# A QoS-Aware Multicast Routing Protocol

Shigang Chen
Internet Security Management Group
Cisco Systems
Champaign, IL 61820
sgchen@cisco.com

Klara Nahrstedt
Dept. of Computer Science
U. of Illinois at Urbana-Champaign
Urbana, IL 61801
klara@cs.uiuc.edu

Yuval Shavitt
Bell Labs, Lucent Technologies
101 Crawfords Corner Road
Holmdel, NJ 07733-3030
shavitt@ieee.org

*Abstract*— **The future Internet is expected to support multicast applications with quality of service (QoS) requirements. To facilitate this, QoS multicast routing protocols are pivotal 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.**

**We propose QMRP, a new Qos-aware Multicast Routing Protocol. QMRP achieves scalability by significantly reducing the communication overhead of constructing a multicast tree, yet it retains a high chance of success. This is achieved by switching between** *single-path routing* **and** *multiple-path routing* **according to the current network conditions. The high level design of QMRP makes it operable on top of any unicast routing algorithm in both intra-domain and inter-domain. Its responsiveness is improved by using a termination mechanism which detects the failure as well as the success of routing without the use of timeout. In addition, QMRP always constructs loop-free multicast trees.**

## I. Introduction

Multicast employs a tree structure in the network to efficiently deliver the same data stream to a group of receivers. Traditionally, the research on Internet multicasting has been centered on scalability and efficiency. The deployment of high-speed networks opens a new dimension of research, which is to provide *quality of service* (QoS) such as guaranteed throughput for audio/video streams. It is technically a challenging and complicated problem to deliver timely, smooth, synchronized multimedia information over a decentralized, shared network environment, especially one that was originally designed for best-effort traffic such as the Internet [1]. Some sort of resource reservation needs to be made so that the quality of data delivery can be ensured in the presence of dynamic background traffic. While a resource reservation protocol (e.g., RSVP [2]) addresses the problem of how to reserve resources on a multicast tree, it is the task of a routing protocol to find a tree which not only covers all members but also has the required resources on each of its routers. Such a tree is called a *feasible* tree.

The traditional multicast routing protocols, e.g., CBT [3] and PIM [4], were designed for best-effort data traffic. They construct multicast trees primarily based on connectivity. Such trees may be unsatisfactory when QoS is considered due to the lack of resources. Recently, several QoS multicast routing algorithms have been proposed to find feasible trees. Some algorithms [5], [6] provide heuristic solutions to the NP-complete *constrained Steiner tree problem*, which is to find the delay-constrained least-cost multicast trees. These algorithms however are not practical in the Internet environment because they have excessive computation overhead, require knowledge about the global network state, and do not handle dynamic group membership. The spanning join protocol by Carlberg and Crowcroft [7] handles dynamic membership and does not require any global network state. However, it has excessive communication overhead because it relies on flooding to find a feasible tree branch to connect a new member. QoSMIC [8], proposed by Faloutsos et al., alleviates but does not eliminate the flooding behavior. In addition, an extra control element, called Manager router, is introduced to handle the join requests of new members.

In this paper, we propose QMRP, a new QoS-aware multicast routing protocol, which achieves the following design goals.

- *Scalability:* Scalability is achieved by significantly reducing the overhead of constructing a multicast tree. QMRP switches between single-path routing and multiple-path routing according to the current network conditions, and incrementally adds additional paths into the searching process only when that is necessary. In many cases it behaves like PIM and searches only one path.
- *QoS awareness:* Minimizing the overhead and maximizing the chance of success are contradictive goals. We balance the two goals by making the routing process QoS-aware. Intuitively speaking, we "spend" the limited overhead wisely based on a careful path selection mechanism which combines the connectivity-based search and the resource-based search so that the routing messages are directed only along those paths which have the required resources.
- *Efficiency:* The protocol may detect multiple feasible tree branches for a new member. A novel distributed selection process is designed to select the best branch connecting the new member to the tree.
- *Robustness:* The protocol does not rely on any extra control element. The routing process is entirely decentralized.
- *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 at any router.
- *Responsiveness:* Many existing protocols such as QoSMIC use timeout to detect the failure when finding a feasible tree branch for a new member is not successful. In

This work was partially supported by the Airforce grant under contract number F30602-97-2-0121 and the National Science Foundation Career grant under contract number NSF CCR 96-23867.

order to accommodate the worst-case situation, the timeout interval has to be large, which makes the protocol less responsive. Our protocol provides a mechanism to detect the termination of the routing process whether it succeeds or fails. Hence, it improves the responsiveness by avoiding the use of timeout.

- *Loop free:* The protocol always constructs loop-free multicast trees.

The rest of the paper is organized as follows. Section II presents the related work. Section III describes the routing protocol. The analysis and simulation results are provided in Sections IV and V, respectively. Section VI draws the conclusion.

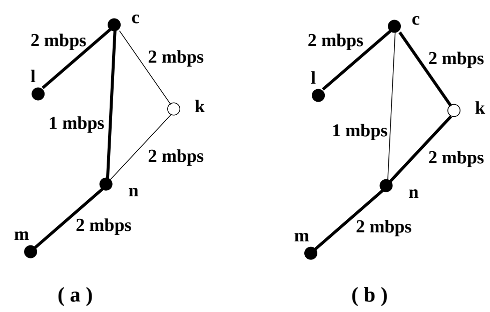


Fig. 1. The multicast tree is shown by bold lines.

## II. Related Work

A multicast tree is incrementally constructed as members leave and join a multicast group. When an existing member leaves the group, it sends a control message up the tree to prune the branch which has no members attached any more. When a new member joins the group, the tree must be extended to cover the new member. Based on how the new member is connected onto the tree, the multicast routing protocols can be classified into two broad categories: *single-path routing protocols* (SPR) and *multiple-path routing protocols* (MPR). An SPR protocol provides a single path connecting the new member to the tree, whereas an MPR algorithm provides multiple candidate paths to choose from.

