Core-based Routing with QoS Support for Distributed Interactive Multimedia Applications

Wanida Putthividhya, Wallapak Tavanapong, and Johnny S. K. Wong

Department of Computer Science, Iowa State University, Ames, Iowa 50011-1040, USA

Summary
Distributed interactive multimedia applications such as video conferencing and virtual collaboration applications typically involve frequent interactions among multiple distributed parties. These applications require an efficient networking support such as core-based routing with Quality of Service (QoS) support. Typical core-based routing selects one router as a *core* for a multicast group and builds a single multicast tree rooted at the core to deliver data to the entire multicast group. Routing with a single core, however, may not satisfy QoS requirements of many distributed group members. Hence, we introduce in this paper new QoS core-based routing called *Core Set Routing* that utilizes the smallest set of cores that can satisfy QoS requirements of as many group members as possible. As part of this paradigm, we present our new distributed core selection protocol and multicast tree construction protocol that ensures loop-free routing and offers several desirable properties. We discuss a novel QoS constrained path search protocol and a protocol for handling member and sender dynamics. Last, we present the performance of our core selection and multicast tree construction protocols. The simulation results demonstrate that our core set routing can satisfy significantly more group members compared with a recent QoS based routing using a single core. Besides, our tree construction protocol does not impose much overhead on the networks.

Key words:
Quality of Service, Multicast, Core-based Routing, Core Selection, Multicast Tree Construction.

1. Introduction

In recent years, a great deal of efforts has been made to support distributed interactive multimedia applications involving multiple senders and receivers. These applications include video conferencing, virtual collaboration, multiparty online role playing games, just to name a few. These applications have a wide range of numbers of users (e.g., from several to over 100,000 users as recently reported in some online playing game). Efficient network support for these applications is necessary. Research in Quality of Service (QoS) and core-based routing fulfill different but complementary needs of these applications. QoS addresses requirements of multimedia applications by enforcing end users' specifications of their desired service quality such as throughput, end-to-end delays, and delay jitters. Core-based routing offers scalability since a number of senders share the same multicast tree rooted at a single router chosen as the *core*. The tree spans all group members of a multicast group. The senders unicast the data for the multicast group towards the core. The core, then, forwards the data to all the group members via the multicast tree.

Routing with a single core, however, may not satisfy QoS requirements of many distributed group members. As a result, the service can be seriously affected. In this paper, we introduce Core Set Routing, a new QoS core-based routing approach that aims to achieve all of the following design goals.

(i) **Ease of QoS specification:** Ability to enable group members to specify their QoS requirements easily as well as to accommodate QoS requirements for different types of distributed interactive multimedia applications.
(ii) **Efficiency and scalability:** Ability to route multicast data with low routing overhead and scale well for large networks and large group sizes.
(iii) **Robustness:** Ability to handle core failures.
(iv) **Loop-freedom:** Ability to always construct loop-free multicast trees.
(v) **Operability:** Ability to operate on top of any existing unicast routing algorithm.

To achieve our first goal, we introduce an *application level service class* framework. In this framework, a multicast group is associated with a set of pre-defined service classes. Each of the classes specifies a different bound for the same end-to-end QoS metric such as delays or transmission bandwidth. A group member (user) selects one of these service classes to indicate the desired service quality. The number of service classes and the bound for each class depend on the types of applications. Table 1 shows possible end-to-end delay classes for virtual collaboration applications. Except for the best effort class, the bounds specified in the rest of the service classes are acceptable to users of virtual collaboration applications [30]. Unlike network-level service classes in

Differentiated Services [4] and the reduced service-set architecture [24] that are transparent from users, in our framework, users are aware of the set of service classes offered by the application level.

Table 1: Example set of service classes for virtual collaboration applications

Service Class	Upper-bound Delay(ms)
very_short_delay	66
short_delay	83
medium_delay	100
best_effort	∞

To achieve efficiency and scalability, we formulate a new *QoS core selection problem* to select the smallest set of cores for a multicast group such that the end-to-end QoS requirements of as many group members can be satisfied. We propose a core selection algorithm to address the problem. Our work is different from existing core selection algorithms that choose only a single core per group [14, 8, 28, 23, 20] and those that use multiple pre-determined cores per group without QoS support [5, 17, 31].

