, low overhead, and low system requirements. This paper presents a new Scalable QoS Multicast Routing Protocol (S-QMRP) 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. 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. Its early-warning mechanism helps to detect and route around the real bottlenecks in the network, which increases the chance of finding feasible paths for additive QoS requirements. S-QMRP minimizes the system requirements; it relies only on the local state stored at each router. The routing operations are completed decentralized. The protocol does not require any global network state to be maintained and can operate on top of any unicast routing protocol.**

I. 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 [1]. 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 shortest path tree [2], [3]. This approach was adopted by the IETF in its multicast standardization effort, where shortest path trees are used as multicast trees: DVMRP [4], PIM [5], MOSPF [6].

Dr. Shigang Chen is with Internet Services Management Group, Cisco Systems, 170 W. Tasman Dr., San Jose, CA 95134, USA, sgchen@cisco.com

Dr. Yuval Shavitt is with the Department of Electrical Engineering - Systems, Tel-Aviv University, Tel-Aviv 69978, Israel. shavitt@ieee.org.

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 [7].

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 [8], [9], [10], [11], [12], [7].

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 [7] 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 probloms. Firstly, QMRP deposits temporary state in the routers for each join request. This hurts its scalability. It is highly desired that the routers only maintain per group information without any per-group-per-join information. Secondly, QMRP was designed for applications with non-additive QoS requirements (bottleneck requirements) such as bandwidth and buffer space. It lacks the mechanisms to handle additive/multiplicative QoS requirements such as delay or packet loss. QoS routing for additive/multiplicative requirements is a much more difficult problem than that for non-additive requirements. Therefore, it is thus a challenging problem by itself to extend QMRP to work for all kinds of requirements while retaining all the good merits. While spanning join [9] and QoSMIC [11] do not have the above problems, they have much higher overhead and lower success probability [7]. Hence, a further study for a scalable, efficient QoS multicast routing protocol that can handle all three types of QoS requirements is under call.

This paper suggests a QoS multicast routing protocol, S-QMRP, that shares the adaptive path-branching idea of QMRP, but eliminates the use of per-group-per-join routing state. In QMRP, each new member initiates a search tree, which grows towards the multicast tree. The search tree is per join state information. In S-QMRP, the multicast tree grows towards the new members.[1] The protocol does not require to store any extra routing state other than the multicast tree itself. It not only gets rid of per join routing state but also allows the dynamic aggregation of multiple join requests, where a single tree branch may grow toward multiple new members. By eliminating the search tree, S-QMRP removes the state machine in QMRP that governs the construction of the search tree, and therefore simplifies the implementation.

During the search for a feasible path, S-QMRP may branch out to search multiple paths just as QMRP. The key difference that allows S-QMRP to support additive requirements is the way it selects the locations where path branching occurs. Such locations are called *branching points*. QMRP searches a single path from a new member to the tree until it hits a link that does not have the required bandwidth. The invalid link triggers QMRP to branch out in order to get around of the link and to search multiple downstream paths along other directions. For additive/mulplicative requirements, however, the appropriate branching points may not be the locations where the requirements are violated. It is often because of an early link having too large delay. That link instead of the last link on the searched path should be excluded. In order to find the right branching points, S-QMRP uses a novel early-warning (EW) mechanism which finds the real bottleneck in the network and helps to detour around the bottleneck. For non-additive requirements such as bandwidth, the early-warning mechanism provides branching points similarly as in QMRP. However, for additive/multiplicative requirements such as delay, it tends to branch out earlier on the searched path when an unfavorable link is encountered. By doing this, S-QMRP significantly increases the chance of finding a feasible path for a new member.

[1] It is problematic to grow tree branches independently from the new members towards the tree, which effectively results in a partitioned tree at run time. The temporarily partitioned tree is OK for traditional multicast routing [5], but is undesired for QoS routing. That is because a growing tree branch may withdraw before joining the multicast tree due to the violation of QoS requirement, which does not happen in traditional best-effort routing.

The rest of the paper is organized as follows. Section II presents our network model. Section III describes the routing protocol. Analysis and simulation results are provided in Section IV and Section V, respectively. Section VI draws the conclusion.

