A QoS-Aware Multicast Routing Protocol

Shigang Chen, *Member, IEEE*, Klara Nahrstedt, *Member, IEEE*, and Yuval Shavitt, *Member, IEEE*

***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 acalability 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 intradomain and interdomain. 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.

***Index Terms*—Multicast routing, multiple-path routing, quality of service.**

I. Introduction

MULTICAST employs a tree structure in the network to efficiently deliver the same data stream to a group of receivers. Traditionally, research on Internet multicast 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 is needed 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 for QoS 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 for nonadditive metrics such as bandwidth and buffer space. It 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 search process only when that is necessary. In many cases it behaves like PIM and searches only a single 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 that 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 onto 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 in the network.
- *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 order to accommodate the worst-case situation, the

Manuscript received October 15, 1999; revised April 15, 2000. This work was supported in part by the Airforce under Contract F30602-97-2-0121 and the National Science Foundation Career under Contract NSF CCR 96-23867.

S. Chen is with Cisco Systems, San Jose, CA 95134 USA (e-mail: sgchen@cisco.com).

K. Nahrstedt is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: klara@cs.uiuc.edu).

Y. Shavitt is with Bell Laboratories, Lucent Technologies, Holmdel, NJ 07733 USA. He is also with the Department of Electrical Engineering - Systems, Tel Aviv University, Tel Aviv 69978, Israel (e-mail: shavitt@ieee.org).

Publisher Item Identifier S 0733-8716(00)09227-1.

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.[1]

- *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.

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. 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 to 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 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.

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 neighborhoods until on-tree nodes are found. This process can increase the overhead significantly.

[1]Note that timeout is often needed in a network environment to detect message loss. However, having timeouts to detect message loss is different from having them as part of the protocol that slows the protocol down even in a lossless environment.

TABLE I
SPR and MPR

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

QoSMIC [8]: In the QoSMIC protocol proposed by Faloutsos *et al.*, the search for candidate paths consists of two procedures, *local search* and *tree search*, which can be executed in parallel or sequentially. 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 (Table I). We also point out their common problem.

The overhead of SPR protocols is low because 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 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 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 a major portion of the network for large, dense multicast groups.

All above-mentioned 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 or QoSMIC.[2] 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 by other metric 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.

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

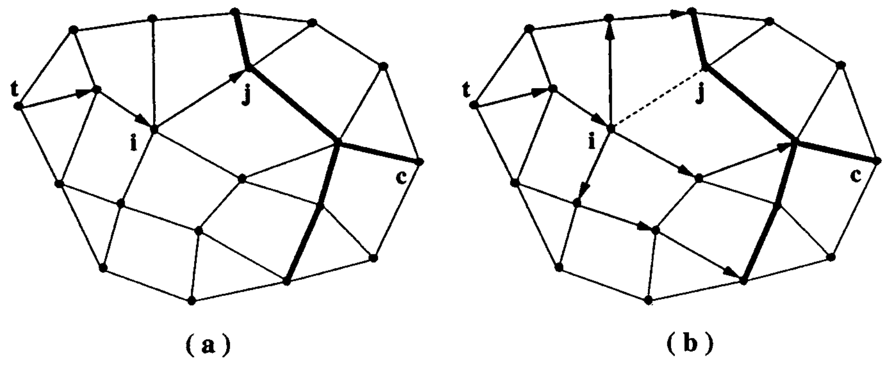


Fig. 1. (a) Single path mode and (b) multiple path mode.

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 up-to-date local state, including resource availability such as the residual (unused) buffer space 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 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 searches 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 hints on how to achieve these two often contradictive goals.

For loose QoS requirements which can be easily satisfied,[3] 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 [12]–[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.[4] 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 specifying the actual mapping between a given QoS requirement and the corresponding required resources. Though some mappings (e.g., throughput and bandwidth) are straightforward, 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: *intradomain routing* and *interdomain 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 an 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 member then initiates the routing process by sending a REQUEST message to the core, following the unicast routing path. Two search 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.

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. 1(a) gives an example. Suppose c is the

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

[4]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.

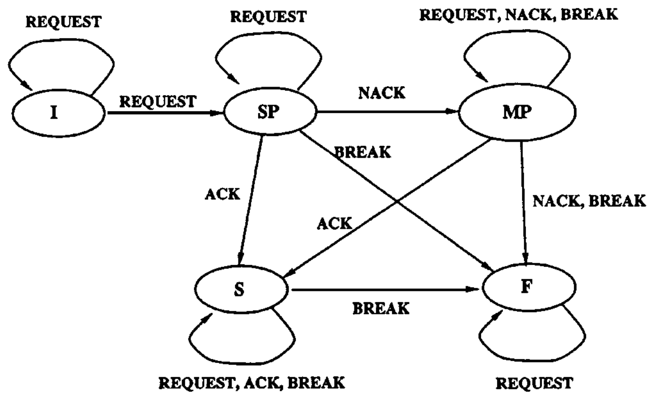


Fig. 2. Possible state transitions.

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 "detours" 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 searches its own subpath.

Intuitively, a *search tree* grows as REQUEST messages travel toward the existing multicast tree. NACK messages, as an indication of resource contention, trigger multiple branches in the search tree to widen the search. Fig. 1(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 toward 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 search 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. 2. QMRP defines 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 pseudocode which implements the state machine, we shall first define the control messages and the data structures.