This paper is a summary of our new core set routing paradigm, which includes the application level service class framework, the formulation of the core selection problem, the core selection algorithm [26], the distributed core selection protocol, the multicast tree construction algorithm [27]. In particular, we review the above work and discuss the following extensions. We improve the distributed core selection protocol to reduce protocol overhead. We discuss a multicast tree maintenance protocol. We introduce a protocol for finding a QoS constrained path between a pair of source and destination nodes. We discuss a mechanism for handling dynamics of senders and members of the group. Finally, we perform over 5000 simulation runs to evaluate our core set routing paradigm.

The remainder of the paper is organized as follows. In Section 2, we provide background on core-based routing. In Sections 3 and 4, we present our core set routing paradigm in detail and its performance. Finally, we offer concluding remarks and discuss future work in Section 5.

2. Background

IP Multicast and many aspects of QoS research such as unicast routing with QoS support, INTSERV, and DIFFSERV have been intensively studied in the late 1990s. Application Layer Multicast (ALM) [13, 21, 2, 22, 32] has gained significant interests in the subsequent years due to slow deployment of IP Multicast routers and other issues. In ALM, hosts implement multicast functions instead of routers. ALM is not as effective in utilizing networking resources as IP Multicast does, but ALM is a solution that can be deployed currently. It is not clear which of the two multicast approaches would be a sustainable solution in the future. Hence, both IP Multicast and ALM warrant further studies. We focus on IP Multicast in this paper.

Given a broad range of research in IP Multicast and QoS, we focus our background discussion on existing works in core-based routing and QoS support for core-based routing since they are most relevant to our work. Readers interested in other aspects of IP Multicast and QoS are referred to [11, 10].

Four main research issues in core-based routing in IP multicast are *core selection* [8, 28, 23, 20, 14, 17, 5], *multicast tree construction* [17, 5, 1, 16], *membership handling* [9, 19, 12], and *tree/core migration* [18]. Protocols in the *core selection* category locate a single core or multiple cores for a multicast group since the location of a core significantly determines routing overhead. The location of any core of the group is crucial since it affects the performance of a multicast tree rooted at the core. After the selection of the core(s), protocols in the *tree construction* category construct one loop-free multicast tree rooted at every core of the group. Each multicast tree spans all the group members. The tree is used by all the senders to multicast data to the group. As new members joining or existing members leaving the group, protocols in the *membership handling* category modify the multicast tree of the group when necessary. Because of membership changes and network dynamics, the core selected initially for the group may no longer be at the optimal location. Protocols in the *tree/core migration* category select a new core when necessary and migrate nodes in the existing multicast tree to the new tree rooted at the new core in an optimal manner. Since we focus on core selection and multicast tree construction in this paper, we discuss the most recent work in these two topics in more details.

2.1 Core Selection

The works in [8, 28, 23, 20] select only a single core per group, aiming to optimize the desired metric of interest such as a tree diameter. These schemes do not take group members' QoS requirement into account. Distributed Core Multicast (DCM) [5] utilizes multiple cores per group to avoid the problem of determining the optimal location of

the single core. DCM employs a hash function to select the set of cores from pre-defined candidate cores. Using multiple cores per group to reduce delays between senders and group members, avoid a single point of failure, and alleviate traffic concentration in the network has been evaluated [31]. DCM and the schemes in [31] do not consider specific QoS requirements of group members.

QoS Core Selection Algorithm (QCSA) [14] is a distributed core selection algorithm that considers group members' QoS requirement. Each candidate core (the router closest to a group member) attempts to construct a path that satisfies the QoS constraints from itself to each of the other candidate cores and uses the maximum number of hops of such paths as its cost. These candidate cores advertise their cost among themselves. The candidate with the smallest cost elects itself as the core of the multicast group. QCSA has the following drawbacks. It only satisfies core-to-end QoS requirements although an end-to-end QoS requirement is more preferable for multimedia applications. Most importantly, QCSA may not satisfy QoS requirements of many distributed group members since only a single core per group is selected.

