QoS Multicast Routing with Heterogeneous Receivers

King-Shan Lui
Department of Electrical and Electronic Engineering
The University of Hong Kong, Hong Kong

Jun Wang, Li Xiao, Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana Champaign, Urbana, IL 61801, USA

***Abstract*—When supporting source-specific heterogeneous-receiver multimedia applications, a multicast tree is built among a source and the receivers such that the path from the source to each receiver satisfies the delay and bandwidth constraints. To optimize the network usage, it is desirable to find a multicast tree that minimizes the total bandwidth used while satisfying the different delay and bandwidth requirements of the receivers. For scalability reason, the desired protocol should require little or minimum storage in the sender and other on-tree routers. Moreover, to allow dynamic member join or leave, a receiver-initiated approach is more appropriate. In this paper, we describe our receiver-initiated QoS multicast protocol that aims at reducing the bandwidth used in building a multicast tree for heterogeneous receivers by actively identifying better sub-optimal paths. Our protocol does not require additional information to be stored in the on-tree routers, and it is able to construct a better sub-optimal tree than existing protocols.**

I. Introduction

To support source-specific multimedia applications such as video broadcast and distance learning in a network, a multicast tree that is rooted at a source and spans over a set of receivers is built. As different receivers may have different processing speeds and local network constraints, they may have different delay and bandwidth requirements. This kind of receiver heterogeneity implies that the tree should be built and link bandwidth should be reserved in a way that the tree path from the source to each receiver satisfies the delay and bandwidth constraints of that receiver. To minimize the resource used in the network, a multicast tree should be built using the least amount of total reserved bandwidth. This problem is NP-complete in nature [1] and a heuristic approach is used to solve the problem.

Multicast protocols can be categorized into *sender-initiated* and *receiver-initiated*. In a sender-initiated protocol, the sender knows the set of receivers and initiates the multicast tree construction process. On the other hand, in the receiver-initiated approach, the sender may not know every receiver and a receiver sends out a join request when it wants to join a multicast group. The receiver-initiated approach is more flexible since it does not require the sender to keep the receiver information and allows dynamic member join and leave operations.

In this paper, we describe our novel receiver-initiated QoS multicast protocol to support source-specific multicast with heterogeneous receivers. There are two phases in our protocol. In the first phase, the *active finding* phase, our protocol aims at finding a path, that spans out from the existing multicast tree, to support the receiver. Any existing heterogeneous receiver-initiated QoS multicast can be used in this phase. In the second phase, the *active optimization* phase, our protocol actively tries to reduce the total bandwidth used in the tree by finding another path for the new receiver to join. The second phase does not require additional information and so the information kept in each on-tree node is minimal. Since we use a new path for the receiver to join in the second phase only when we are sure that the new path is more sub-optimal, the performance of our protocol is no worse than other existing protocols. Our simulations demonstrate the effectiveness of the active optimization phase.

This work was supported by grant NSF ANI 00-73802. The opinions and findings of this paper do not necessarily reflect those of the funding agent.

This paper is organized as follows: we describe the related work and the model in Sections II and III respectively. Section IV describes the active optimization phase. We presents our simulation result in Section VI and conclude in Section VII.

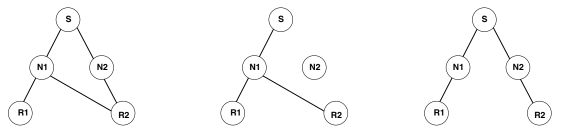
II. Related Work

Heterogeneous receiver multicast has been studied in the application level and also in the network level. There is a lot of work at the application level, which mainly focuses on video multicasting. Examples are [2], [3], [4].

Our paper studies the QoS multicast problem from the network layer perspective. That is, we aim at finding a multicast tree that can support heterogeneous QoS requirements of the receivers. Finding a minimum cost multicast tree, which is called the *Steiner Tree*, is NP-complete. There are many heuristics in finding a sub-optimal Steiner tree [5]. When QoS is considered other than connectivity, the multicast tree construction becomes even more complicated [6].