II. 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 and OSPF 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 [13], [14] and what packet scheduling algorithms are used [15], [16] 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 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* and *core-based trees*. 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. A delay-constrained core-based tree can be modeled as two subtrees. One is from the core to all receivers, and the other is from all senders to the core. Given a delay requirement D, if both subtrees are bounded by $D/2$, then the delay requirement is met. Constructing the first subtree is the same problem as constructing a sender-based tree, while constructing the second subtree is a reverse problem. To simplify the presentation, we will use sender-based trees to illustrate the protocol in the rest of the paper. Our solution with modifications is also applicable to the core-based tree by constructing the two subtrees with half of the delay requirement.

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 [17].

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 $delay(i, j)$.

III. 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.

A. 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 [9], [11], [7] 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 V, 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.[2]

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.

[2]Even when QoS traffic is light, best-effort traffic may be heavy [18]. In order to improve the performance of best-effort traffic, the excess overhead by the QoS routing protocol is always undesired.

Scalability The spanning-joins protocol has large overhead for small multicast groups and QoSMIC has large overhead for large multicast groups [7]. 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[3]. 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.

B. 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 III-C. Optimization of the basic protocol is discussed in Section III-D, which is crucial in reducing the overhead of S-QMRP.

B.1 Protocol overview

S-QMRP consists of two phases. The first phase is similar to shortest path routing (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 k, if the accumulated delay plus the in-tree delay from r to k 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 k, 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 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

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

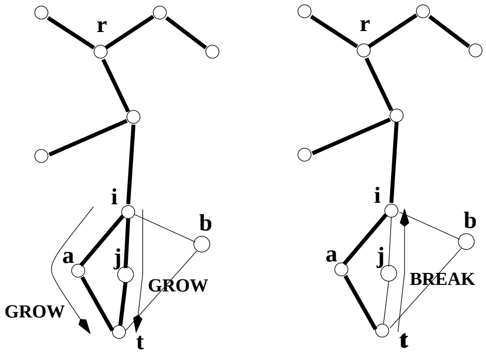

Fig. 1. GROW and BREAK

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 III-D).

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 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 triggers branching 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 simulations consistently show that S-QMRP performs better this way. The reason is that an early branching 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, i.e., the SPR path failed.

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 a 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.

Multiple GROW messages can grow multiple tree branches temporarily for the same new member. However, only one branch will remain and the other extra ones will be detected and pruned automatically. No timeout mechanism is used during the process. One important point is that, no matter how many simultaneous joins there are, each node keeps only the information about whether it is on the tree and does not keep any information about a particular join. Temporary per-join information was maintained in QMRP [7].

B.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.[4] 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.

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.

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.

Arriving at i, the GROW message has the in-tree delay of the new tree branch ($r \to a \to b \to i$). i knows the in-tree delay of the old tree branch ($r \to$