2.2. Multicast Tree Construction

To construct a multicast tree, two approaches can be considered. One approach builds a multicast tree from the core towards the group members using path information obtained during core selection [14]. The other approach incrementally constructs a multicast tree by attaching group members to the tree one by one using a protocol for handling new members. In this case, core migration is typically employed. Protocols for handling new members can be classified into three categories: single-path search (SP), multiple-path search (MP), and hybrid. A protocol in the SP category searches only one path at a time to find a QoS constrained path[†] to an on-tree node of the existing multicast tree. Once the path is found, the new member is made a child of the on-tree node. Examples of protocols in this category are CBT [1] and PIM [16]. These protocols do not consider users' QoS requirements. A protocol in the MP category concurrently searches multiple candidate paths towards the core to select the best path to attach the new member to the tree. Examples include Spanning-Joins [9], QoSMIC [19], and [29]. QMRP [12] is the protocol in the hybrid category that considers the tradeoff between the search overhead and the latency of finding the path to the on-tree node. In QMRP, the search process starts with the single-path search and switches to the multiple path search when the single-path search fails.

[†] A QoS constrained path is a path between a pair of source and destination nodes that satisfy a desired QoS requirement.

A multicast tree construction protocol in either approach must guarantee loop-free routing. Furthermore, a good protocol should build a tree with desirable properties, for example, a shortest delay tree if delays are the QoS metric of interest. To the best of our knowledge, there is no evidence that clearly indicates the superiority of one approach over another.

3. New Core Set Routing in IP Multicast

We first review our service class framework and the general concept of our core set routing. Then, we discuss a new QoS core selection problem under the service class framework and our distributed core selection protocol. We present our multicast tree construction protocol, and protocols for finding QoS constrained path and handling group dynamics.

3.1 Core Set Routing under Service Class Framework

The service class framework aims to *(1)* accommodate users by requesting for a desired service quality from a set of pre-determined service classes and *(2)* support various types of interactive applications. Each class specifies the bound of the service quality that can be assured to users subscribing to the class.

In our Core Set Routing, we use the closest multicast-capable router of a group member or a sender to represent the corresponding group member or sender. This assumption was used in other previous work as well. Under this assumption, QoS requirements between a member/sender and its designated router can easily be assured because they are typically in the same local network. Therefore, we can focus on a harder problem — guaranteeing end-to-end QoS requirements along the paths between routers representing the senders and group members. Specifically, we use the term *member router* and *sender router* for the router representing a group member and a sender, respectively. In a local network that has both group members and senders in the same multicast group, the closest router of the network acts as both member router and sender router. Fig. 1 highlights the major difference between routing with a single core and our approach.

Fig. 1(a) depicts a scenario that shows three unsatisfied member routers when a single core at the optimal location is used. In this example, these unsatisfied member routers receive multicast data through the bottleneck network. Since this single core is at the optimal location, choosing a different core or different routes from the core will result in more unsatisfied member routers. In contrast, our new technique chooses two cores ($Core_1$ and $Core_2$) for the same

scenario (see Fig. 1(b)) to satisfy every member router. Each core is assigned a disjoint set of member routers. A multicast tree is rooted at each core and spanned all the member routers assigned to the core. Each sender router unicasts data towards every core of the group. Each core, in turn, forwards the data to the assigned member routers via the corresponding multicast tree. In general, we choose the smallest non-empty set of cores to satisfy the QoS requirements of as many group members as possible. If it is possible that one core can satisfy QoS requirements of all the group members, our scheme will attempt to select one core as well.

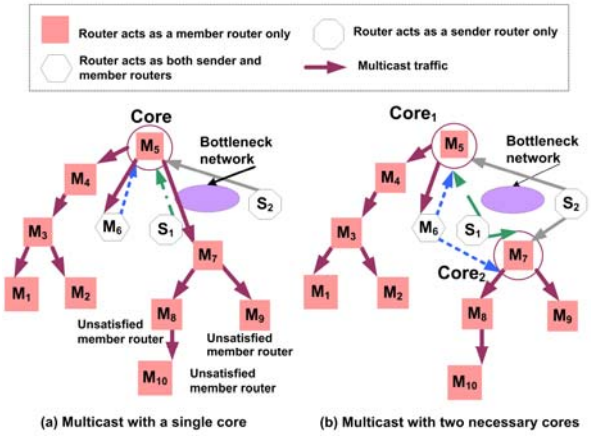


Fig. 1 Routing with a single-core vs. Core Set Routing.