The research presented in [7], [8] studied how to build multicast trees for video or multimedia streams. They considered only bandwidth but not delay constraint in their routing mechanisms. The problem of constructing a tree that satisfies both delay and bandwidth requirements of heterogeneous receivers was considered in [9], [10], [11], [12]. The approach in [11] is not receiver-initiated. In [9], [10], [12], when a receiver wants to join a multicast group, it finds a feasible path from the sender to itself and sends a *join* message towards the sender along the reversed path. When an on-tree router receives the join message, it determines how to setup a path from the sender to itself. Usually, if the on-tree router realizes that the existing path from the sender to itself and to the new receiver satisfies the request of the new receiver, the on-tree router will reply to the receiver that the join request is successful. In this case, the path from the sender to the on-tree router is used for multiple receivers so that resource used can be optimized.

Unfortunately, the path found by the receiver may not be the most optimal path. For example, Figure 1(a) shows a simple network, where S is the sender and $R1$ and $R2$ are two receivers. When $R1$ uses the path $S \rightarrow N1 \rightarrow R1$ and $R2$ uses the path $S \rightarrow N1 \rightarrow R2$ to join the multicast group respectively, $N1$ can "merge" the two paths starting from itself to the receivers and the resultant tree is optimal (Figure 1(b)). However, when $R1$ uses the path $S \rightarrow N1 \rightarrow R1$ and $R2$ uses the path $S \rightarrow N2 \rightarrow R2$ to join the multicast group respectively, the constructed tree will not be optimal since the two paths do not pass through any common on-tree router and no router can join or optimize the paths among them.



(a) Whole Network (b) Optimal Tree (c) Inoptimal Tree

Fig. 1. Example

In a receiver-initiated approach, this suboptimal situation may happen frequently because a new receiver is not aware of the bandwidth reserved for the multicast group that it is joining. For example, suppose that the capacities of all the links in Figure 1(a) are all 1 unit and the bandwidth required of each receiver is 0.6 unit. Then, after $R1$ has set up the path, the bandwidth available on link $(S, N1)$ becomes 0.4 unit, since 0.6 unit is reserved for the multicast tree. The link state advertisement will advertise 0.4 as the new bandwidth of $(S, N1)$ because information of a multicast tree will not be advertised. In this case, when $R2$ tries to join the multicast tree, it will not find the path $S \rightarrow N1 \rightarrow R2$ because $R2$ thinks that the path does not have sufficient bandwidth. As a result, we would end up having a suboptimal multicast tree as in Figure 1(c).

Our protocol solves the problem by actively finding a more sub-optimal path ($S \rightarrow N1 \rightarrow R2$) after a feasible path ($S \rightarrow N2 \rightarrow R2$) is found based on only local state information in each on-tree router. Therefore, our protocol does not require additional information to be kept in on-tree routers but it is able to build a more sub-optimal tree.

III. MODEL

A network is modeled as a digraph $G = (V, E)$, where V is the set of nodes and E is the set of links among the nodes in V. A link from node x to y is represented as (x, y). We assume that a link is two-way, that is, if (x, y) in E, (y, x) also in E. Each link has two parameters associated with it: delay and bandwidth available. We use $d(x, y)$ and $c(x, y)$ to represent the delay and the bandwidth of link (x, y), respectively. A path from node x to node y in G is denoted as $[x, y]$. If a link (i, j) is on the path $[x, y]$, we write $(i, j) \in [x, y]$. $(i, j) \notin [x, y]$ if $[x, y]$ does not pass through (i, j). The delay of $[x, y]$, denoted as $d[x, y]$, is the sum of the delays of the links along the path. The capacity of $[x, y]$, denoted as $c[x, y]$, is the minimum bandwidth of the links along the path. That is, $d[x, y] = \sum_{(i,j)\in[x,y]} d(i, j)$ and $c[x, y] = min_{(i,j)\in[x,y]} c(i, j)$.

A source-specific multicast group M is defined by a unique source $s \in V$ and a set of receivers $R \subset V$, where each receiver $r \in R$ has an end-to-end delay requirement $D(r)$ and a bandwidth requirement $B(r)$. The problem of multicast routing is to find a tree $T = (V_T, E_T)$, where $R \subset V_T \subseteq V$ and $E_T \subseteq E$, for multicast group M such that the path provided in T from s to each receiver satisfies the delay and bandwidth requirement of that receiver. Formally, for each $r \in R$, the path $[s, r]$ in T must satisfy $d[s, r] \leq D(r)$ and $c[s, r] \geq B(r)$.