### A. Single-path routing

Most SPR protocols were originally designed for the best-effort data traffic. We briefly discuss two representative protocols and point out why they are not suitable for QoS traffic.

CBT (Core-Based Tree) [3], [9] and PIM (Protocol Independent Multicast) [4] connect a new member $i$ to the multicast tree along the unicast routing path from $i$ to the root (core) of the tree. The unicast path is typically the shortest path in term of hops. The resulting shortest-path trees are good for best-effort traffic. However, when QoS is considered, such shortest-path trees may not have the resources to support the quality requirement. Fig. 1 gives an example. Suppose the underlying unicast routing protocol provides the shortest path in terms of hops between each pair of nodes. Applications have diverse QoS requirements. A multicast group may want a delivery tree with 2 Mbps bandwidth on every link. Given a core $c$ and the set of group members $\{l, m, n\}$, the shortest-path tree in Fig. 1 (a) violates the bandwidth requirement. While the tree in Fig. 1 (b) satisfies the requirement, CBT or PIM can not find this tree.

### B. Multiple-path routing

In order to increase the chance of finding a feasible tree, the MPR protocols provide multiple candidate paths for a new member to be connected to the tree. Among the candidates the new member selects the best one.

**Spanning-joins [7]:** In the spanning-joins protocol proposed by Carlberg and Crowcroft, 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 algorithm, is a candidate path. The new member may receive multiple reply messages corresponding to multiple candidate paths. Each reply message collects the QoS properties 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.

**QoSMIC [8]:** In the QoSMIC protocol proposed by Faloutsos et al, the search for candidate paths consists of two parallel 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.

### C. Pros and cons

We compare the pros and cons of SPR protocols and MPR protocols. We also point out their common problem.

The overhead of SPR protocols is low as only one path is searched. However, if the shortest path does not have the resources to meet the QoS requirement, the protocols fail in constructing a feasible tree branch which connects the new member.

In MPR protocols, multiple candidate paths are searched

TABLE I
SPR and MPR

| protocols | overhead | candidate paths | QoS-aware path selection |
|---|---|---|---|
| SPR | low | single | no |
| MPR | high | multiple | no |

to increase the chance of finding a feasible tree branch to connect a new member to the multicast tree. However, the overhead of the current MPR protocols is high in large networks. The spanning-joins protocol makes consecutive broadcasts, which may flood into large areas of the network if the new member is far away from the tree. The tree search in QoSMIC "floods" within the multicast tree which may still reach most of the network for large, dense multicast groups.

All above protocols are not QoS-aware in selecting candidate paths. The selection of on-tree nodes, to which the new member may join, is based on the connectivity in the spanning-joins protocol and QoSMIC.[1] The candidate paths are simply the unicast routing paths from the selected on-tree nodes to the new member. These paths are typically the shortest paths in terms of number of hops, and may not be the best choice for the QoS requirements specified in other terms such as bandwidth. Hence, the information about the specific QoS requirement and the availability of relevant resources should be used to make more effective selection of candidate paths.

## III. QMRP: QoS-Aware Multicast Routing Protocol

In this section, we define the network model, motivate our design objectives, present the routing protocol, and prove several properties.

### *A. Network model*

The network is modeled as a set of nodes that are interconnected by a set of full-duplex, asymmetric communication links. The following assumptions are made.

1. There exists a unicast routing protocol which can deliver a message from one node to any other node in the network.
2. Each node maintains its up-to-date local state, including the resource availability such as the residual (unused) buffer for each network interface and the residual bandwidth on each outgoing link.
3. A node has neither the knowledge of any global network state nor the knowledge of the state of any other node.
4. If a shared multicast tree is used, a new member is able to map a multicast group address to the core node of the tree on demand possibly by a query/response Session Directory [10]. How to select the core of a multicast tree is out of the scope of this paper. Interested readers are referred to [11] for a study.

Given an existing multicast tree and a new member, a *feasible tree branch* is a network path which connects the new member to the tree and has the required resources on every node of the path. The routing process is to search one or more paths in order to find a feasible tree branch.

### *B. Motivation*

First, a cost-effective design is the key for scalability. We want QMRP to have both low average overhead and high overall performance. The following observation provides useful hint on how to achieve these two often-contradictive goals.

For loose QoS requirements which can be easily satisfied[2], searching a single path may be sufficient. The costly process of searching multiple paths can be avoided. For tight QoS requirements, searching multiple paths is necessary in order to increase the chance of success. However, path selection is important and blind flooding is not advisable.

Based on the above observation our protocol starts with a single path but, when necessary, it can expand the search by splitting at one or multiple points in a controlled manner (some related ideas were explored in previous work [12], [13], [14]). The splitting points and contracting points are selected dynamically according to the perceived network conditions in these points.

Second, we want the protocol to be as general as possible. It should be able to operate on top of any unicast routing protocol.[3] With a high-level design the protocol should support different QoS requirements for different users so that it can be used in conjunction with RSVP. With the above objectives in mind, we shall avoid imposing unnecessary restrictions such as to specify the actual mapping between a given QoS requirement and the corresponding required resources. Though some mappings (e.g., throughput and bandwidth) are straightforward, some others may be system-dependent. The protocol does not deal with reserving resources, which is the job of a separate resource reservation protocol such as RSVP.

### *C. Protocol overview*

The Internet is a two-level hierarchy and so is the routing: *intra-domain routing* and *inter-domain routing*. QMRP can work at both levels. In addition, QMRP may be used to construct both sender-based trees or shared trees. In this paper we shall use the shared trees as example.