Our scheme has the following advantages. First, it increases the number of group members with satisfied QoS requirements. Second, the cores can be backups for each other should some cores fail. Last, when many new member routers simultaneously join the group, they can be directed to different existing cores to prevent any core from becoming a hot spot. Nevertheless, these advantages do not come for free. The sender-to-core traffic increases proportionally to the number of cores since each sender router has one unicast stream to each core. To minimize the overhead, our scheme selects the smallest non-empty set of cores. In this paper, we focus on one QoS metric (any of end-to-end delays, bandwidth, or loss rates) as typically assumed in several previous works in core-based routing.

3.2 Distributed Core Selection Protocol

We formulated a QoS Core Selection problem and proposed distributed core selection protocol to solve the problem in our previous work. In this section, we briefly review the two concepts and state the improvement we have made to the protocol to reduce the protocol overhead. Due to limited space, readers interested in the problem formulation, the proof, and the protocol in details are referred to Reference [25, 27].

We say that *a member router j covers a member router i subscribing to the service class c* if the member router *i* is able to receive multicast data from all the sender routers through the router *j* within the end-to-end bound of the service class *c*. In other words, the router *j* is capable of being a core for the router *i* for this service class. We define the *class c covering set of j* ($cover_c(j)$) as the set of member routers (subscribing to the service class *c*) that can be covered by the member router *j*.

It is important to understand that if a group member requests for a requirement that is too stringent for the current network condition between the sender routers and its designated member router, the group member cannot be satisfied regardless of which core selection algorithm is used. We call such a group member and its designated member router an *insatiable group member* and an *insatiable member router*, respectively.

Let *SC* be a set of pre-determined service classes offered to a multicast group. Let *R* be a set of all member routers of the multicast group. All the member routers of the group are the only eligible candidate cores for the group. We state our core selection problem as follows.

Problem Statement: *Given SC and the covering set information for each candidate core in R, select the smallest non-empty set* $C \subseteq R$ *such that every satiable group member has its end-to-end QoS requirement satisfied if it receives data through one of the selected core in C.*

Our distributed core selection protocol solves the problem in a distributed manner. Every member router and sender router first registers with a bootstrap router. This is to let member and sender routers learn about each other. At the end of this registration step, every member router knows about the IP addresses of all other member and sender routers and the membership information[‡] of all member routers. Sender routers learn about IP addresses of all member routers. For each service class *sc* (starting from the class with the most stringent QoS requirements), each member router *m (1)* finds the set of candidate cores that can cover itself[§], *(2)* exchanges this information within the multicast group, *(3)* applies our *greedy algorithm* based on *cover heuristic*[**] to select the smallest set of cores for the

[‡] The membership information of a member router tells which service classes the designating members of this router subscribe to.

[§] To be more specific, *m* finds out whether a QoS constrained path from every sender router to *m* via a candidate core exists. This can be done using our QoS constrained path search protocol discussed in Section 3.4.

[**] The cover heuristic selects the candidate core with the largest covering set in each round. This is repeated until all satiable

class *sc*, and *(4)* releases resources reserved along QoS constrained paths via non-selected candidate cores. Repeat these steps for the next less stringent class until all service classes are considered. Each member router combines the core set for every service class using our deterministic *core merging algorithm* [25, 27]. Our core merging algorithm yields the following. First, every member router selects the same smallest set of cores for an entire group. Second, each member router is assigned to only one chosen core.

The previous version of this protocol [27] performs step (1) for all the service classes before performing core selection (i.e., steps (2) and (3)). As a result, too many networking resources are unnecessarily reserved before core selection takes place. Our improvement here is to perform all the three steps one service class at a time.

At the end of the core selection, each core maintains the information of end-to-end QoS assured paths from every sender router to each of its assigned member routers via the core itself. This information has been obtained since step (1) and will be used in the new multicast tree construction protocol discussed next. Our multicast tree construction protocol constructs one multicast tree rooted at each core.

3.3 Multicast Routing with Core Set

We discuss our protocol for constructing a multicast tree, routing multicast data through the constructed tree(s), and maintaining the multicast tree(s).