Sufficient bandwidth must be reserved on T for the multicast. Therefore, the bandwidth needed to be reserved for M on a link e, denoted $B(e)$, is the maximum bandwidth requirement among all receivers r that the link is supporting. Formally, $B(e) = max\{B(r)|e \in [s, r]\}$. The cost of multicast tree T is the sum of the bandwidth needed for all links in E_T. That is, $B(T) = \Sigma_{l\in E_T} B(l)$. An optimal multicast tree is a multicast tree T where $B(T)$ is less than or the same as the cost of other multicast trees for the same multicast group. The problem of finding an optimal multicast tree has been proved to be NP-complete.

We assume that link-state information is available for every node in the network. There exists a unicast QoS routing algorithm, such as [13], in the network to find a feasible path that satisfies certain delay and bandwidth requirement. Different nodes can execute different unicast QoS routing algorithms. There is also a reservation protocol that can reserve bandwidths on a path. However, no node has the knowledge of the multicast tree structure. A node n that is on a multicast tree keeps only up-to-date local state information of the multicast group, which includes (1) the sender of the multicast group (s), (2) its parent ($parent(n)$) and its children ($A(n)$), (3) the bandwidth reserved in the parent interface, that is, $B(parent(n), n)$, denoted B_{parent}, and in each child interface, $B(n, a)$, $a \in A(n)$, and (4) the delay from the sender to n ($d[s, n]$).

IV. ACTIVE OPTIMIZATION

The reason why [9], [10], [12] cannot find an optimal tree in the case as shown in Figure 1 is because their mechanisms implicitly assume that the first on-tree router that receives a join message is the best point to setup another branch in the tree for the new receiver. An observation from Figure 1 is that the best point to setup another branch can be a descendant from the first on-tree router that receives a join message. We derive our protocol by this observation. There are two phases when an on-tree node handles a join request in our protocol: *active finding phase* and *active optimization phase*. In the *active finding phase*, an on-tree node tries to find a feasible path for a receiver to join the tree. Any existing receiver-initiated QoS multicast routing protocol ([9], [10], [12]) can be used. After a feasible path is found, we intentionally try to optimize the tree by finding a more sub-optimal path for the receiver to join the group in the *active optimization phase*. In this phase, the on-tree node checks whether it is more sub-optimal if the receiver joins at one of its children. If so, it asks that child to see whether it is more sub-optimal to join at one of the child's children. The process continues until we find the lowest level node that is more sub-optimal for the receiver to join at that node than any of the children of that node. The information is then sent to the receiver and the receiver can setup a path to that node accordingly. As this active optimization step is an additional procedure to existing schemes to optimize a multicast tree, the tree built by our protocol cannot be worse than the trees built by existing protocols. Whether the active optimization is useful depends on how often it can identify more sub-optimal location for a receiver to join. We use simulation to study how often our protocol can find a more sub-optimal path and how much our protocol can improve in terms of bandwidth usage. We also analyze the storage, computation, and message overheads.

When r wants to join a multicast tree, it finds a path from s to r and sends a join request on the reverse direction along that path. On receiving the join request, an off-tree node would reserve bandwidth for r, if the bandwidth on the link is sufficient. When an on-tree node n receives the join request, it decides whether to accept the join request or not in the active finding process. We denote the tree path from s to n as $[s, n]$ and the the reverse path of the join request traverses from r to n as $[n, r]_{join}$ as shown in Figure 2. Due to space limitation, we describe only the active optimization phase in this paper. We refer interested readers to [14] for the active finding phase.

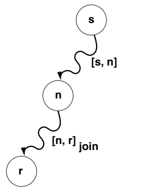


Fig. 2. n receives the join request of r

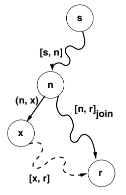


Fig. 3. Path $[s, n] + (n, x) + [x, r]$

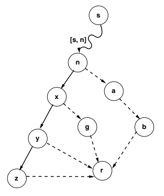


Fig. 4. Finding the best descendant

A. Finding More Sub-Optimal Path

Now, n determines whether the resultant multicast tree will be of better optimality if r joins at one of its descendant node. The cost of supporting r is the sum of the total bandwidth additionally reserved for r on the multicast tree. This includes the bandwidth reserved on $[n, r]_{join}$ and the bandwidth increased on $[s, n]$. The bandwidth that is reserved on $[n, r]_{join}$ can be calculated by $hop[n, r]_{join} * B(r)$. As $[n, r]_{join}$ is stored in the join request message, this value can be calculated locally at n. However, finding the additional bandwidth needed in $[s, n]$ is not always trivial. In order to avoid requiring message probing to obtain the information, we focus on reducing the bandwidth used on the path from n to r.