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

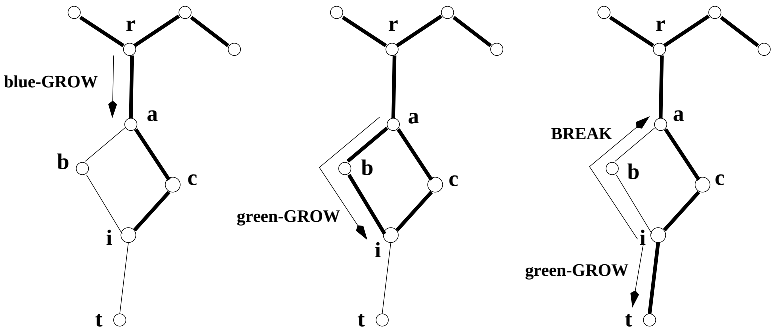

Fig. 2. Break loops

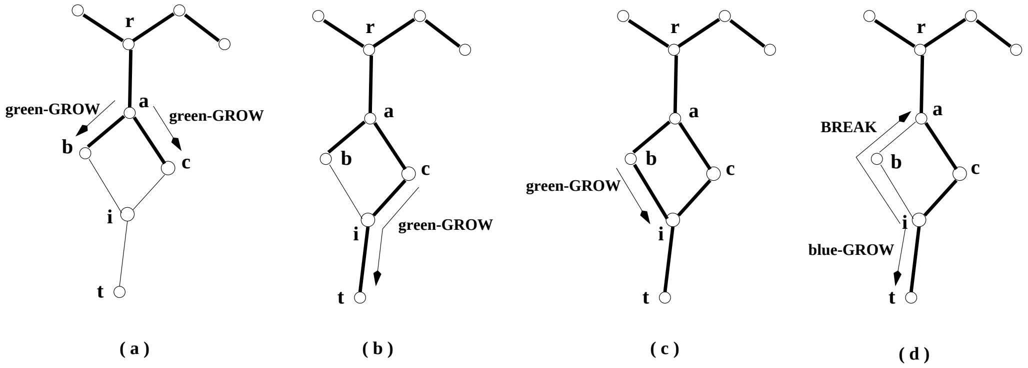

Fig. 3. Two growing tree branches meet

$a \to c \to i$). The BREAK message can be sent to tear down the new branch, in which case the in-tree delay in the GROW message needs to be updated to equal that of the old tree branch. Or the BREAK message can be sent to break the old branch based on certain optimization criteria (e.g., the in-tree delay of the new branch is smaller), and in this case the in-tree delay stored at i needs to be updated.

Two growing branches may meet at a common node to form a loop, which will be broken similarly. In Fig. 3 (a), two green-GROW messages grow two tree branches simultaneously. Suppose the green-GROW message from c arrives at i first in Fig. 3 (b). The message adds i to the tree and continues to travel towards the destination along unicast path. Then in Fig. 3 (c) the second green-GROW message arrives at i. This message triggers a BREAK message to break the loop and a blue-GROW message follows the same unicast path to the destination in Fig. 3 (c). The blue-GROW message is necessary because i does not keep per-join routing state to indicate that a previous green-GROW for the same destination has been sent. More specifically, i does not know if the downstream tree branch will actually reach the destination.

No blue-GROW messages but only green-GROW messages will cause BREAK messages.

B.3 Concurrent joins

S-QMRP can efficiently support concurrent join requests. Fig. 4 <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. 4 <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.

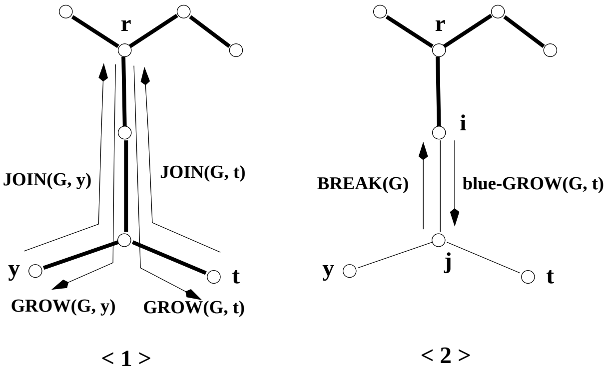

Fig. 4. Concurrent joins

C. Pseudo code

The first phase of S-QMRP is quite straight forward thus we present only the pseudo 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 = ∅
    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 [7], 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.

D. 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.[5] 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.
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 $\sum_{i=0}^{m-1}(d-1)^i$, where d is the maximum degree of a node. When $d = 2$, $\sum_{i=0}^{m-1}(d-1)^i = m$; when $d > 2$, $\sum_{i=0}^{m-1}(d-1)^i = \frac{(d-1)^m-1}{d-2}$. Such a restricted version of S-QMRP is denoted as S-QMRP-m. This number m is called the *maximum branching level*. An

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

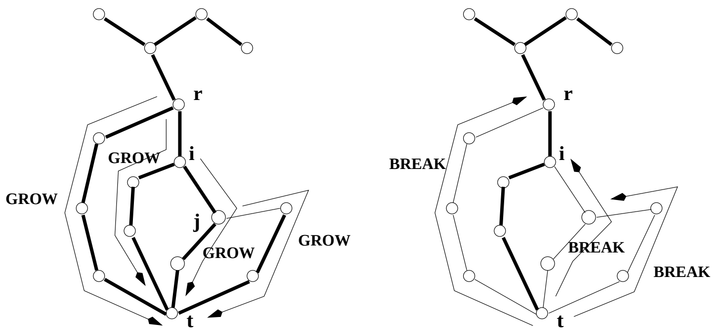

Fig. 5. an example of S-QMRP-3

illustration of S-QMRP-3 is given in Fig. 5. 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,[6] 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.