- REQUEST message grows the search tree.
- NACK message shrinks the search tree but triggers the receiving node to enter the MP state and thus may subsequently widen the search.
- BREAK message shrinks the search tree without triggering the MP state.
- ACK message transforms a branch of the search tree to part of the multicast tree. It also accumulates some 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 search tree and the multicast tree. The search tree is recorded by the temporary *routing entries* at the nodes visited by the REQUEST messages. The data structure of a routing entry is $R\{\mathrm{in},\, \mathrm{out},\, id,\, \beta\}$. "$R$.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. Since concurrent routing activities initiated by different new members of the same or other groups are separated by different ids, we shall focus on a single instance of routing in most of our discussion. The state of the node is kept at β. 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 not in the multicast tree and do not have a routing entry indexed by id, their states for this instance of routing are I by default.

The multicast tree is recorded by the *multicast entries* at the nodes in the multicast tree. The data structure of a multicast entry is $M\{G,\, \mathrm{in},\, \mathrm{out},\, \cdots\}$, where "$M.G$" is the group address, "M.in" is the incoming network interface, and "M.out" is the set of outgoing network interfaces.

Initially, a new member t of multicast group G is in the SP state, all nodes in the multicast tree are in the S state, and all other nodes are in the I state. Suppose node i receives a message from network interface l and j is the next hop on the unicast routing path to the core. The following pseudocode 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 $R\{\mathrm{in}, \mathrm{out}, id, \beta\}$
    $R.\mathrm{in} := l$
    $R.\mathrm{out} := \{j\}$
    $R.\beta := \mathrm{SP}$/* 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($id$):
  $R.\beta := \mathrm{MP}$
  $R.\mathrm{out} := \emptyset$
  for every network interface $l'$ do
    if $(l' \neq l \,\mathbf{and}\, l' \neq R.\mathrm{in})$
      $R.\mathrm{out} := R.\mathrm{out} + \{l'\}$
      send REQUEST($id$) to $l'$
case BREAK($id$):
  $R.\beta := \mathrm{F}$
  send BREAK($id$) to $R.\mathrm{in}$
case ACK($id$):
/* join multicast tree, which changes $i$ to
  the S state automatically */
  create multicast entry $M\{G, \mathrm{in}, \mathrm{out}, prop\}$
  $M.\mathrm{in} := l$
  $M.\mathrm{out} := \{R.\mathrm{in}\}$
  $M.prop := ACK.prop$
  send ACK($id$) to $R.\mathrm{in}$
  for every $R'\{\mathrm{in}', \mathrm{out}', id', \beta'\}$ of $G$ do
      $M.\mathrm{out} := M.\mathrm{out} + \{R'.\mathrm{in}'\}$
      send ACK($id'$) to $R'.\mathrm{in}'$
```

```
Multiple Path state (MP):
  switch (the received message)
  case REQUEST($id$):
    return NACK($id$) to $l$
  case NACK($id$):
    $R.\mathrm{out} := R.\mathrm{out} - \{l\}$
    if $(R.\mathrm{out} = \emptyset)$
      $R.\beta := F$
      if (BREAK($id$) was received previously)
        send BREAK($id$) to $R.\mathrm{in}$
      else
        send NACK($id$) to $R.\mathrm{in}$
  case BREAK($id$):
    $R.\mathrm{out} := R.\mathrm{out} - \{l\}$
    if $(R.\mathrm{out} = \emptyset)$
      $R.\beta := F$
      send BREAK($id$) to $R.\mathrm{in}$
  case ACK($id$):
    create a multicast entry $M\{G,\mathrm{in},\mathrm{out},prop\}$
    $M.\mathrm{in} := l$
    $M.\mathrm{out} := \{R.\mathrm{in}\}$
*   $M.prop := ACK.prop$
    send ACK($id$) to $R.\mathrm{in}$
    for every $R'\{\mathrm{in}', \mathrm{out}', id', \beta'\}$ of $G$ do
      $M.\mathrm{out} := M.\mathrm{out} + \{R'.\mathrm{in}'\}$
      send ACK($id'$) to $R'.\mathrm{in}'$
```

```
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($id$):
    return ACK($id$) to $l$
  case BREAK($id$):
    $M.\mathrm{out} := M.\mathrm{out} - \{l\}$
    if $(M.\mathrm{out} = \emptyset)$
      remove the multicast entry
  $M(G, \mathrm{in}, \mathrm{out}, prop)$
      $R.\beta := F$
  case ACK($id$):
*   if ($ACK.prop$ is better than $M.prop$)
*     send BREAK($id$) to $M.\mathrm{in}$
*     $M.\mathrm{in} := l$
*   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 keeps the best feasible branch and tears down all the others using BREAK messages. Fig. 3 gives a simple example. In (a), the existing multicast tree is presented in bold lines and the search tree is presented in thin lines with a branching point at i. Both branches of the search tree are feasible and they reach the multicast tree at a and d, respectively. We assume, for this example, that the information carried in ACK.$prop$ is the number of hops. Suppose the ACK from d arrives at i first, and it will set $M.prop$ to be 3 hops (see the ACK case in the code segment for MP). This ACK transforms the branch, $d \rightarrow e \rightarrow f \rightarrow i \rightarrow t$, to become part of the multicast tree as shown in (b). In (c), the ACK from a arrives at i at a later time. It transforms $a \rightarrow b \rightarrow i$ to part of the multicast tree, and it carries a better ACK.$prop$, which is 2 hops. In (d), i sends a BREAK message to trim the inferior branch, $d \rightarrow e \rightarrow f \rightarrow i$. At the end, a single path, $a \rightarrow b \rightarrow i \rightarrow t$, is left to connect t to the multicast tree. For