Let x be a child of n and $[x, r]$ be a path from x to r that does not pass through any on-tree node as shown in Figure 3. For path $[s, n] + (n, x) + [x, r]$ to be a feasible path, the following conditions have to be satisfied:

(1) $c(n, x) \geq B(r) - B(n, x)$ (for bandwidth requirement)
(2) $c[x, r] \geq B(r)$ (for bandwidth requirement)
(3) $d[x, r] \leq D(r) - d(n, x) - d[s, n]$ (for delay requirement)

To find a feasible $[x, r]$ that satisfies conditions (2) and (3), n can invoke a unicast QoS algorithm. As n does not know the whole tree structure, it is possible that the $[x, r]$ found traverses one or more on-tree nodes. We will describe how to handle this case in the next section. In the rest of this section, we assume that $[x, r]$ does not pass through any on-tree nodes. After finding a feasible $[x, r]$, n determines whether $[s, n] + (n, x) + [x, r]$ is more sub-optimal in the sense that the additional bandwidth used in this path for receiver r is smaller than the bandwidth needed in $[s, n] + [n, r]_{join}$.

It is not difficult to see that if $hop[x, r] < hop[n, r]_{join}$, then path $[s, n]+(n, x)+[x, r]$ is more sub-optimal than $[s, n]+[n, r]_{join}$. n uses its own link-state information and the provided QoS routing algorithm to find a path from its child x to the receiver r that satisfies the delay and bandwidth requirement. If the number of hops on $[x, r]$ is smaller than the number of hops on $[n, r]_{join}$, the multicast tree would be more sub-optimal if r joins at x. By using this mechanism, we can find the optimal tree for the example described in Section II. If n finds that no child can provide a more sub-optimal path, it can go ahead and setup $[n, r]_{join}$. If it is more sub-optimal for r to join at a child x, it tries to setup the more sub-optimal path.

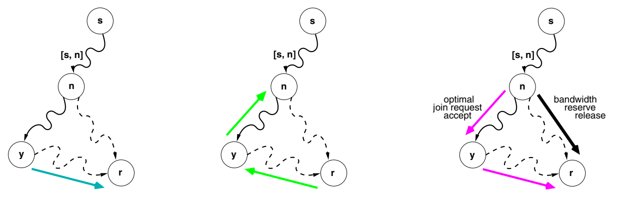
B. More Sub-Optimal Path Setup

After n realizes that it would be more sub-optimal if r joins at its child x, it sends a message to inform x. Then, x can determine whether it is better for r to join at x itself or x's child. If x also finds that it is better for r to join at one of its children instead of itself, it can ask that child to check its children. Finally, there must be a descendant, y, of n such that it is more sub-optimal for r to join at y than at y's children. We refer to this descendant node as the *best descendant*.

Consider the network in Figure 4, where solid lines are tree edges and dashed lines are other links in the network. Suppose that all links have sufficient bandwidth for joining receiver r and delay is not important. Assume that r finds the path $n \rightarrow a \rightarrow b \rightarrow r$ as $[n, r]_{join}$. However, it is not an optimal path and the optimal path is $n \rightarrow x \rightarrow y \rightarrow r$. n finds $[x, r]$ and it realizes that $hop[x, r] = 2 < hop[n, r]_{join}$. n then asks x to check its children. x finds that y is a better position for r to join the tree since $hop[y, r] = 1 < hop[x, r]$. x then sends a message to y to ask y to finds the best child of y for r to join. As y is only one hop away from r, it does not have to check its children.

We now describe how the best descendant sets up the more sub-optimal path from itself to r. Let the best descendant node be y and the path from y to r be $[y, r]_{optimal}$ as shown in Figure 5. After realizing itself as the best descendant, y informs r the optimal path $[y, r]_{optimal}$ (Figure 5(a)). When r receives $[y, r]_{optimal}$, r sends the `optimal join request` message to y along the reverse path of $[y, r]_{optimal}$ (Figure 5(b)). This message contains r, $B(r)$, $D(r)$, and $[y, r]_{optimal}$. An off-tree node processes an `optimal join request` in the same way as it processes a join request message from a joining receiver.