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

IV. Analysis

Theorem 1: S-QMRP never forms a persistent loop in the multicast tree.

Proof: Only green-GROW messages add new links to the tree and can form loops. Suppose j sends a green-GROW message to i, adds link (j, i), and forms a loop. Let k be the parent node of i in the tree. By the structure of trees, both link (j, i) and link (k, i) belong to the loop. According to Section III-B.2, a BREAK message will be sent to either j or k to tear down (j, i) or (k, i), which breaks the loop immediately. Hence, any loop can not persist. □

S-QMRP grows new tree branches from the tree towards new members. In many other protocols, new branches are constructed from new members towards the tree. PIM is an example. Before new branches actually reach the tree, the tree can be viewed as being partitioned. This is fine without QoS constraints because those growing branches will eventually reach the tree by following the unicast paths. However, with QoS constraints, it can cause a problem. When a growing branch reaches an on-tree node, the node may belong to another growing branch, which may be torn down later due to the violation of QoS constraints. This case never arises in S-QMRP.

Theorem 2: S-QMRP never partitions the tree.

Proof: New tree branches always grow from the tree. Hence, any new on-tree nodes must be connected with the tree. BREAK messages only remove leaf nodes from the tree and thus have no chance to partition the tree. □

The above two theorems can also be applied to S-QMRP-m for any m.

The distributed algorithm terminates when no control message of a join request is still in transit.

Theorem 3: S-QMRP-m terminate in finite time.

Proof: When a GROW message branches out to k GROW messages, we regard the parent message and the child messages as $k+1$ individual messages in this proof.

Consider an arbitrary join request. S-QMRP-m has at most $\sum_{i=0}^{m-1}(d-1)^i$ branching points, where d is the maximum degree of a node (Section III-D). Each branching point generates at most d GROW messages. Hence, the total number of GROW messages is finite. Every GROW message follows the unicast path to the destination and thus terminates in finite time. Each GROW message may or may not generate one BREAK message. Hence, there are finite BREAK messages. Every BREAK message follows a multicast tree branch backward to prune the branch and terminates when reaching a non-leaf node. There are one JOIN message and one CONSTRUCTION message. Both of them follow the unicast path and terminate in finite time. Because there are finite control messages that all terminate in finite time, S-QMRP-m terminates in finite time. □

In practice, S-QMRP with unlimited branching points should never be used.

S-QMRP-m prefers the shortest path provided by the underlying unicast routing protocol. However, when the shortest path does not satisfy the QoS requirement, the GROW messages are sent along other paths. Though it increases the chance of success, too long paths are often undesired due to larger consumption of resources. The following theorem shows that S-QMRP-m can make sure that the increase in path length is bounded.

Theorem 4: Suppose the unicast routing paths are the shortest paths in terms of hops. For S-QMRP-m which implements MBL, a tree branch from the root to a member is at most $2m$ hops longer than the shortest path. If the directivity is implemented, a tree branch is at most two hop longer than the shortest path.

Proof: When a tree branch is constructed, S-QMRP-m allows at most m branching points on the branch. A GROW message always travels along the shortest path unless at the branching points. At each branching point, GROW may travel one hop backward away from the new member, which increases two hops over the shortest path. There are at most m branching points, which can increase at most $2m$ hops over the shortest path. Hence, any constructed tree branch to a new member is no more than $2m$ hops longer than the shortest path.