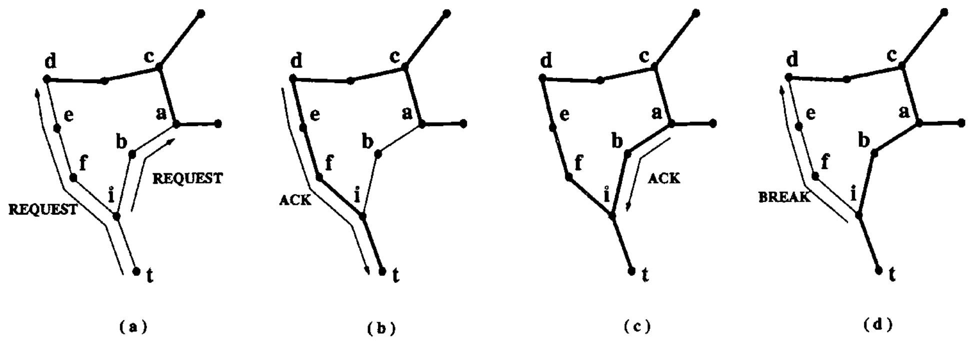


Fig. 3. Path selection.

more complex search trees with multiple branching points, the above process will be repeated at every branching point.

The entire routing process can be illustrated as follows. A search tree is formed incrementally. Every node in the search tree must have the required resources. Initially, the search 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 search tree may grow more branches. The tree expands as REQUEST messages are sent forward and shrinks as NACK messages are sent backward. If the search tree shrinks completely and t changes to the F state, the routing process fails to find 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 that connects t to the multicast tree.

When a group member in the multicast tree leaves the multicast group, if it is a leaf node, it sends a control message up the tree to prune the tree branch that no long connects any group member. When an on-tree node receives this pruning message, if the node is not a group member and does not have any other child node, it propagates the pruning message to its parent and prunes itself from the tree. If the node is a group member, or it has a child node other than the one from which the pruning message is received, it discards the message.

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 state I receives a REQUEST from node j and consequently changes to state SP, 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 state MP, triggered by a NACK message. However, if sufficient resources are available on every node of the primary branch, no node will ever enter state MP 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 states SP or MP. Recall that any routing entry has a single *in* interface and has one or multiple *out* interfaces. Therefore, the nodes form a tree structure based on *in* and *out* variables of the routing entries. This tree is the *search 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 state I or in state F.

Proof: QMRP terminates without success only when the new member enters state F. By the construction of the protocol, a node enters state F after all child nodes in the search tree enter state F and send back NACK messages.[5] Since the new member is at the root of the search tree, when it enters state F, all nodes in the search tree must be in state F. The nodes outside the search tree remain in state I. □

The following theorem holds only for the unrestricted QMRP, and does not hold for QMRP-m, for any finite m, which is defined in the next section.

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 state I. Hence, v is in state F by Lemma 2. However, v must arrive at this state through state MP,[6] which requires v to

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

[6]Although a node may change from state SP directly to state F by receiving a BREAK message, there will not be such a message in this case by the same reason given in the previous footnote.

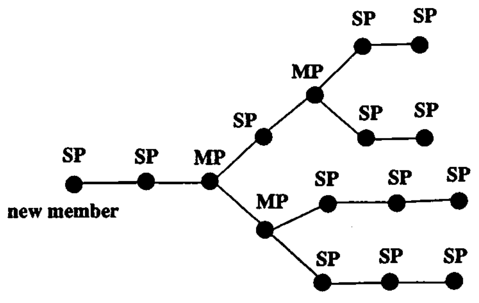


Fig. 4. A search tree of QMRP-2.

explore all outgoing links including (v, u). It contradicts the assumption that (v, u) is not explored. □

F. Restricted QMRP

QMRP, if not restricted, can potentially grow big search trees and lead to large overhead. We define two protocol parameters that are used to restrict QMRP.

Maximum Branching Level (MBL): If there are too many nodes entering the MP state, the overhead will be large. An easy way to control the overhead is to maintain an assertion: between the new member and any node in the search 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 $\sum_{i=0}^{m-1}(d-2)^i = ((d-2)^m - 1)/(d-3)$, where d is the maximum degree of a node. Such a restricted version of QMRP is denoted as QMRP-m. This number m is called the *maximum branching level*. 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) it achieves better routing performance than previous protocols such as QoSMIC and spanning join. QMRP-m can be easily implemented by augmenting the routing messages with a counter.

Maximum Branching Degree (MBD): A node entering the MP state may have a large number of adjacent links, which can also cause excessive overhead. Any QMRP-m can be further argumented with an additional parameter, *maximum branching degree*, which specifies the maximum number of REQUEST messages that are allowed to be sent by an MP-state node. If the maximum branching degree x is greater than the node degree minus two,[7] the node selects x outgoing links (randomly or based on distance to the tree core) from which REQUEST or NACK has not been received, and sends REQUEST 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 performance but may add much overhead. An MBD of 10 worked well in our simulation for the power-law network topologies [15].

We suggest both MDL and MBD to be implemented. With MDL $= m$ and MBD $= x$, the maximum number of nodes allowed to enter the MP state is $\sum_{i=0}^{m-1} x^i = (x^m - 1)/(x-1)$. Therefore, the overhead can be controlled by these two parameters.

[7]Before a node enters the MP state, it should have received a REQUEST and a NACK. The node should not send REQUEST to a link from which a REQUEST (NACK) has been previously received.

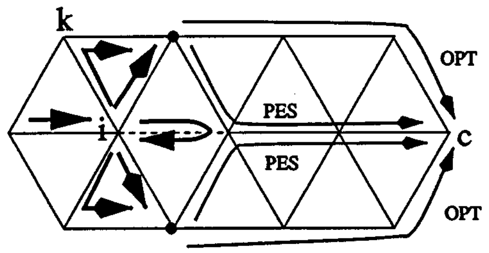


Fig. 5. A depiction of the algorithm work on a lattice of triangles.