If $[y, r]_{optimal}$ does not pass through any intermediate on-tree node, which is the case illustrated in Figure 5, the `optimal join request` will arrive at y. y then sends the message along the path to s towards n (Figure 5(b)). When n receives the `optimal join request`, it sends `optimal join request accept` to r along path $[n, y] + [y, r]_{optimal}$ to confirm the bandwidth reservation and `bandwidth reserve release` along $[n, r]_{join}$ to release the pending reservation made on that path (Figure 5(c)).



(a) y informs r about $[y, r]_{optimal}$ (b) Sending `optimal join request` (c) n finalizes the reservations

Fig. 5. Optimal Path Setup when $[y, r]_{optimal}$ does not pass through any on-tree node

However, if $[y, r]_{optimal}$ passes through one or more intermediate on-tree node, setting up $[y, r]_{optimal}$ will create loop in the multicast tree and has to be avoided. An example of this scenario is Figure 6(a). In that figure, $[y, r]_{optimal}$ passes through z and setting up that path will create a loop. Let z be the first on-tree node that receives the `optimal join request` that r sends towards y as shown in Figure 6(b). There are two cases: r can join at z and r cannot join at z. r can join at z if the path $[s, z] + [z, r]_{optimal}$ is a feasible path. $[s, z] + [z, r]_{optimal}$ is a feasible path if $B(parent(z), z) \geq B(r)$ and $d[s, z] \leq D(r) - d[z, r]_{optimal}$. Note that $hop[z, r]_{optimal} < hop[n, r]_{join}$, since $[z, r]_{optimal}$ is a subpath of $[y, r]_{optimal}$. Therefore, the bandwidth needed on $[z, r]_{optimal}$ must be less than the bandwidth required on $[n, r]_{join}$. As we require $B(parent(z), z) \geq B(r)$ for r to join at z, no additional bandwidth is needed on $[s, z]$. Therefore, the extra bandwidth needed to build the tree if r joins at z must be less than the extra bandwidth required if r joins the multicast tree at n.

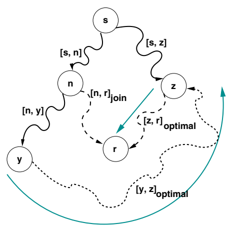
Case I: r can join at z

In this case, z informs r that the optimal join request is accepted. On the other hand, z has to inform y and n that r is not going to join at them. That is, as shown in Figure 6,

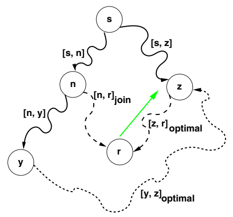
(a) z sends `optimal join request accept` to r using path $[z, r]_{optimal}$ (Figure 6(c)).
(b) z sends `join request clear` to y using the reverse path of $[y, z]_{optimal}$ (Figure 6(d)).
(c) when y receives the `join request clear` message, it sends the message to n using the reverse path of $[n, y]$ (Figure 6(d)).
(d) when n receives the `join request clear` message, it sends `bandwidth reserve release` on $[n, r]_{join}$ (Figure 6(e)).

We now use an example to illustrate the details of this case. Consider the network in Figure 7(a) where a solid edge represents a tree edge and a dash edge represents a non-tree edge. The numbers associated with an edge show the bandwidth reserved on the edge for the multicast group and the residual bandwidth on that link. For example, 2/1 means that there are 2 units of bandwidth reserved for the multicast group and there is 1 unit of bandwidth remaining in the edge. Since a non-tree edge should not have bandwidth reserved for the multicast group, the number representing that is always zero for a dash edge in the figure. Suppose that receiver r wants to join the multicast tree with source s. Note that the link-state information of receiver r reflects only the bandwidth remaining on an edge. That is, it realizes that there is only 1 unit of bandwidth available on (s, z).