QMRP is briefly described as follows. When a new member joins the group, it obtains the address of the core of the multicast tree by inquiring the Session Directory. The new

[1] After candidate paths are selected, the protocol becomes QoS-aware because the QoS properties of the candidate paths are collected and checked to see if any of them meet the requirement.

[2] Note that the same QoS requirement may be considered *loose* when the network load is light and *tight* when the network load is heavy.

[3] Just like PIM, QMRP builds the multicast routing table, but during the process it needs the unicast routing table to direct control messages. It does not care what protocol is used to construct the underlying unicast routing table, and therefore it can be deployed in any network that supports unicast routing.

member then initiates the routing process by sending a REQUEST message to the core, following the unicast routing path. Two searching modes are defined: *single path mode* and *multiple path mode*. The routing process starts with the single path mode, attempting to search only the unicast routing path traveled by REQUEST. That is the known path which is able to connect to the tree.

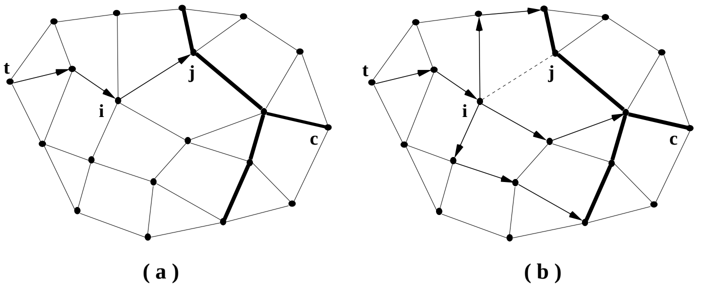

Fig. 2. single path mode and multiple path mode

A REQUEST message has a number of functions. It carries the QoS requirement, e.g., a bandwidth lower bound. 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. Fig. 2 (a) gives an example. Suppose $c$ is the core and the bold lines form the existing multicast tree. Let $t$ be the new member and arrows form the path of the REQUEST message. If every node on the path has the required resources, the path is a feasible tree branch and it is the only path searched by QMRP.

If an intermediate node does not have the required resources, it triggers the *multiple path mode* by sending a NACK message back to the previous node. Upon receipt of NACK, the previous node "detour" the REQUEST message toward directions other than the one defined by the unicast routing path. Namely, REQUEST messages are sent to all neighbor nodes except those from which REQUEST and NACK are previously received. Each REQUEST message independently searchs its own subpath.

Intuitively, a *searching tree* grows as REQUEST messages travel towards the existing multicast tree. NACK messages, as an indication of resource contention, trigger multiple branches in the searching tree to widen the search. Fig. 2 (b) gives an example. Suppose $j$ does not have the required resources, e.g., there is not sufficient bandwidth on link $(j, i)$ to support the quality requirement. We use dotted lines to indicate the lack of bandwidth in this paper. By our assumption, the local state of $i$ does not include the state of incoming links, e.g., $(j, i)$. Hence, the lack of bandwidth on $(j, i)$ will be detected at $j$ when it receives a REQUEST. $j$ replies by sending a NACK to $i$. Upon receipt of NACK, $i$ sends REQUEST messages to search multiple paths. In the figure three paths are searched and all of them are feasible. Once a feasible branch is detected, an ACK message is sent back along the branch towards $t$. In this example three ACK messages will converge at $i$. Node $i$ will select the best branch and reject the other two. In a general routing case, the searching tree may branch at multiple nodes.

### *D. Detailed description*

QMRP implements a five-state state machine at *every node*. The behavior of a node depends on which state it is in. A node may change its state after receiving a control message; the possible state transitions are illustrated in Fig. 3. All QMRP does is to define how a node behaves at each state and when a node changes its state. While every node implements its own state machine, the collective behavior of all nodes realizes the routing process. The five states are: *initial state* (I), *single path state* (SP), *multiple path state* (MP), *failure state* (F), and *success state* (S). Before we present the pseudo code which implements the state machine, we shall first define the control messages and the data structures.

- REQUEST message grows the searching tree.
- NACK message shrinks the searching tree but triggers the receiving node to enter the MP state and thus may subsequently widen the search.
- BREAK message shrinks the searching tree without triggering the MP state.
- ACK message transforms a branch of the searching tree to part of the multicast tree. It also accumulates the property (e.g., delay, bottleneck bandwidth, or number of hops) of the path it traverses for optimization purpose. Such accumulated property carried by ACK is denoted as *ACK.prop*. When multiple feasible tree branches are detected, this value can be used to select the best branch.

Two trees are of concern: the searching tree and the multicast tree. The searching tree is recorded by the temporary *routing entries* at the nodes visited by the REQUEST messages. The data structure of a routing entry is $R\{in, out, id, \beta\}$, which is created upon the receipt of the first REQUEST. *R.in* is the network interface from which the REQUEST is received, and *R.out* is the set of network interfaces to which REQUEST messages are forwarded. It should be noted that data packets will travel in the opposite direction of the REQUEST. *R.id* is a system-wide unique identifier, which may consist of the group address, the new member address, and a sequence number. All control messages must carry this identifier to identify which routing instance they belong to. The state of a node is kept at $\beta$. There are two exceptions: (1) For the nodes that are in the multicast tree, their states are automatically $S$; (2) for the nodes that are neither in the multicast tree nor in the searching tree (i.e., they do not have a routing entry indexed by $id$), their states are $I$ by default.

Since concurrent routing activities initiated by different new members of the same or other groups are seperated by different $id$s, we shall focus on a single instance of routing in most of our discussion.