IV. Analysis

In this section we compute the improvement in the success probability when QMRP is used in two networks with regular structures. Throughout the analysis we assume that the probability to succeed in finding sufficient resources on a link is p for all the links, and this probability is independent for every link. We call a link (subroute) on which successful routing can continue a *feasible link (subroute)*. 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 triangulated networks and grids in our analysis since they are of practical interest. The structure of many WAN's, and to a greater extend MANS, is very close to a combination of triangles and square (e.g., see the networks in [16]). In this 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

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 $p_b = 1-(1-p)(1-p^2/2) = (p/2)(2+p-p^2)$, 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.[8]

After passing the black points, the two search processes (assuming both bypasses succeed) may continue along one of many possible shortest paths. In particular, the search may continue along the original path immediately (denoted in Fig. 5 by PES),

[8]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 an 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.

taking two disjoint paths until reaching the destination (denoted in Fig. 5 by OPT), or share part of the path. We make no assumption about 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. For success, we require that only one of the two bypasses will succeed to find a feasible route; thus, the conditional success probability is given by [using the fact that the success along two parallel routes with success probability p_x is $p_x(2-p_x)$]:

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

and the total success probability is

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

The best-case analysis depends on l, the location where the nonfeasible 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 nonfeasible link was encountered, which is given by p^{l-1}. Thus, the overall success probability for $0<l\le h$ is

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

and the total success probability is

$$\begin{aligned}P_{suc}&=p^h+\sum_{i=1}^{h}P_{suc}\&\text{ failure at link } i\\&=p^h+\tfrac{1}{2}p^{h+1}(1-p)(2+p-p^2)\\&\quad\cdot\sum_{l=1}^{h}\left(2-\tfrac{1}{2}p^{h+2-l}(2+p-p^2)\right)\\&=p^h+hp^{h+1}(1-p)(2+p-p^2)\\&\quad-\tfrac{1}{4}p^{h+3}(2+p-p^2)^2(1-p^h).\end{aligned}\quad(4)$$

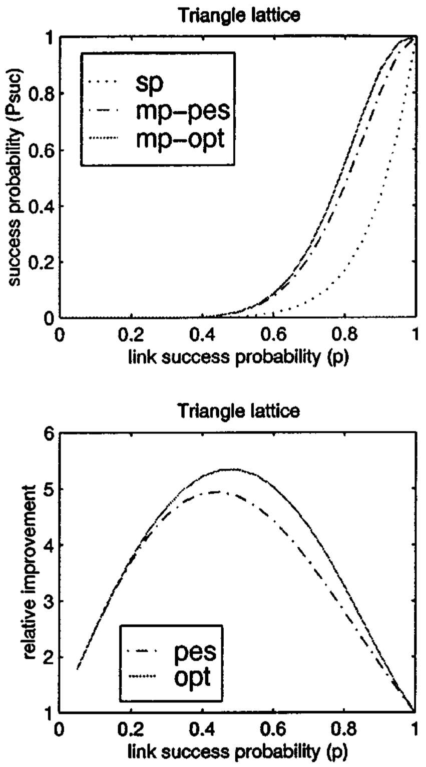


Fig. 6. The success probability of QMRP-1 on an eight link path in a lattice of triangles (mp-pes and mp-opt are the pessimistic and optimistic analysis results, respectively. sp is the single path success probability).

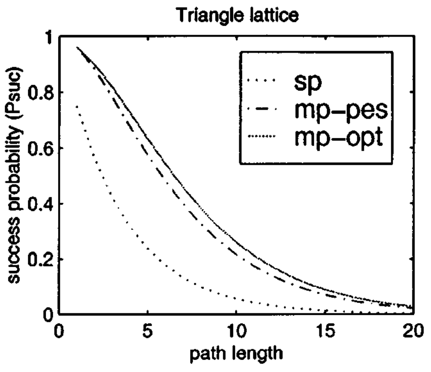
Fig. 6 shows the difference in success probability between QMRP-1 and a single path protocol such as [9]. For the entire range of p, the difference between the optimistic and pessimistic performance of QMRP-1 is small relative to the improvement achieved over the single path protocol. The difference between QMRP-1 and the single path protocol grows linearly with the path length for all values of p (see Fig. 7 for $p=0.75$). However, the cost of extra control messages is at most a factor of two.

B. Grids

The grid analysis is similar to the above analysis, only simpler. Here, too, when a nonfeasible 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 one used to bypass the first nonfeasible link (the pessimistic scenario), or share some subpath.

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

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



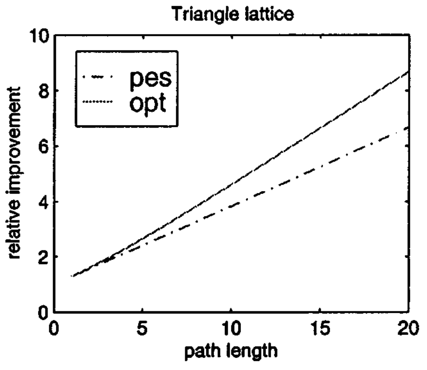


Fig. 7. The success probability of QMRP-1 for $p = 0.75$ as a function of the path length in a lattice of triangles (mp-pes and mp-opt are the pessimistic and optimistic analysis results, respectively. sp is the single path success probability).