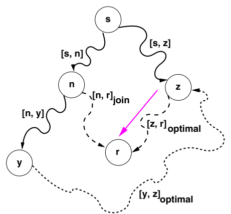
When r wants to join the multicast group, r computes $[s, r]_{join}$ that satisfies the delay and bandwidth requirements. Suppose that $B(r) = 2$ and $D(r) = \infty$. According to the link-state information of r, $s \rightarrow n \rightarrow a \rightarrow b \rightarrow r$ is the only path that satisfies the requirement. Therefore, r sends a join request to b. As b is an off-tree node,



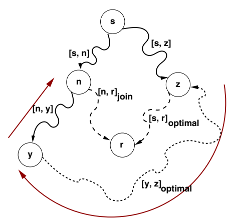
(a) y informs r about $[y, r]_{optimal}$



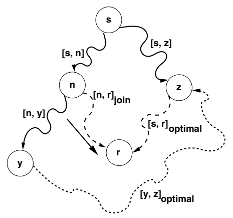
(b) `optimal join request` arrives at z



(c) z sends `optimal join request accept` to r

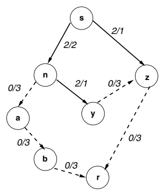


(d) z sends `join request clear` to y and y sends it to n

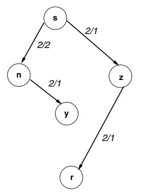


(e) n sends `bandwidth reserve release` to along $[n, r]_{join}$

Fig. 6. Optimal Path Setup when $[y, r]_{optimal}$ passes through on-tree node z and r can join at z



(a) Initial Tree



(b) Resultant Multicast Tree

Fig. 7. Optimal Path Setup Example

it marks the bandwidth on (b, r) as pending and sends the join request message to a. Subsequently, the message will arrive at n.

When n receives the join request message, n determines that r can join the multicast tree at itself. It then goes to the active optimization phase. It finds a feasible path from y to r and $[y, r] = y \rightarrow z \rightarrow r$. As $hop[y, r] < hop[n, r]$, n determines that it is better for r to join the tree at its child y. Therefore, n informs y to check its children.

As y does not have any child, it knows that it is the best descendant and $[y, r]_{optimal} = y \rightarrow z \rightarrow r$. y then informs r the more sub-optimal path $[y, r]_{optimal}$ as illustrated in Figure 6(a).

When r receives the information of $[y, r]_{optimal}$, it sends an `optimal join request` on the reverse path of $[y, r]_{optimal}$ to z as in Figure 6(b). As $B(parent(z), z)$ $(B(s, z))$ is 2, which is the same as $B(r)$, z realizes that r can join the multicast tree at itself. Thus, z replies r with an `optimal join request accept` message (Figure 6(c)). z also sends a `join request clear` message to y and y sends the `join request clear` message to n as shown in Figure 6(d). Upon receiving the `join request clear` message, n knows that r no longer joins the multicast group at itself through $n \rightarrow a \rightarrow b \rightarrow r$ and so it sends a `bandwidth reserve release` message on the path to inform a and b to release the pending bandwidth. The final multicast tree is shown in Figure 7(b).

In this example, our protocol is able to support r by using 2 additional units. If we used $[n, r]_{join}$ to support r, we would need 6 units. Therefore, our protocol finds a more sub-optimal path to support a new receiver.

Case II: r cannot join at z

In this case, z informs r that the optimal join request is rejected. It also informs n that n should set up $[n, r]_{join}$. In this case, we have the same number of messages generated as in Case I. The message flow illustrated in Figure 6 still applies, except that the messages are of different types in Figures 6(c) - (e). Due to space limitation, we refer interested readers to [14] for details.

V. Performance Analysis

In this section, we analyze the performance of the active optimization phase.

Storage

Each on-tree node has to keep only local state information about each multicast tree. We described the information needed in Section III. Therefore, our proposed active optimization phase does not require additional information to work on.

Runtime Complexity

The most expensive step is for n to check whether it is better for r to join at its child. It involves at most one execution of a unicast QoS routing algorithm for each child. Since it is not likely that a node has a lot of children, the runtime complexity of finding the best child is not a serious concern.

Message Complexity

Extra messages are generated when n, the first on-tree node that the `join request` of receiver r encounters on the reverse path of $[s, r]_{join}$, finds that it is more sub-optimal for r to join at one of its descendants. Note that if n determines that $[n, r]_{join}$ is already the best path, no extra message is generated. Note also that we don't increase the messages traversing $[n, r]_{join}$, no matter whether r finally joins at n or not. Therefore, in the following, we describe only the case when n identifies a potentially more sub-optimal path for r to join. Moreover, the messages on $[n, r]_{join}$ are ignored in the following discussion.