When the directivity is implemented, according to Section III-D, if a GROW message does not follow a shortest path after a branching point, its MBL is set to be zero and no more branching is allowed. Therefore, a GROW message can travel backward only once. Hence, any constructed tree branch is at most two hops longer than the shortest path. □

In the following, we compare the worst case overhead of three protocols: spanning join, QoSMIC, and S-QMRP. To simplify the problem, we consider a network of n uniformly connected nodes. Let the diameter of the network be 2ω hops. Assume the number of nodes in the k-neighborhood of a node, $\mathcal{N}_k$, grows quadratically with k, i.e., $\mathcal{N}_k = \alpha k^2$. Thus, the diameter is given by $\alpha\omega^2 = n$. When the spanning join protocol broadcasts in a neighborhood with a radius of k hops, the number of messages sent is αk^2. Hence, in the worst case the total number of messages sent in consecutive broadcasts are

$$\sum_{k=1}^{\omega}(\alpha k^2) = \alpha\frac{\omega(\omega+1)(2\omega+1)}{6} \approx \frac{n(2\omega+1)}{6} \in O(n\omega)$$

The local search of QoSMIC broadcasts in a small neighborhood with a constant radius. The message overhead of this part can be considered as a constant. Let T be the size of the multicast tree. The worst case overhead of the tree search is $O(T)$. For a dense tree that populates the entire network, $O(T) = O(n)$.

Consider S-QMRP-m with MBD $= x$. The maximum number of branching points is $\frac{x^m-1}{x-1}$ (Section III-D). The maximum number of branches is $x\frac{x^m-1}{x-1}$. Note that x and m are both small constants. The length of any branch is bounded by $O(2\omega)$. Therefore, the total number of messages sent is bounded by $O(x\frac{x^m-1}{x-1}2\omega) = O(\omega)$ in the worst case. We shall do a more detailed study on overhead in Section V by simulation. What the above analysis tells us is that, as the network size increases, the worst case overhead of spanning join, QoSMIC, and S-QMRP increases in the order of $n\omega$, n, and ω, respectively. For a perfect uniformly-connected network with every node degree being d, $\omega = O(\sqrt[d/2]{n})$. Among the three, S-QMRP is the least sensitive to the size of the network, which means better scalability.