In the optimistic case, the probability for success given that the first nonfeasible 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 nonfeasible link is longer. The pessimistic and optimistic scenarios gain a factor of 1.75 and 3.25, respectively, for an eight-hop path. Also, the difference between the two is larger (graphs omitted). But the gain here also grows linearly with the path length.

A simulation performance comparison of QMRP to several other protocols is presented in the next section.

V. Simulation Results

In this section, we study the performance 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 path of l hops is counted as l messages.

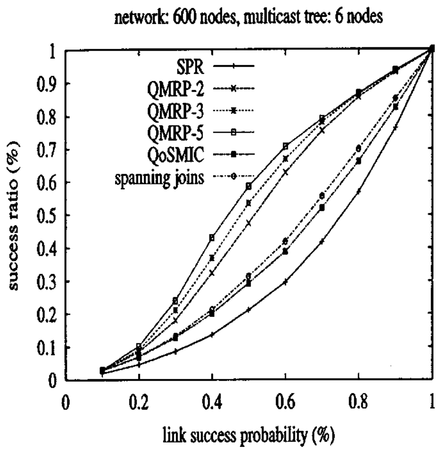
The following multicast routing protocols were simulated: SPR, QMRP-m ($m = 2, 3$, and 5), QoSMIC, and *spanning-joins protocol*. The maximum branching degree of QMRP-m is 10, i.e., a node with degree greater than 12 that enters the MP state limits the submission of REQUEST messages to only 10 of its neighbors. For QoSMIC, the local search and the tree search are implemented as sequential procedures in our simulation[9] —the tree search is conducted only when the local search fails. This results in minimizing the overhead of QoSMIC, but may introduce additional delay to the Join operation when the local search fails. *Directivity*, *local minima*, and *fractional choice* [8] were also implemented. For spanning joins, we implemented its directed flooding version, called *directed spanning joins* in [7]. We assume a unicast routing protocol providing the shortest path in term of hops between each pair of nodes. The simulations were conducted on power-law network topologies [15] (Figs. 8, 10, and 11) and Waxman network topologies [18] (Figs. 12 and 13).

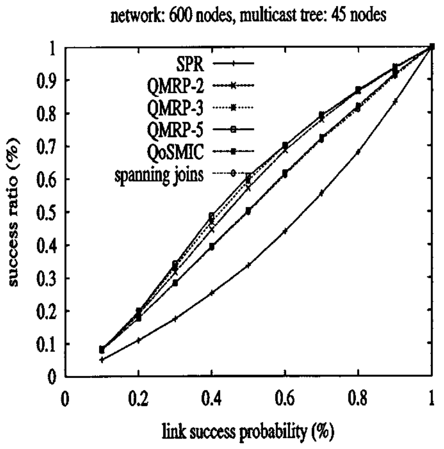
The power-law topologies are based on the results reported in [15], which showed that the node degrees in the Internet obey a power log law: most nodes have small degrees, and a small number of nodes have large degrees; as the degree increases, the number of nodes with that degree decreases exponentially. We used a topology generator described in [19]. For the Waxman-based topologies, we used the Waxman method [18] 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.

Networks with 600 nodes were used in the simulation. Each point in the figure is the result of 60 000 simulation runs conducted on six different networks. In each simulation run, a random multicast tree was generated and a new member out of the tree was randomly selected. We used three tree sizes: 6, 45, and 180 nodes (the tree size includes internal nodes that are not members of the multicast group). The state of each *directed* link[10] 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 routing protocols were then used to find a feasible branch

[9]The local search and the tree search of QoSMIC were implemented as parallel procedures in [17].

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





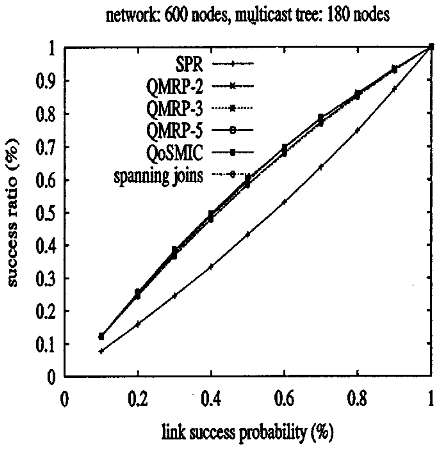


Fig. 8. Success ratios of various routing algorithms with respect to small, medium, large multicast trees, using power-law topologies.

for the new member. For most data points plotted, the standard deviation is less than 4% of the data point.[11]

Fig. 8 compares the success ratios of the simulated protocols when the power-law topologies are used. The simulation

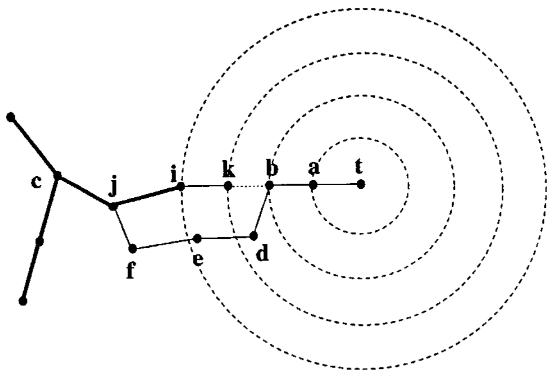


Fig. 9. A flooding scheme does not necessarily achieve better success ratio.