Figures 5 and 6 show how to setup more sub-optimal paths. The first extra message generated is for n to find the best descendant y. To find y, n passes a message to its child and that child passes a message to its own child until the message reaches y. There will be $hop[n, y]$ number of such messages generated. When y informs r about the path $[y, r]_{optimal}$, it takes $hop[y, r]_{optimal}$ messages (Figures 5(a) and 6(a)). The total number of messages generated up to this point is $hop[n, y] + hop[y, r]_{optimal}$.

If $[y, r]_{optimal}$ does not pass through any on-tree node, the `optimal join request` message sent by r will travel $hop[n, y] + hop[y, r]_{optimal}$ hops (Figure 5(b)). It takes the same number of hops for the `optimal join accept` to travel back to r (Figure 5(c)). That is, in this case, $2 * (hop[n, y] + hop[y, r]_{optimal})$ messages are generated.

If $[y, r]_{optimal}$ passes through an on-tree node z, no matter whether r can join at z or not, the messages generated will be $2 * hop[z, r]_{optimal} + hop[y, z]_{optimal} + hop[n, y]$ (Figure 6(b) - (d)). Note that in this case, the total number of messages generated is less than the case where $[y, r]_{optimal}$ does not pass through any on-tree node.

Therefore, in the worst case, there will be $3 * (hop[n, y] + hop[y, r]_{optimal})$ extra messages generated. As $hop[n, y] + hop[y, r]_{optimal} < hop[n, r]_{join}$, our protocol does not increase the message complexity asymptotically. In fact, in some protocols that find a feasible path for r to join the multicast group, the messages needed may be more than $3 * hop[n, r]_{join}$, which is more than the worst case of our active optimization phase.

Total Bandwidth Used
The main goal of our protocol is to find a more sub-optimal tree than existing protocols do. As our protocol allows a receiver to join at a different node only when the resultant tree utilizes less bandwidth, our protocol will not build a less optimal tree than existing protocols w.r.t. to the same receiver. On the other hand, as the protocol always selects a path with fewer hops for the receiver to join, the size of the final tree is also reduced.

Optimality vs. Scalability
It is undeniable that our protocol cannot find the most optimal tree. However, there is a tradeoff between optimality and scalability. For example, to make the optimization more effective, instead of only searching for the *best descendant*, we can search the whole tree to find the *best node*. Unfortunately, this extensive search may generate a lot of messages and is not scalable. As our protocol requires only local state information and the message overhead is reasonable, our protocol achieves a good balance between optimality and scalability.

VI. Simulation

To measure how effective the active optimization is, we conduct simulation. We generate 100 different networks for each network size based on the BRITE topology generator and 10 different multicast groups for each network. That is, there are 1000 different instances for each network size. The network sizes are 50, 100, 150, 200, 250, and 300 while the average node degree is 4. The size of a multicast group is 10-20% of the network size. The link bandwidths and link delays are generated using a uniform distribution within a certain range. The ranges of link bandwidth and link delay are 3-10 and 2-10, respectively. We use the algorithm in [13] as the QoS routing algorithm.

Whether the bandwidth required for a new receiver can be decreased depends on whether the active optimization phase finds a more sub-optimal path for the receiver to join. Table I shows how often the protocol is able to realize a more sub-optimal path than the path found in the active finding phase and shows how much bandwidth our protocol can optimize. The 2nd column presents the percentage of the multicast groups that the active optimization phase identifies a better path for one or more receivers in that group. That is, among the 1000 multicast groups generated for networks of size 50, 583 groups are benefited by our protocol. The 3rd column presents on average, how many members (in terms of percentage) within the same multicast group, that successfully join the tree, are benefited. That is, on average, our protocol sets up more sub-optimal paths for 22% of the members in a multicast group of network size 50. We also measure how much bandwidth is saved for each optimized path. The percentage saved is calculated by the following formula $\frac{\text{original bandwidth used - optimized bandwidth used}}{\text{original bandwidth used}}$.

As illustrated in Table I, our protocol saves more than 40% of the bandwidth used when the active optimization phase successfully sets up a more sub-optimal path.