3.3.1. Multicast Tree Construction Protocol

Since QoS constrained path information has been obtained during the core selection, it is natural to use the path information to construct a multicast tree rooted at each core and spanned all the member routers assigned to the core. To ensure that routing within a multicast group is loop-free, a router being a part of different multicast trees for the same group maintains one routing entry for each of these trees. For ease of exposition, we use end-to-end delays to explain our tree construction protocol. The protocol is also applicable when a different end-to-end QoS metric is used. The routing entry format is <*gid*, *cid*, *in*, *s2c_delay*, *c2e_delay*, *out*> where *gid* is the multicast group ID; *cid* is the IP address of the core of the group; *in* is the incoming interface receiving multicast data from the core *cid* of the group; *sc2_delay* is the longest guaranteed delay among the delays from all the sender routers to the core *cid*; *c2e_delay* is the cumulative guaranteed delay from the core to this router via the current multicast tree; and *out* is the list of interfaces to forward the multicast data. Each routing entry is uniquely identified by *gid* and *cid* to indicate the tree to which this router involves.

Step 1: Each core determines the best path from itself to each of its assigned member routers using the path information maintained at the end of core selection. For the QoS delay metric, the best path from the core to its assigned member router is the shortest core-to-end delay path. For other QoS metrics, the best path must be determined accordingly.

Step 2: Each core independently sends a BUILD message onto the associated best path to each of its assigned member routers. The core fills the source routing option field in the IP header of the BUILD message with the list of the IP addresses of the routers along the corresponding best path. The message is, then, forwarded to the routers in the path in the order specified in the list.

Upon receiving a BUILD message, router *r* takes actions based on the interface the message came from and the cumulative delay specified in the message. If the BUILD message with a shorter cumulative delay came from a different interface, the path along the multicast tree from the core to the router is switched to the one with the shorter delay. The router, then, propagates the updated cumulative delay to every downstream router along the current multicast tree using the UPDATE messages. However, if the BUILD message does not indicate a shorter cumulative delay, the router sends a PRUNE message upstream (onto the interface where the BUILD message came) to eliminate the longer delay path from the core to the router.

3.3.2. Multicast Data Routing

When receiving multicast data packets for the group, each core of the group adds its own identity to the packets. The on-tree router forwards the packet onto the out-going network interfaces indicated in the routing entry corresponding to the group and the core identified in the data packet. This routing method is loop-free and does not incur any duplicate multicast data to arrive at the group members. Furthermore, for the end-to-end delay QoS metric, the path from the core to each of designating member routers assigned to the core is the shortest delay path. Readers, interested in the proof and the detailed multicast tree construction protocol, are referred to Reference [25].

3.3.3. Multicast Tree Maintenance

Tree maintenance takes place after the multicast tree rooted at each of the selected cores has been successfully constructed. Existing multicast tree maintenance protocols

---

member routers with designating members subscribing to this class have been covered.

such as the Hello protocol can be used. That is, an on-tree node periodically sends a HELLO message to its parent node. If no response is received from the parent node within a time period, the sending node assumes that its parent node or the link between itself and the parent node is down. The sending node informs every node in the subtree rooted at itself about the failure. Any member router in the subtree re-joins the multicast group as a new receiver. It is known that the overhead of the HELLO protocol depends on how frequent HELLO messages are sent. A number of improvements can be made to the protocol. For instance, Hello messages can be aggregated across multicast trees of the same group. Each on-tree router determines the set of active neighbors (the parent nodes of this on-tree router in any multicast tree of the group). The router sends HELLO messages to these active neighbors. By properly determining the frequency for sending HELLO messages and determining the set of active neighbors, the tree maintenance overhead can be minimized.

3.4. Protocol for Finding a QoS Constrained Path

The goal of the protocol is to find a path from a sender router to a member router via a candidate core such that the QoS metric along the path satisfies the required bound. Each participating router searches for the QoS constrained path in rounds. In each round, the router simultaneously searches a set of out-going links that have not been searched in the previous rounds. The number of links searched in each round follows that in Fibonacci series (i.e., *1, 1, 2, 3, 5, 8, . . ., x*) such that the total number of searched links equals N where N is the total number of out-going links at this router. The router enters the i^{th} round if all the searches in the $(i - 1)^{th}$ round fail. To prevent searching in a loop, a router that has already started the search for this reservation does not consider subsequent searches for the same reservation.