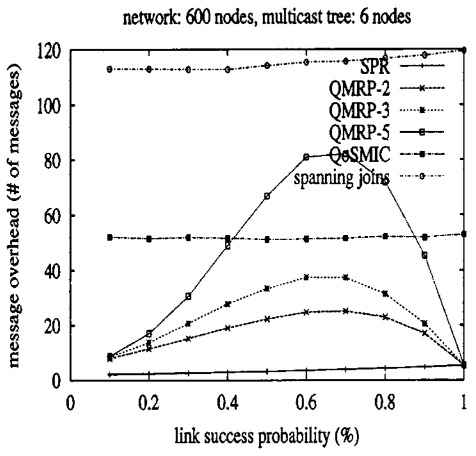
results show that the success ratios of all the QMRP variants are better than those of SPR, QoSMIC, and the spanning-joins protocols. The difference is significant for small groups as shown in the top plot, up to 50% (QMRP-2), 70% (QMRP-3), and 100% (QMRP-5) when comparing to the spanning-joins protocol. The reason is that when the multicast tree is small, QoSMIC and spanning-joins protocols can only select a small number of candidate paths, which limits their ability of finding a feasible path, especially when the link success probability is relatively low. On the contrary, the branching of QMRP is independent of the tree size. Hence, QMRP can search a large number of paths if needed.

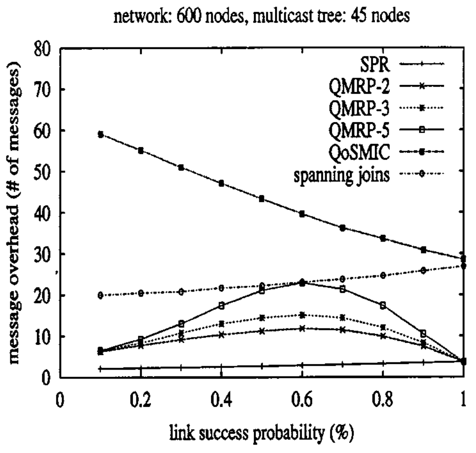
The ability of QMRP-m to achieve better success ratio than a flooding scheme such as spanning joins is surprising and requires elaboration. Consider the example in Fig. 9. Suppose a new member t wants to join an existing multicast tree shown by bold lines. In order to find an on-tree node, the spanning-joins protocol searches increasingly larger areas indicated by the spanning rings in the figure. Once a spanning ring hits an on-tree node i, the search 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.[12] In our simulation, we found that the flooding scheme of spanning joins typically results in a limited number of 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 toward the core. Once it encounters an infeasible link (k, b), it branches and detours to follow other downstream paths such as $b \rightarrow d \rightarrow e \rightarrow f \rightarrow j$ to join the tree. It avoids flooding but can potentially search many 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 search tree, which means flooding in the worst case. By limiting the number of branches, QMRP-m achieves good success ratio while keeping the overhead low. The selection of the branching

[11]For every 50 simulation runs, a *sample* success ratio is calculated. Hence, there are 1200 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 $\pm 8\%$ of the mean.

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





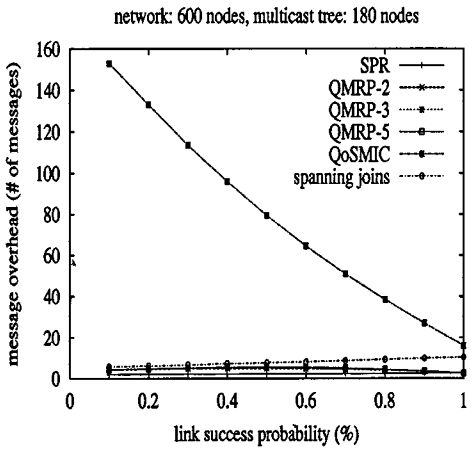
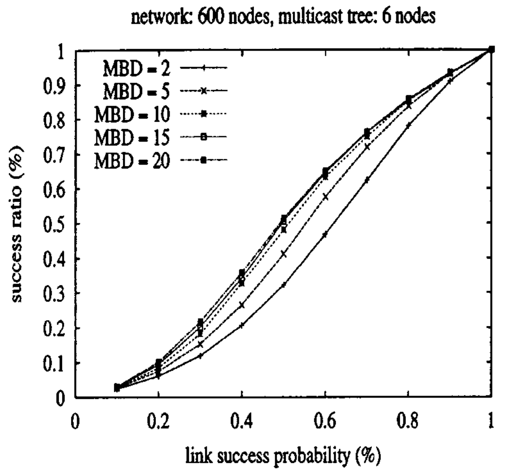


Fig. 10. Message overhead of various routing algorithms with respect to small, medium, large multicast trees, using power-law topologies.



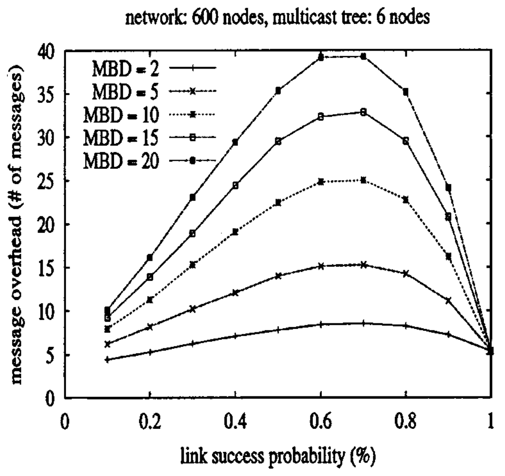


Fig. 11. Comparing QMRP-2 with different MBDs.

parameter m enables us to make a tradeoff between success probability and message overhead. We believe that $m = 2$ represents the most reasonable selection.