V. 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

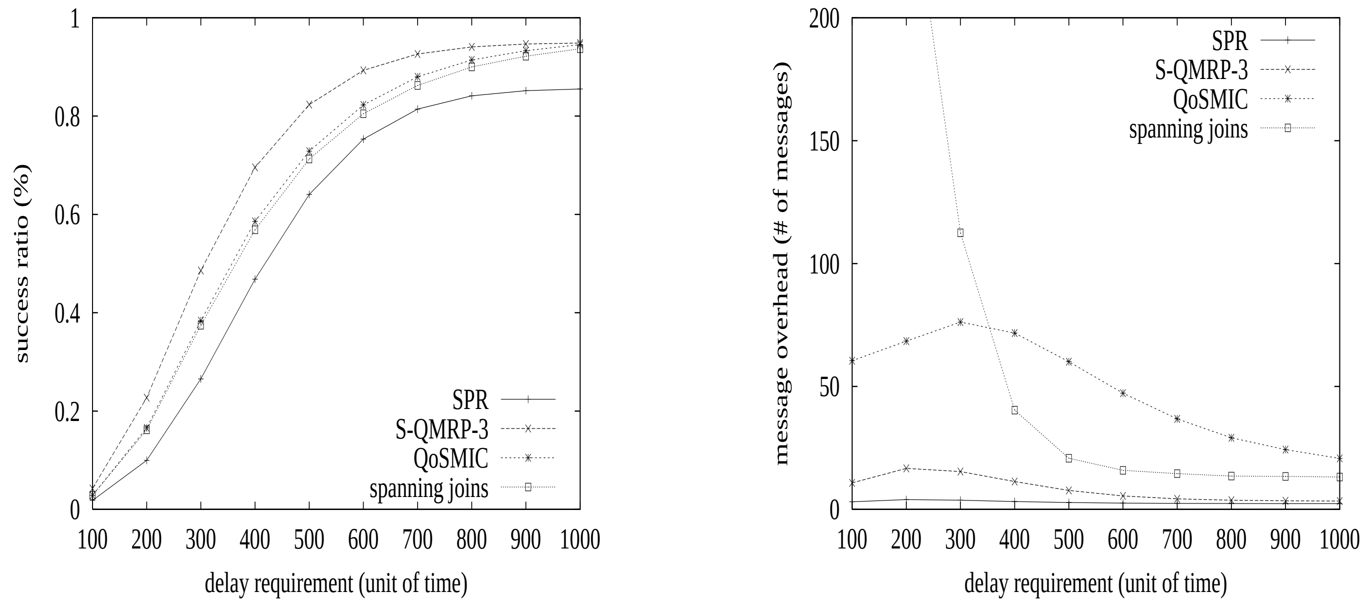

Fig. 6. Power-Law topology, 600 nodes, 5% links saturated

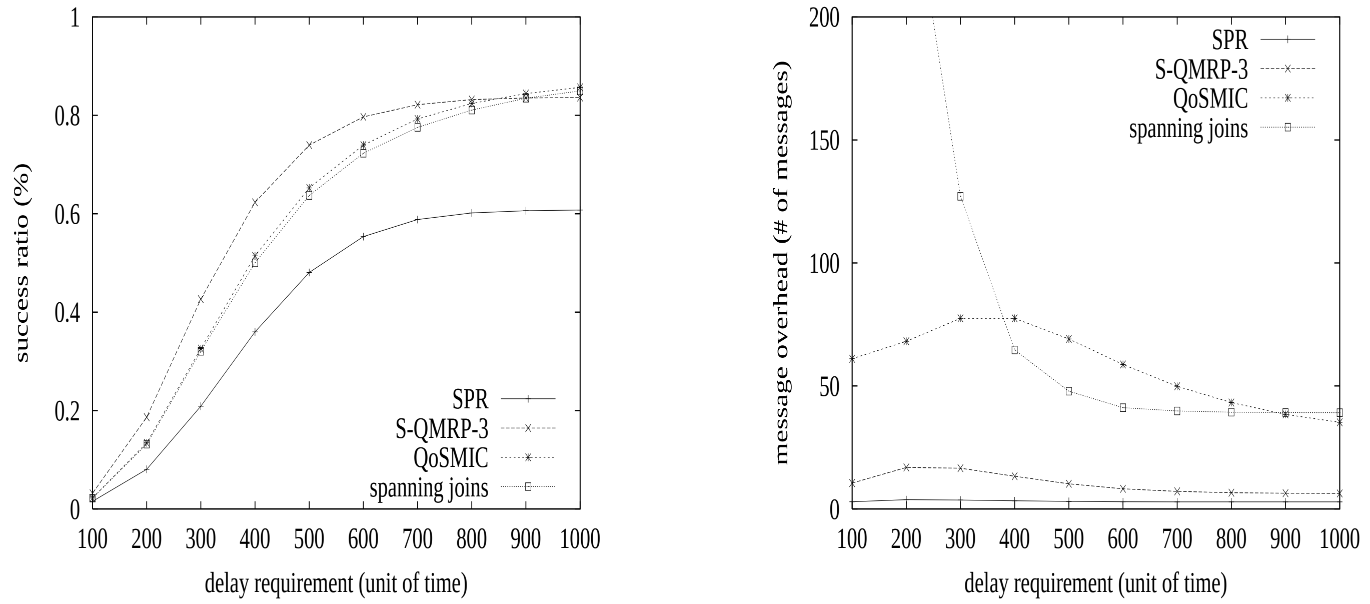

Fig. 7. Power-Law topology, 600 nodes, 15% links saturated

implementation minimizes the overhead, but may introduce additional delay. *Directivity*, *local minima*, and *fractional choice* [11] were also implemented. For spanning joins, we implemented its directed flooding version, called *directed spanning joins* [9]. We assume a unicast routing protocol providing the shortest path in term of hops between each pair of nodes.

The major advantage of S-QMRP over QMRP [7] is that S-QMRP does not maintain any per-group-per-join state information. Note that the multicast tree is per-group information and has to be there. QMRP needs to maintain the temporary search trees, which is per-group-per-join information. In this section, we do not compare S-QMRP with QMRP because S-QMRP is mainly designed for additive QoS requirements such as delay whereas QMRP was designed for non-additive QoS requirements such as bandwidth. If we do not consider the global resource efficiency but only consider satisfying a bandwidth requirement, all links that do not meet the requirement, called *infeasible links*, are equally bad, and all links that meet the requirement, called *feasible links*, are equally good.