The multicast tree is recorded by the *multicast entries* at the nodes in the multicast tree. The data structure of a multicast entry is $M\{G, in, out, ...\}$, where $M.G$ is the group

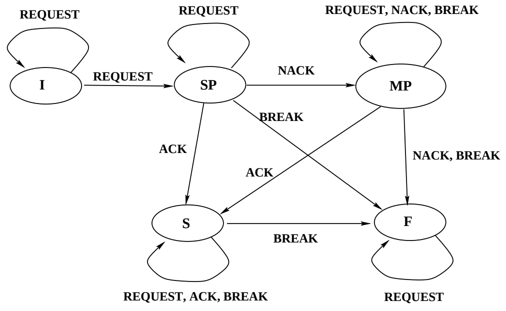


Fig. 3. possible state transitions

address, $M.in$ is the incoming network interface, and $M.out$ is the set of outgoing network interfaces.

Consider a new member $t$ of multicast group $G$. Initially, the searching tree consists of only $t$, which starts with the SP state. All nodes in the multicast tree are in the S state, and all other nodes are in the I state. Suppose a node $i$ receives a message from a network interface $l$ and $j$ is the next hop on the unicast routing path to the core. The following pseudo code defines how $i$ behaves at each state and how $i$ changes its state. Note that every control message carries $id$ of the routing instance it belongs to.

### Initial state (I):

```
switch (the received message)
case REQUEST(id):
    if (i has the required resources)
        create a routing entry R{in, out, id, β}
        R.in := l
        R.out := {j}
        R.β := SP /* change to the SP state */
        send REQUEST(id) to j
    else
        return NACK(id) to l
otherwise:
    discard the received message
```

### Single Path state (SP):

```
switch (the received message)
case REQUEST(id):
    return NACK(id) to l
case NACK(id):
    R.β := MP
    R.out := ∅
    for every network interface l' do
        if (l' ≠ l and l' ≠ R.in)
            R.out := R.out + {l'}
            send REQUEST(id) to l'
case BREAK(id):
    R.β := F
    send BREAK(id) to R.in
case ACK(id):
/* join multicast tree, which changes i to the */
/* S state automatically */
    create multicast entry M{G, in, out, prop}
    M.in := l
    M.out := {R.in}
    M.prop := ACK.prop
    send ACK(id) to R.in
    for every R'{in', out', id', β'} of G, β' = SP or MP, do
        M.out := M.out + {R'.in'}
        send ACK(id') to R'.in'
```

### Multiple Path state (MP):

```
  switch (the received message)
  case REQUEST(id):
      return NACK(id) to l
  case NACK(id):
      R.out := R.out − {l}
      if (R.out = ∅)
          R.β := F
          if (BREAK(id) was received previously)
              send BREAK(id) to R.in
          else
              send NACK(id) to R.in
  case BREAK(id):
      R.out := R.out − {l}
      if (R.out = ∅)
          R.β := F
          send BREAK(id) to R.in
  case ACK(id):
      create a multicast entry M{G, in, out, prop}
      M.in := l
      M.out := {R.in}
*     M.prop := ACK.prop
      send ACK(id) to R.in
      for every R'{in', out', id', β'} of G, β' = SP or MP, do
          M.out := M.out + {R'.in'}
          send ACK(id') to R'.in'
```

### Failure state (F):

```
switch (the received message)
case REQUEST(id):
    return NACK(id) to l
otherwise:
    discard the received message
```

### Success state (S):

```
  switch (the received message)
  case REQUEST(id):
      if (a REQUEST was not received previously)
          return ACK(id) to l
      else
          return BREAK(id) to l
  case BREAK(id):
      M.out := M.out − {l}
      if (M.out = ∅)
          remove the multicast entry M(G, in, out, prop)
          R.β := F
  case ACK(id):
*     if (ACK.prop is better than M.prop)
*         send BREAK(id) to M.in
*         M.in := l
*     else
*         send BREAK(id) to l
  otherwise;
      discard the received message
```

As ACK messages are sent back to $t$, a distributed path selection process is implemented by the above code lines marked by "*", which will keep the best feasible branch and tear down all the others by BREAK messages.

The entire routing process can be illustrated as follows. A searching tree is formed incrementally. Every node in the searching tree must have the required resources. Initially, the searching tree grows along a single path as the intermediate nodes enter the SP state. However, when the lack of resources is detected and an intermediate node enters the

MP state, the searching tree may grow more branches. The tree expands as REQUEST messages are sent forward and shrinks as NACK and BREAK messages are sent backward. If the searching tree shrinks completely and $t$ changes to the F state, the routing process fails in finding a feasible branch. On the other hand, if any tree branch grows to reach the existing multicast tree, the branch is feasible and can be used to connect $t$ to the multicast tree. More than one feasible branch may be detected. ACK messages are used to transform these branches to become part of the multicast tree, while BREAK messages are used to tear down all branches except the best one, so that eventually there will be only one branch which connects $t$ to the multicast tree.

### *E. Protocol properties*

We state and prove some properties of the suggested protocol.

*Definition 1:* The *primary branch* from a new group member to the multicast tree is defined as the shortest prefix of the unicast routing path from the new member to the core that contains a tree node.

*Definition 2:* If a node $i$ in the I state receives a REQUEST from a node $j$ and consequently changes to the SP state, $i$ is called a *child* of $j$.

*Theorem 1:* If the primary branch from a new member to the multicast tree has sufficient resources to accommodate the QoS requirement, QMRP acts as an SPR protocol, i.e., it searches only one path.

*Proof:* Note that a necessary condition for multiple paths to be searched is that at least one node enters the MP state, triggered by a NACK message. However, if sufficient resources are available on every node of the primary branch, no node will ever enter the MP state because no NACK message is produced. Hence, the theorem holds. □

*Lemma 1:* At any time during the routing process, all paths being searched form a tree structure.