Fig. 10 compares the message overhead of the protocols when the power-law topologies are used. As shown in the top plot, the spanning-joins protocol has high message overhead for small multicast groups because it floods until reaching the tree. For large, dense groups, its overhead is low because flooding in small neighborhoods typically hits the tree. The overhead of QoSMIC is high when the multicast group is large and the link success probability is low, as shown in the bottom plot. Under such conditions, both local search and tree search have to be executed and the tree search is expensive. When the link success probability increases, the chance for the local search to succeed increases, which lowers the chance for the tree search to be carried out and thus leads to lower overhead. From the figure, QMRP-3 and QMRP-2 have much lower overhead than spanning join and QoSMIC. Although the overhead is significantly higher than that of SPR, it is worth mentioning that for join requests, which SPR is able to find feasible paths for, QMRP-m behaves just like SPR and thus has the same overhead. The two reasons QMRP-m achieves low overhead for small values of m are: 1) it becomes SPR if the shortest path succeeds, and 2) it enters the MP state only when necessary; and it allows only limited number of nodes to enter the MP state. Of course, when m increases so does the overhead.

Fig. 11 provides a case study on how the maximum branching degree (MBD) affects the performance of QMRP-2. As expected, a larger MBD results in a better success ratio and larger overhead due to more branches from the MP-state nodes. As MBD increases, the rate of improvement on success ratio decreases. Therefore, a medium MBD such as 10 provides

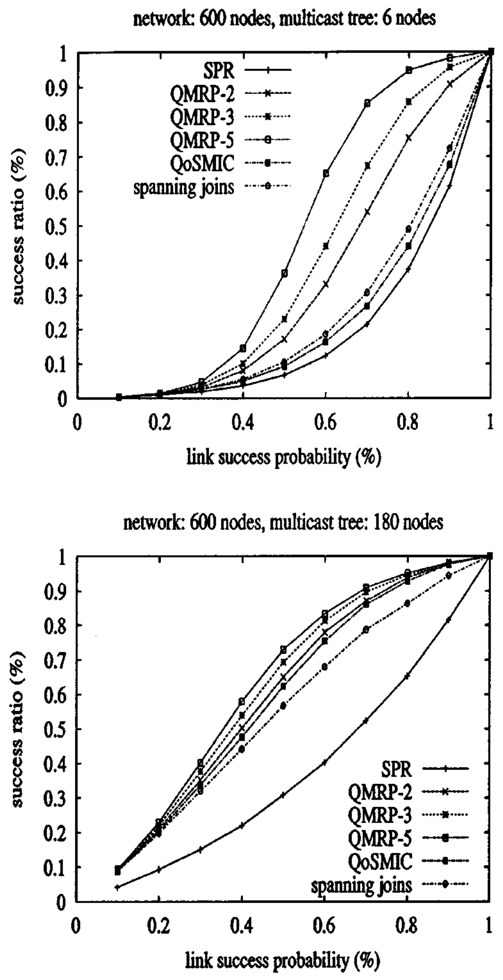


Fig. 12. Success ratio when Waxman topologies are used.

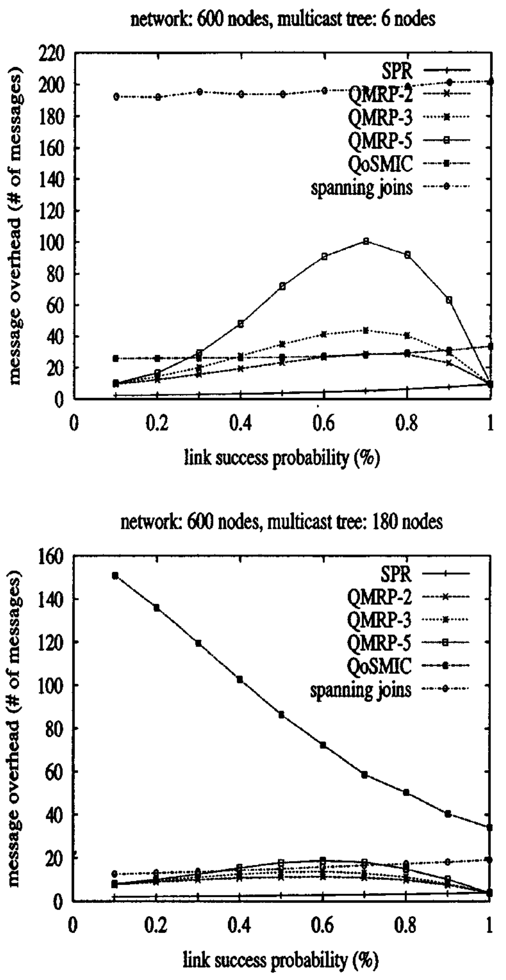


Fig. 13. Message overhead when Waxman topologies are used.

a good tradeoff between the success ratio and the message overhead.

We repeated the simulation on Waxman topologies and similar results were observed. Fig. 12 compares the success ratios of the protocols on 600-node Waxman networks with an average node degree of 3.5. QMRP-m achieves better success ratios than the other three protocols. In Fig. 13, we compare the message overhead of the protocols on the same networks. QMRP-2 has an overhead that is smaller than or comparable to that of QoSMIC and spanning join. Note that although the difference here is smaller than in the power-law topology, neither of the other protocols presents acceptable overhead for all parameter selections, and they are all inferior to QMRP-m in their success ratios.

VI. Conclusion

We proposed QMRP—a new QoS-aware multicast routing protocol—and showed its superior performance in terms of high success probability and low message overhead. However, there are still several open problems and research directions. QMRP-m increases the success ratio for finding a route by examining multiple paths that may be up to m hops longer than the shortest path. When many multicast trees exist in the same network region, the cross-effect of the extra network utilization is an interesting research direction. Furthermore, it would also be interesting to examine the need for rerouting in highly dynamic scenarios where users frequently join and leave. Other important open problems are the adaptation of QMRP to additive requirement, such as delay and loss probability, and the minimization of the state the algorithm is required to keep in the intermediate network nodes.