Suppose a member router r wants to search for a QoS assured path that satisfies the end-to-end delay of *req_delay* seconds from a sender router s to the router r via a candidate core *cc*. The router r sends a reservation message *RESV* (r, *cc*, s, *req_delay*, *acc_delay*, f) onto each of the out-going links selected to be searched in the current round. The first three fields specify the router initiating the message, the candidate core and the intermediate destination, and the final destination. The fourth field indicates the required end-to-end delay. The *acc_delay* is the cumulative delay along the path from the router receiving the message back to the router r as multicast data flow from a core towards a member router. The flag f is initialized to zero by the initiating member router r to indicate that the message should be forwarded towards the intermediate destination (candidate core *cc*) specified in the message. When receiving this message, the candidate core sets the flag f to one to indicate that the message should be forwarded towards the final destination of the message from now on.

This search strategy is a variant of QMRP [12] for handling new joining members. QMRP searches the out-going links of a router in only two rounds. The numbers of links searched in the two rounds are 1 and $N - 1$, respectively. Searching the entire $N - 1$ links in the second round may cause unnecessary control messages and resource reservations. Restricted QMRP addresses these drawbacks by limiting the number of searched paths, which may exclude the paths that can satisfy the QoS requirements of some group members.

3.5. Protocol for Handling Group Dynamics

We briefly discuss techniques for handling group dynamics due to joining or leaving senders and group members in this section. The goal of these techniques is to avoid violating QoS requirements of existing group members and to keep management overhead low. Recall that without a renegotiation mechanism, users whose QoS requirement cannot be satisfied are rejected in the beginning. Hence, only group members whose QoS requirements can be met participate in the multicast. Various scenarios causing group dynamics are listed in Table 2.

For an addition of a new member router or a new sender router, we employ the QoS path search protocol in Section 3.4 to find a QoS constrained path to attach a new node to one of the existing multicast trees of the group. For deletion of an existing member router or a sender router, we update corresponding routing entries and prune the proper paths to the leaving router from the corresponding tree when appropriate. We discuss Case I in more details to convey our ideas. Since handling group dynamics is itself a research topic that requires more investigation to provide a good conclusion, we provide comments on other options and leave the detailed investigation for future work.

Table 2: Scenarios causing group dynamics

Case I: A router wishes to join the group as a new member router because the router is designated by a new member wishing to join the group. This router could be a new router that has not participated in the group before or the router is currently participating as a sender router only.
Case II: An existing member router wishes to leave the group since none of its designating members wants to participate in the group any longer.
Case III: A router wishes to become a new sender router of the group because either this router has not participated in this group before but has a new joining sender or the router is currently participating as a member router, but the member wishes to multicast its data to the group.
Case IV: An existing sender router wishes to stop participating because its designating senders no longer want to multicast data to the group.
Case V: A router wishes to participate in the group as both a member router and a sender router. This case is a hybrid of cases I and III.
Case VI: A router that currently is both member router and sender router of the group wishes to leave the group. This case is a combination of cases II and IV.

Case I: The joining router (i.e., the router that wishes to join the group as a member router) uses the QoS constrained path search protocol discussed in Section 3.4 to search for a QoS assured path from an on-tree node in one of the multicast trees of the group to itself. The joining router gets the information about the core(s) of the group from a bootstrap router. It then sets the final destination of the RESV message to be the root of each of the multicast tree and sets the value of the flag field to one. Upon receiving the RESV message, an on-tree router [††] determines whether the requested QoS metric can be assured if the joining router becomes its descendant. For the end-to-end delay QoS metric, the on-tree router compares the requested delay, RESV.*req_delay*, with the sum of *s2c_delay* and *c2e_delay* fields of its corresponding routing entry *r*. That is, if *r.s2c_delay* + *r.c2e_delay* + RESV.*acc_delay* + delay (*l*) ≤ RESV.*req_delay*, where *l* is the link from which the RESV message has arrived, the on-tree router can accept the joining router as its descendant. The joining router is rejected if no on-tree router of any tree of the group is able to accept the joining router as its descendant.