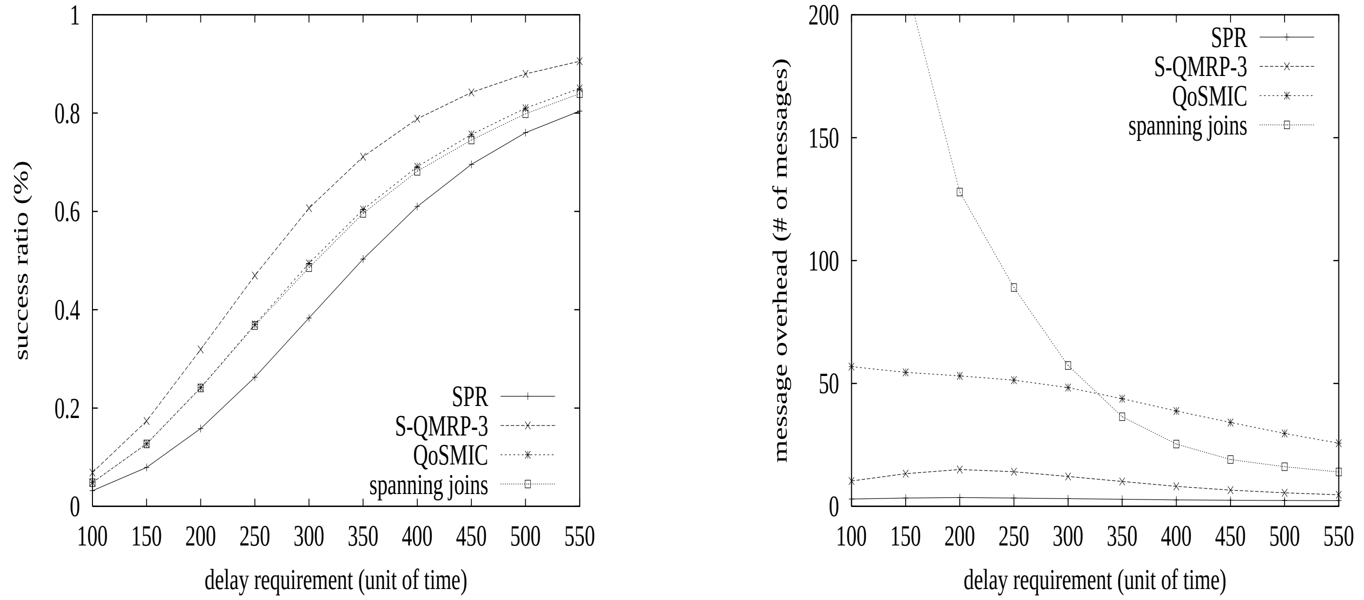

Fig. 8. Power-Law topology, 300 nodes, 5% links saturated

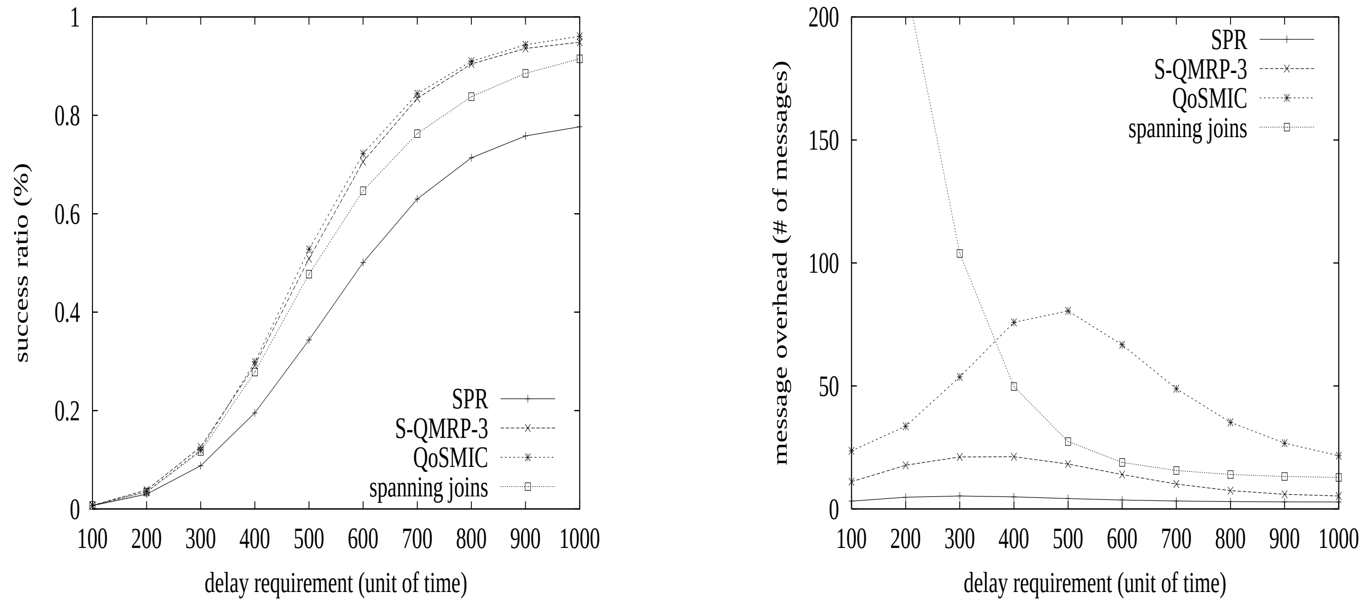

Fig. 9. Waxman topology, 600 nodes, 5% links saturated

Therefore, When the search tree of QMRP hits an infeasible link, it branches to exclude this link, and all the rest of the search tree is still good. However, for a delay requirement, links are not equally good or equally bad. For instance, after a link is added to the searched path, if the delay requirement is violated, it is better to exclude the bottleneck link on the path instead of the last added link, and that is what motivated us to introduce the EW test into S-QMRP. Instead of branching out at the very last minute, S-QMRP branches out at bottleneck links that have large delays causing the EW test to fail, which makes S-QMRP suitable for additive QoS requirements.

Our simulations were conducted on Power-Law network topologies [19] and Waxman network topologies [20]. The Power-Law topologies are based on the results reported in [19], 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 [21]. For the Waxman based topologies we used the Waxman method [20] 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.[7] When the unicast path has a link that is saturated for QoS traffic, SPR will fail but the other protocols may still succeed because they explore more paths than the unicast one. 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 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. 6 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.

[7] Each link typically has a "quota" on the maximum amount of resources allowed to be reserved for QoS traffic in order not to starve the best-effort traffic. Once this quota is reached, the link refuses to accept more QoS traffic. It is then a *saturated link*. Note that the delay of a saturated link is infinite for new QoS traffic but is not infinite for the best-effort traffic. While the underlying unicast routing algorithm works on the best-effort traffic, it may select saturated links on its routing paths.