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*, Sept. 1993.

[3] T. Ballardie, P. Francis, and J. Crowcroft, “An architecture for scalable inter-domain multicast routing,” in *Proc. ACM SIGCOMM*, Sept. 1993, pp. 85–95.

[4] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, “An architecture for wide-area multicast routing,” in *Proc. ACM SIGCOMM*, Aug. 1994, pp. 126–135.

[5] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, “Multicast routing for multimedia communication,” *IEEE/ACM Trans. Networking*, June 1993.

[6] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves, “A source based algorithm for delay-constrained minimum-coat multicasting,” in *Proc. IEEE INFOCOM 95*, Boston, MA, Apr. 1995.

[7] K. Carlberg and J. Crowcroft, “Building shared trees using a one-to-many joining mechanism,” *Computer Commun. Rev.*, pp. 5–11, Jan. 1997.

[8] M. Faloutsos, A. Banerjea, and R. Pankaj, “QoSMIC: Quality of service sensitive multicast internet protocol,” in *Proc. SIGCOMM’98*, Sept. 1998.

[9] H.-Y. Tyan, J. Hou, B. Wang, and Y.-M. Chen, “QoS extension to the core based tree protocol,” in *Proc. NOSSDAV’99*, 1999.

[10] M. Handley and V. Jacobson, “SDP: session directory protocol (draft 2.1),” in *Internet Draft—Work in Progress*, Feb. 1996.

[11] D. G. Thaler and C. V. Ravishankar, “Distributed center-location algorithms,” *IEEE J. Select. Areas Commun.*, vol. 15, pp. 291–303, Apr. 1997.

[12] Y. Shavitt, “Burst control in high-speed networks,” Ph.D. dissertation, Technion—Israel Inst. Technol., Elect. Eng. Dep., Technion City, Israel, June 1996.

[13] I. Cidon, R. Rom, and Y. Shavitt, “Multi-path routing combined with resource reservation,” in *Proc. IEEE INFOCOM’97*, Apr. 1997, pp. 92–100.

[14] S. Chen and K. Nahrstedt, “Distributed QoS routing in ad-hoc networks,” *IEEE J. Select. Areas Commun., Special Issue on ad-hoc Networks*, Aug. 1999.

[15] M. Faloutsos, P. Faloutsos, and C. Faloutsos, “On power-law relationships of the internet topology,” in *Proc. ACM SIGCOMM 1999*, Aug. 1999.

[16] Y. Xiong and L. G. Mason, “Restoration strategies and spare capacity requirements in self-healing ATM networks,” *IEEE Trans. Networks*, vol. 7, pp. 98–110, Feb. 1999.

[17] S. Chen, K. Nahrstedt, and Y. Shavitt, “A qosaware multicast routing protocol,” in *Proc. IEEE INFOCOM 2000*, Mar. 2000.

[18] B. M. Waxman, “Routing of multipoint connections,” *IEEE J. Select. Areas Commun.*, pp. 1617–1622, Dec. 1988.

[19] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, “On the placement of internet instrumentation,” in *Proc. IEEE INFOCOM 2000*, Mar. 2000.

[20] R. G. Busacker and T. L. Saaty, *Finite Graphs and Networks: An Introduction with Applications*. New York: McGraw-Hill, 1965.

[21] Y. K. Dalal and R. M. Metcalfe, “Reverse path forwarding of broadcast packets,” *Commun. ACM*, vol. 21, no. 12, pp. 1040–1048, 1978.

[22] S. Verma, R. K. Pankaj, and A. L. Garcia, “Performance of QoS based multicast routing algorithms for real-time communication,” *Performance Evaluation J.*, vol. 21, 1998.

Shigang Chen (M’99) received the B.S. degree from the University of Science and Technology of China; the M.S. degree from the University of Illinois at Urbana-Champaign (UIUC), both in computer science, and the Ph.D. degree in computer science from UIUC in May 1999.

In June 1999, he joined the Cisco Company. His current research interests are in the area of quality of service, multimedia networking, resource management, distributed computing, and network security.

Klara Nahrstedt (S’93–M’95) received the A.B. and M.Sc. degrees in mathematics from the Humboldt University, Berlin, Germany, and the Ph.D. degree in computer science from the University of Pennsylvania.

She is an Assistant Professor at the University of Illinois at Urbana-Champaign, Computer Science Department, where she does research on quality of service(QoS)-aware systems with emphasis on end-to-end resource management, routing and middleware issues for distributed multimedia systems. She is the coauthor of the widely used multimedia book *Multimedia: Computing, Communications and Applications* (Prentice-Hall).

Dr. Nahrstedt is the recipient of the Early NSF Career Award, and the Junior Xerox Award for Research Achievements.

Yuval Shavitt (S’89–M’96) received the B.Sc. degree in computer engineering (*cum laude*), the M.Sc. degree in electrical engineering, and the D.Sc. degree from the Technion–Israel Institute of Technology, Haifa, Israel, in 1986, 1992, and 1996, respectively.

From 1986 to 1991, he served in the Israel Defense Forces, first as a System Engineer and the last two years as a Software Engineering Team Leader. He spent the summer of 1992 at IBM T. J. Watson Research Center. After graduation he spent a year as a Postdoctoral Fellow at the Department of Computer Science at Johns Hopkins University, Baltimore, MD. Since 1997 he is a Member of Technical Stuff at Bell Labs, Lucent Technologies, Holmdel, NJ. His recent research focuses on active networks and their use in network management, QoS routing, and Internet mapping and characterization.