*Proof:* The paths being searched are recorded by the routing entries at the nodes that are in SP or MP state. Recall that any routing entry has a single *out* interface and has one or multiple *in* interfaces. Therefore, the nodes form a tree structure based on *in* and *out* variables of the routing entries. This tree is the *searching tree.* □

*Theorem 2:* A feasible branch found by QMRP must be loop-free.

*Proof:* This follows directly from Lemma 1. □

*Lemma 2:* If QMRP terminates without finding a feasible branch, all nodes out of the multicast tree is either in the I state or in the F state.

*Proof:* QMRP terminates without success only when the new member changes to the F state. By the construction of the protocol, a node changes to the F state after all child nodes in the searching tree change to the F state and send back NACK messages.[4] Since the new member is at the root of the searching tree, when it changes to the F state, all nodes in the searching tree must be in the F state. The nodes outside the searching tree remain in the I state. □

[4]Though BREAK messages may also result in the F state, there won't be such messages in this case because BREAK messages are originated from a node having entered the S state, meaning a feasible branch has been detected, which contradicts the assumption.

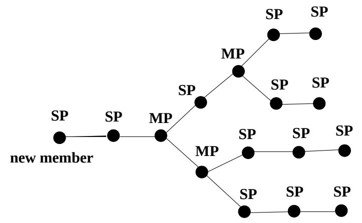


Fig. 4. searching tree of QMRP-2

The following theorem holds only for those QoS requirements specified in non-additive metrics such as bandwidth and buffer space because we assume that the check of resource availability can be performed independently at every node.

*Theorem 3:* QMRP finds a feasible branch if one exists.

*Proof:* We prove the theorem by contradiction. Suppose QMRP fails while a feasible branch does exist. Let $(v, u)$ be the first link in this branch that the protocol did not explore. Namely, there is not a REQUEST message sent from $v$ to $u$. Since $(v, u)$ is the first unexplored link of the branch, $v$ must have received a REQUEST message from the previous link or $v$ is the new group member issuing the routing request. In either case, $v$ is not in the I state. Hence, $v$ is in the F state by Lemma 2. However, $v$ must arrive at this state through the MP state[5], which requires $v$ to explore all outgoing links including $(v, u)$. It contradicts the assumption that $(v, u)$ is not explored. □

### *F. Restricted QMRP*

Any number of nodes may enter the MP state, which results in high worst-case overhead. An easy way to control the overhead is to maintain an assertion: between the new member and any node in the searching tree, there are at most $m$ nodes entering the MP state. In other words, the maximum number of nodes allowed to enter the MP state is $\Sigma_{i=0}^{m-1}(d-2)^i = \frac{(d-2)^m-1}{d-3}$, where $d$ is the maximum degree of a node. Such a restricted version of QMRP is denoted as QMRP-$m$. An illustration of QMRP-2 is given in Fig. 4. Our simulations show that (1) QMRP-2 scales well because of its limited overhead and (2) its routing performance does not degrade significantly from the unrestricted QMRP. QMRP-$m$ can be easily implemented by augmenting the routing messages with a branching counter.

## IV. Analysis

In this section we compute the improvement in the reservation success probability when QMRP is used in two networks with regular structures. Throughout the analysis we assume that the probability of finding the required resources

[5]Though a node may change from the SP state directly to the F state by receiving a BREAK message, there won't be such a message in this case by the same reason given in the previous footnote.

on any link is $p$, and this probability is independent for every link. A *feasible link* (*subroute*) is a link (subroute) which has the required resources. To simplify the analysis, we assume that the underlying (Internet) unicast routing algorithm uses the shortest path routing. We assume an empty tree, i.e., only the core node is a valid point for connecting to the tree. We analyze the most conservative implementation of the algorithm, QMRP-1. In this implementation the algorithm is allowed to branch, at most, once.

We selected to analyze triangulated networks and grids since they are of practical interest. The structure of many WANs, and to a greater extend MANs, is very close to a combination of triangles and square (e.g., see the networks in [15]). In the analysis we concentrate on the case when the shortest path between two nodes is a straight line, since this case is the least advantageous for our algorithm.

### A. lattice of triangles

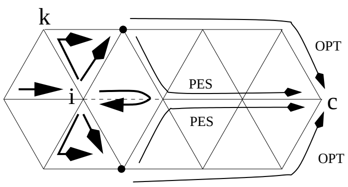


Fig. 5. A depiction of the algorithm work on a lattice of triangles

As mentioned above, we consider an $h$-hop shortest route that has no turns. It is feasible with probability $p^h$. If one of the links is not feasible, the search branches to four directions (see Fig. 5). To be successful, the search has to pass through the two points marked by black circles in Fig. 5. From node $i$, the conditional probability to arrive at any of these two points is $p_b = 1-(1-p)(1-p^2/2) = \frac{p}{2}(2+p-p^2)$, since $p$ is the probability that the direct link to the black point is feasible, and $p^2/2$ is the probability that the two hop route to the black point is feasible.[6] After passing the black points, the two search processes (assuming both bypasses succeed) may continue along one of many possible shortest pathes. In particular, the search may continue along the original path immediately (denoted in Fig. 5 by PES), take two disjoint paths until reaching the destination (denoted in Fig. 5 by OPT), or share part of the path. We make no assumption obout the underlying unicast routing protocol and thus analyze the two extreme cases: a pessimistic scenario (PES) where the underlying unicast routing protocol selects the path offering the minimal expected success improvement, and an optimistic scenario (OPT) where the underlying unicast routing protocol selects the path offering the maximal expected success improvement. The success probability of any other path which may be chosen by the underlying unicast routing protocol lays between these calculated probabilities.