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. 7), different topology size (Fig. 8), and different type of topology (Fig. 9). Similar results were always observed. In Fig. 7, 600-nodes Power-Law topologies are used with 15% of saturated links. In Fig. 8, 300-nodes Power-Law topologies are used with 5% of saturated links. In Fig. 9, 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.

VI. 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

[1] F. K. Hwang and D. S. Richards, "Steiner Tree Problems," *Networks*, vol. 22, pp. 55–89, 1992.

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

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

[4] S. Deering, "Host extensions for IP multicasting," Aug. 1989, Internet RFC 1112.

[5] 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.

[6] John Moy, "Multicast extensions to OSPF," Mar. 1994, Internet RFC 1584.

[7] S. Chen, K. Nahrstedt, and Y. Shavitt, "A QoS-Aware Multicast Routing Protocol," *IEEE JSAC*, vol. 18, no. 12, pp. 2580–2592, Dec. 2000.

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

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

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

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

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

[13] D. Ferrari and D. C. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks," *IEEE JSAC*, vol. 8, no. 3, pp. 368–379, April 1990.

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

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

[16] H. Zhang, "Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE*, vol. 83, no. 10, October 1995.

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

[18] 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.

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

[20] B. M. Waxman, "Routing of Multipoint Connections," *IEEE Journal of Selected Area in Communications*, pp. 1617–1622, Dec. 1988.

[21] Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, , and Lixia Zhang, "On placement of internet instrumentation," in *IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000, pp. 295 – 304.

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