Size	% of Groups	% of Members	% saved
50	58.3	22	45.5
100	78.1	21	46.2
150	82.7	18	46.4
200	89.4	16	44.9
250	93.0	17	44.7
300	97.0	15	44.3
average	83.1	18.2	45.3

TABLE I
FREQUENCY OF FINDING MORE SUB-OPTIMAL PATH

As described in Section IV, the first on-tree node a join request encounters attempts to setup a more sub-optimal path only when it realizes that it is more sub-optimal for the receiver to join at one of its child. Since the more sub-optimal path is found by using up-to-date link-state information, as long as the network state does not change, the more sub-optimal path can be setup successfully if it does not pass through other on-tree node. However, if the more sub-optimal path passes through one or more on-tree nodes, the setup may not be successfully (Case II of Section IV-B). In this case, although extra messages are generated, bandwidth is not saved. This is undesirable and we measure how often this happens. Among all the attempts in setting up a more sub-optimal paths, less than 4% (2.5% in average) of them are unsuccessful. As we can save more than 40% of the bandwidth in a successful attempt, this unsuccessful rate is not serious.

VII. Conclusion

In this paper, we study the problem of finding an optimal multicast tree for heterogeneous receivers. We demonstrate why existing protocols fail to find an optimal path for a joining receiver. We then describe our protocol that aims at solving the problem by actively finding a more sub-optimal path. We show that our protocol requires only local state information and never produces a tree that is worse than a tree built using existing protocols with reasonable message overheads. Our simulation results show that our protocol successfully improves the bandwidth usage in building a multicast tree.

References

[1] P. Winter, "Steiner Problem in Networks: A Survey," *Networks*, pp. 129 – 167, 1987.
[2] S. McCanne and V. Jacobson, "Receiver-driven Layered Multicast," in *ACM Proceedings of the SIGCOMM '96*, 1996, pp. 117 – 130.
[3] S. McCanne, M. Vetterli, and V. Jacobson, "Low-Complexity Video Coding Receiver-driven Layered Multicast," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, pp. 983 – 1001, 1997.
[4] S. Servetto et. al., "Video Multicast in (Large) Local Area Networks," in *IEEE Proceedings of the INFOCOM '02*, 2002.
[5] L. Sahasrabuddhe and B. Mukherjee, "Multicast Routing Algorithms and Protocols: A Tutorial," *IEEE Network*, vol. 14, no. 1, pp. 90 – 102, Jan/Feb 2000.
[6] B. Wang and J. Hou, "Multicast Routing and Its QoS Extension: Problems, Algorithms, and Protocols," *IEEE Network*, vol. 14, no. 1, pp. 22 – 36, Jan/Feb 2000.
[7] N. Maxemchuk, "Video Distribution on Multicast Networks," *IEEE Journal of Selected Areas in Communications*, vol. 15, no. 3, pp. 357 – 372, Apr. 1997.
[8] N. Shacham and J. Meditch, "An Algorithm for Optimal Multicast of Multimedia Streams," in *IEEE Proceedings of the INFOCOM '94*, 1994, pp. 856 – 864.
[9] S. Chen, K. Nahrstedt, and Y. Shavitt, "A QoS-Aware Multicast Routing Protocol," in *IEEE Proceedings of the INFOCOM '00*, 2000, pp. 1594 – 1603.
[10] A. Gei and M. Gerla, "Receiver-Initiated Multicasting with Multiple QoS Constraints," in *IEEE Proceedings of the INFOCOM '00*, 2000, pp. 62 – 70.
[11] B. Wang and J. Hou, "QoS-Based Multicast Routing for Distributing Layered Video to Heterogeneous Receivers in Rate-based Networks," in *IEEE Proceedings of the INFOCOM '00*, 2000, pp. 480 – 489.
[12] D.-N. Yang, W. Liao, and Y.-T. Lin, "MQ: An Integrated Mechanism for Multimedia Multicast," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 82 – 97, Mar. 2001.
[13] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, Sept. 1996.
[14] K. Lui, J. Wang, X. Li, and K. Nahrstedt, "Qos multicast routing with heterogeneous receivers," Tech. Rep., Department of Computer Science, University of Illinois at Urbana-Champaign, Jan. 2003.