In the pessimistic scenario, the success probability of each bypass is given by $p_b$ as explained above multiplied by $p$ which is the probability that the link leading to the original route is feasible. The original route should have, at least, $h-1$ feasible links. Since for success we need that only one of the two bypasses will succeed to find a feasible route, the success probability is given by (using the fact that the success along two parallel routes with success probability $p_x$ is $p_x(2-p_x)$):

$$\begin{aligned} & P_{suc}\ \&\ \text{failure at link}\ _i \qquad (1)\\ &= p^{h-1}(1-p)\cdot p\cdot \frac{p}{2}(2+p-p^2)\left(2-p\frac{p}{2}(2+p-p^2)\right)\\ &= \frac{1}{2}p^{h+1}(1-p)(2+p-p^2)\left(2-\frac{1}{2}p^2(2+p-p^2)\right)\end{aligned}$$

and the total success probability is

$$\begin{aligned} P_{suc} = p^h &+ \sum_{i=1}^{h} P_{suc}\ \&\ \text{failure at link}\ _i \qquad (2)\\ = p^h &+ h\cdot\frac{1}{2}p^{h+1}(1-p)(2+p-p^2)\cdot\\ &\left(2-\frac{1}{2}p^2(2+p-p^2)\right)\end{aligned}$$

The optimistic scenario analysis depends on $l$, the location where the non-feasible link is encountered. The probability that one of the disjoint routes will be feasible is $p^{h+1-l}p_b$. This has to be multiplied by the success probability along the path before the non-feasible link was encountered, which is given by $p^{l-1}$. Thus the success probability for $0 < l \leq h$ is

$$\begin{aligned} & P_{suc}\ \&\ \text{failure at link}\ _l \qquad (3)\\ &= p^{l-1}(1-p)p^{h+1-l}p_b(2-p^{h+1-l}p_b)\\ &= p^{h+1}(1-p)(2+p-p^2)\left(1-\frac{p^{h+2-l}(2+p-p^2)}{4}\right)\end{aligned}$$

and the total success probability is

$$\begin{aligned} P_{suc} = p^h &+ \sum_{i=1}^{h} P_{suc}\ \&\ \text{failure at link}\ _i \qquad (4)\\ = p^h &+ \frac{1}{2}p^{h+1}(1-p)(2+p-p^2)\cdot\\ &\sum_{l=1}^{h}(2-\frac{1}{2}p^{h+2-l}(2+p-p^2))\\ = p^h &+ hp^{h+1}(1-p)(2+p-p^2)\\ &- \frac{1}{4}p^{h+3}(2+p-p^2)^2(1-p^h)\end{aligned}$$

[6]The division by 2 in $p^2/2$ is explained as follows: Let the middle node of the two hop route be $k$ and the node in the MP state be $i$. Both $i$ and the black point are on shortest paths from $k$ to the core. After $k$ receives a REQUEST from $i$, it forwards the message to the next hop provided by the underlying unicast routing protocol. The unicast routing protocol operates *independently* from QMRP-1 and thus may select either $i$ or the black point as the next hop. Let us assume that $i$ and the black point have equal chance to be the next hop. Then, the chance for $k$ to forward the REQUEST to the black point, which completes the two hop route shown in the figure, is one half. Note that, if $i$ is selected as the next hop, because a REQUEST should not be sent back to the originator of the message, a NACK will be sent back to $i$.

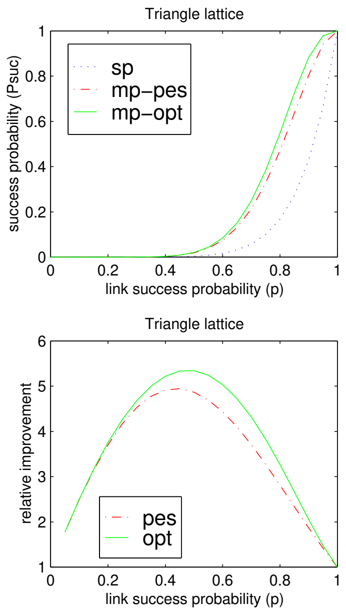


Fig. 6. The success probability of QMRP-1 on an eight link path in a lattice of triangles (mp-pes and mp-opt are the pessimistic and optimistic analysis results, respectively. sp is the single path success probability).

Fig. 6 shows the difference in success probability between QMRP-1 and a single path protocol such as [9]. For the entire range of $p$, the difference between the optimistic and pessimistic performance of QMRP-1 is small relative to the improvement achieved over the single path protocol. The difference between QMRP-1 and the single path protocol grows linearly with the path length for all values of $p$ (graphs omitted due to space limitation). However, the cost in extra control messages is at most a factor of two.

A performance comparison with several other protocols will be done in the next section by simulation.

### B. Grids

The grid analysis is similar to the above, only simpler. Here, too, when a non-feasible link is encountered the search for a route may continue in two paths that may be totally disjoint (the optimistic scenario), share the same links besides the ones used to bypass the first non-feasible link (the pessimistic scenario), or share some sub-path.

The pessimistic scenario requires, at least $h-1$ of the links along the path to be feasible ($p^{h-1}$), and that, at least, one three-hop bypass route be feasible ($p^3(2-p^3)$), which translates to

$$\begin{aligned} P_{suc} &= p^h + hp^{h-1}(1-p)p^3(2-p^3) \\ &= p^h + hp^{h+2}(1-p)(2-p^3) \end{aligned} \tag{5}$$

In the optimistic case, the probability for success given that the first non-feasible link is at distance $l$ is given by

$$\begin{aligned} & P_{suc}\ \&\ \text{failure at link}\ l \\ &= p^{l-1}(1-p)p^{h+3-l}(2-p^{h+3-l}) \\ &= p^{h+2}(1-p)(2-p^{h+3-l}) \end{aligned} \tag{6}$$

and thus

$$\begin{aligned} P_{suc} &= p^h + \sum_{i=1}^{h} P_{suc}\ \&\ \text{failure at link}\ i \\ &= p^h + 2hp^{h+2}(1-p) - p^{h+5}(1-p^h) \end{aligned} \tag{7}$$

Here the results are not as impressive as with the triangle lattice since a detour around a non-feasible link is longer. The pessimistic and optimistic scenarios bring gains of 1.75 and 3.25, respectively, for an eight hop path (graphs omitted). The gain here also grows linearly with the path length.