The mechanisms to handle the group dynamics outlined in this section do not cause any disruptions for the services provided to the existing group members. This is since neither creating a new multicast tree nor adjusting existing shared trees of the group is considered. However, a number of new group participants may be rejected. Some improvements along the ideas proposed in previous work can be considered if higher overhead and/or service disruptions are allowed. For instance, to accommodate new joining routers as member routers, existing multicast trees may need to be adjusted or a new multicast tree may need to be constructed. To accommodate new sender routers without violating the QoS requirement of any existing group members, a number of source-specific trees can be used. If none of these can reasonably increase the number of new participants that can join the group, most of the steps of our core selection protocol may have to be repeated with a new set of sender routers and/or a new set of member routers. A multicast tree rooted at each newly selected core is, then, constructed using our multicast tree construction protocol. Due to the complexity of this problem, the detailed investigation to identify the best approach is beyond the scope of this paper.

[††] The on-tree router includes the core that is the destination of the RESV message as well.

4. Performance Evaluation

In this section, we extensively evaluate the performance of our core selection and multicast tree construction protocols via simulations. We compare the performance of our core selection technique with the optimal solution and QCSA discussed in Section 2. Finally, we investigate overhead of our multicast tree construction protocol. Table 3 summarizes the performance metrics used in this study.

4.1. Simulation Model

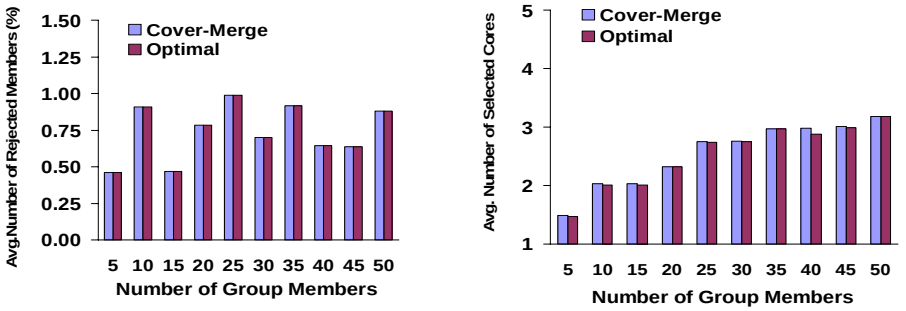
The QoS requirements used in our study are end-to-end delays. We constructed fifty networks based on the transit-stub model using GT-ITM [7, 6]. The stub domains and stub nodes represent regional multicast capable networks. Each network has a total of 300 nodes. GT-ITM assigns a distance to each link. We map the link distance to a link delay such that the longest path in the network has the total delay of approximately 70 ms, the delay we observed from separate experiments with the Internet. Queuing delays at the routers were not simulated since we are only interested in relative performance comparisons.

We conducted simulations under various group sizes ranging from 5 to 50 members. These group sizes cover many multi-sender multimedia applications. For each simulation run, the member routers were randomly selected only from the routers in the stub domains since the backbone routers are not directly connected to any user. All the member routers joined a single multicast group. In most simulations, each member router was also a sender router unless stated otherwise. Service classes in Table 1 were used in the simulations. We assigned only one service class to each member router. This restriction is

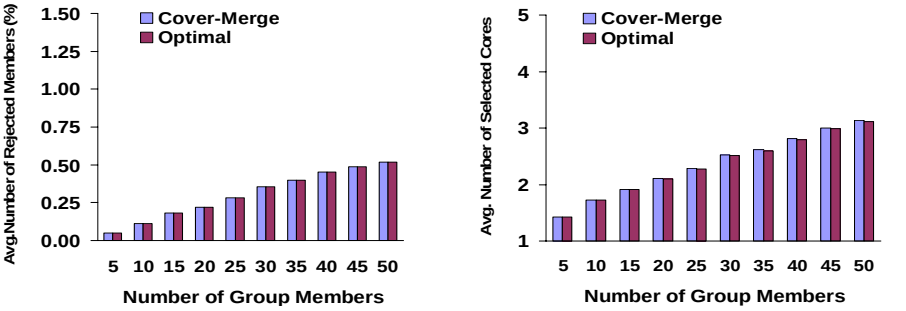
required to formulate the QoS core selection problem as an integer programming problem in which the optimal solution exists. In addition, each member router or sender router had only one group member or sender attached to it, respectively. In other words, the number of member routers and group members in our simulations were the same. This allows for clear observations on the effect of increasing a group size or the number of senders. Similarly, the number of sender routers and senders were the same. We generated the number of member routers subscribing to each service class based on the following assumption unless stated otherwise. That is, many more group members subscribe to the classes with a reasonable service fee (*short_delay* and *medium_delay*) compared to the most expensive class (*very_short_delay*) or the least expensive class (*best_effort*) that does not provide any guarantee. The member routers were randomly assigned to each service class. For QCSA, these member routers requested for the bounds indicated by the service classes.