## V. Simulation Results

We study the effectiveness and the scalability of QMRP by simulation. 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 link is counted as *one message*. Hence, for a message having traversing a path of $l$ hops, $l$ messages are counted.

The network used in our simulation is randomly generated by the following procedure: N nodes are randomly placed in an one-by-one square. Each node initiates two links to other nodes. Therefore, the minimum degree of a node is two and the average degree is four. For nine out of ten links, a node tries to connect to its nearest unconnected neighbor; for one out of ten links, a node connects to a randomly selected unconnected node. (The simulation resutls for Waxman network topologies [16], omitted due to space limitation, are similar.)

Five algorithms are simulated: *SPR*, *optimal protocol*, *QMRP-2*, *QoSMIC* [8], and *spanning-join protocol* [7]. We assume a unicast routing protocol providing the shortest path in term of hops between each pair of nodes. Given a new member, the SPR protocol searches only the shortest path, and hence it has the smallest success ratio and message overhead. The optimal protocol does exhaustive search and has the optimal success ratio. The other three protocols are between the two extremes and search multiple paths.

Fig. 7 and Fig. 8 compare the success ratios of the above protocols. Random networks with 300 nodes were used in the simulation. Each point in the figure is the result of 40000 simulation runs. In each simulation run, a random multicast tree with 25 (100) nodes was generated and a new member out of the tree was randomly selected. The state of each *directed* link [7] is randomly generated, either having the required resources or not having the required resources, based on certain probability (link success probability along the horizontal axis). The five routing protocols were then used to find a feasible branch for the new member. To achieve

[7] This is to allow asymmetric state along two directions of a link.

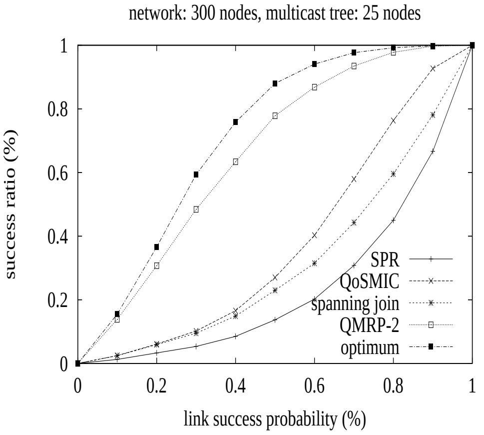


Fig. 7. success ratio

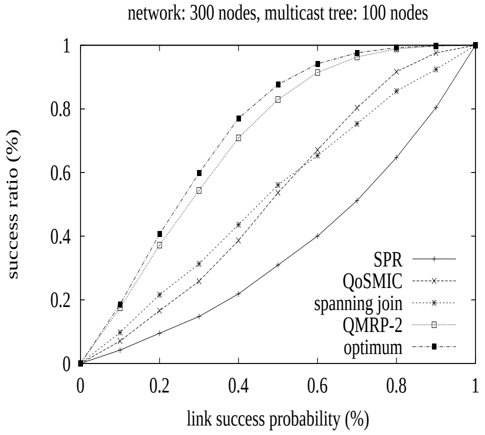


Fig. 8. success ratio

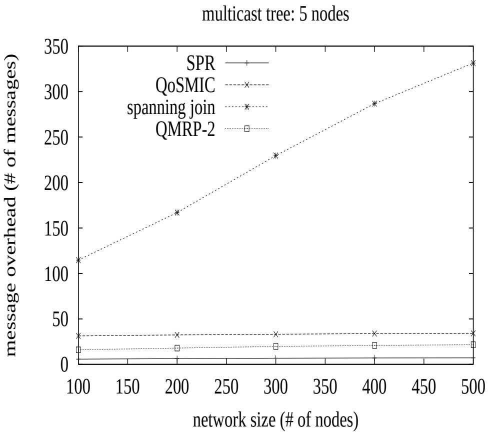


Fig. 9. message overhead

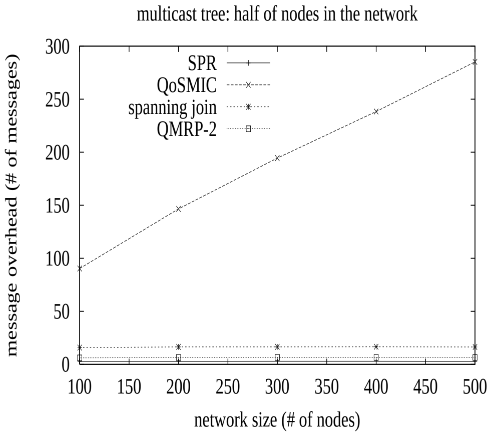


Fig. 10. message overhead

a data point in the figures, the above simulation run was executed for 4000 times for each of 10 randomly created networks. For any data point plotted, the standard deviation is less than 4% of the value of the data point.[8]

The simulation results show that the success ratios of QMRP-2 are close to those of the optimal protocol and significantly better than those of SPR, QoSMIC, and the spanning join protocol. The success-ratio curves of QMRP-2 are different from those of QMRP-1 in Fig. 6 (upper curve). That is because QMRP-2 allows more nodes to enter the MP state whereas QMRP-1 allows only one. Hence, QMRP-2 has much better success ratio than QMRP-1.

The ability of QMRP-2 to achieve better success ratio than a flooding scheme such as spanning join is surprising and requires elaboration. Consider the example in Fig. 11. Suppose a new member $t$ wants to join an existing multicast tree shown by bold lines. In order to find an on-tree node,

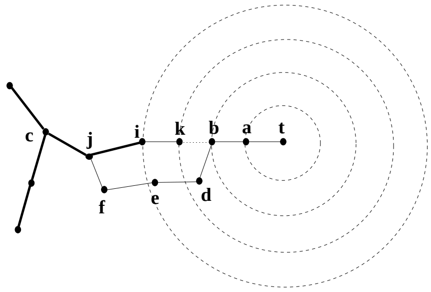


Fig. 11. A flooding scheme does not necessarily achieve better success ratio.

the spanning join protocol searches increasingly larger areas indicated by the spanning rings in the figure. Once a spanning ring hits an on-tree node $i$, the searching process terminates. Then a single path (the unicast routing path) from $i$ to $t$ is checked to see if the QoS requirement can be satisfied. If one link $(k, b)$ does not have the required resources, the routing fails.[9] In our simulation, we found that the flooding

[8]For every 50 simulation runs, a *sample* success ratio is calculated. Hence, there are 800 samples. A plotted data point is the estimated *mean* success ratio, which is the average of the samples. With a confidence level of 95%, the confidence intervals are within ± 4% of the mean for most data points. A few data points have confidence intervals within ± 8% of the mean.

[9]If we apply QoSMIC to this example, the same thing happens. Only $i$ is selected and a single candidate path is searched.

scheme of spanning join typically results in few candidate paths. It selects those on-tree nodes which are closest to the new member and use the unicast routing paths as candidate paths. QMRP takes a very different approach. It starts with a single path towards the core. Once it encounters an infeasible link $(k, b)$, it branches and make detour to follow other downstream paths such as $b \to d \to e \to f \to j$ to join the tree. It avoids flooding but can potentially search a lot of paths, and it always branches for additional paths at the *right places*.

In Theorem 3, we proved that the success ratio of the unrestricted QMRP is equal to that of the optimal protocol which finds a feasible branch when such a branch exists. However, the unrestricted QMRP allows unlimited branches in the searching tree, which means flooding in the worst case. By limiting the number of searching branches, QMRP-2 can achieve sub-optimal success ratio while keeping the overhead low.

The above simulation was repeated with random networks of different sizes to study scalability. Fig. 9 and Fig. 10 present the average message overhead when the link success ratio is 80%. Similar results were observed for other link success ratios. The result for small multicast groups is presented in Fig. 9, where multicast trees with 5 nodes [10] were used. The result for large multicast groups is presented in Fig. 10, where multicast trees with half of the nodes in the network were used. The spanning join protocol does not scale well for small multicast groups because it floods until reaching the tree. QoSMIC does not scale well for large multicast groups because it "floods" in the multicast tree which covers a major portion of the network. QMRP-2 always scales well because it becomes SPR if the shortest path succeeds, it enters the MP state only when necessary, and it allows limited nodes to enter the MP state.

The simulations show that QMRP-2 achieves better success ratio and lower overhead than QoSMIC or the spanning join protocol. We repeated the simulations on Waxman network topologies [16] and similar results were observed.

## VI. Conclusion

We proposed QMRP, a new QoS-aware multicast routing protocol. In order to achieve scalability, QMRP adopts a cost-effective design which achieves both low communication overhead and high routing performance. The protocol is self-adaptable according to the network conditions without the administrative help. It searches only a single path if that is sufficient, whereas the spanning-join protocol or QoSMIC always has flooding behavior even for a trivial join. QMRP switches to multiple path search when necessary. The path selection and optimization are realized by a distributed implementation where nodes switch among five states, providing a fine mechanism to tune the amount of overhead needed to find a feasible tree branch. Such a mechanism is simple to understand and easy to implement, while leaving enough flexibility to achieve efficiency, robustness, responsiveness, and loop-free routing. The efficiency is achieved by growing the searching tree along the positive directions and shrinking the tree from the directions which do not have the required resources. The robustness comes from a fully distributed implementation which does not rely on any extra control element. The responsiveness is achieved by a termination detection mechanism which avoids the use of timeout. The loop-free routing is achieved by maintaining a searching tree at any time.

[10] The 5 nodes includes the members and the non-leaf nodes which are not members.

## References


[1] 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.

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

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

[4] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "An Architecture for Wide-Area Multicast Routing," *ACM SIGCOMM*, pp. 126–135, August 1994.

[5] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast Routing for Multimedia Communication," *IEEE/ACM Transactions on Networking*, June 1993.

[6] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting," *Proc. IEEE INFOCOM 95, Boston, MA*, April 1995.

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

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

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

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

[11] D. G. Thaler and C. V. Ravishankar, "Distributed Center-Location Algorithms," *IEEE JSAC*, vol. 15, pp. 291–303, April 1997.

[12] Y. 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.

[13] I. Cidon, R. Rom, and Y. Shavitt, "Multi-Path Routing Combined with Resource Reservation," *IEEE INFOCOM'97*, pp. 92 – 100, April 1997.

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

[15] Y. Xiong and L. G. Mason, "Restoration strategies and spare capacity requirements in self-healing ATM networks," *IEEE Tran. on Networks*, vol. 7, no. 1, pp. 98–110, Feb. 1999.

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

[17] R. G. Busacker and T. L. Saaty, *Finite Graphs and Networks: an introduction with applications*, McGraw-Hill, 1965.

[18] Y. K. Dalal and R. M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," *Communications of the ACM*, vol. 21, no. 12, pp. 1040–1048, 1978.

[19] S. Verma, R. K. Pankaj, and A. L. Garcia, "Performance of QoS Based Multicast Routing Algorithms for Real-Time Communication," *Performance Evaluation Journal*, vol. 21, 1998.