Table 3: Performance Metrics

For evaluation of core selection algorithms, we use percentage of group members and number of selected cores as the metrics.
Percentage of rejected group members: The rejected group members include both insatiable members and satiable members whose QoS cannot be satisfied by some core selection algorithm. The lower the percentage, the better the protocol. **Number of selected cores**: A small number indicates low sender-to-core traffic.
For evaluation of the proposed tree construction algorithm, we use degree of overlapping, routing overhead, and stress as the metrics.
Degree of overlapping: Numbers of on-tree routers with multiple multicast routing entries. A small overlapping degree indicates that small portions of multicast trees of a group are overlapped. **Routing overhead**: Average of the numbers of routing entries per group at an on-tree router. **Stress**: Average of the numbers of duplicate packets on every tree branch in all multicast trees of a multicast group. The stress close to one is desirable since it indicates low packet duplication.



(a) Results when different group members may select different classes



(b) Results when group members selected only the *short_delay* class

Fig. 2. Performance of the proposed algorithm (Cover-Merge) and the optimal algorithm.

4.2. Simulation Results

We use the terms *Optimal*, *Cover-Merge*, and *QCSA* to refer to the optimal algorithm, our core selection algorithm, and QCSA, respectively. Each point in each of the subsequent plots is the average of the results of 5000 simulation runs.

4.2.1. Effectiveness of the Algorithms

Fig. 2 shows the performance comparison between Optimal and Cover-Merge. The optimal algorithm invokes the linear programming solver [3] to solve the QoS core selection problem formulated as an integer programming problem. The plots in the left column depict the average

percentage of rejected group members (member routers) whereas the corresponding plots on the right indicate the number of cores used under the same simulation parameters.

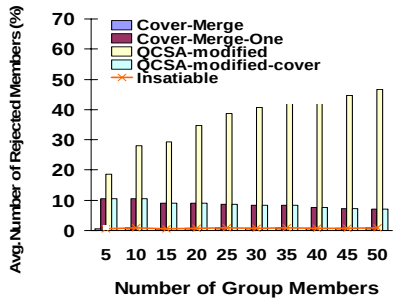
Fig. 2(a) shows the results when different group members may select different classes. This is the default condition we used in most of our simulations unless stated otherwise. In this scenario, Cover-Merge uses slightly more cores than Optimal on average. This demonstrates an effectiveness of our greedy algorithm. From the figure, Cover-Merge rejects the same number of group members as Optimal does. All of the rejected members are insatiable members which may come from any service class.

Fig. 2(b) contains the results when group members selected only the *short_delay* class. This is to simulate the situation when every group member subscribes to only one service class. In this scenario, Cover-Merge also chooses slightly more cores than Optimal on average. All of the rejected members in this case are also insatiable members. They are from *short_delay* class only.

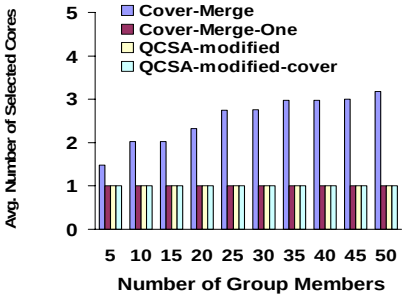
Notice that in the second scenario, the number of rejected members increases as the group size increases. On the other hand, in the first scenario, the number of rejected members varies as the group size increases. This indicates the influences of insatiable members from the *very_short_delay* and *medium_delay* classes and requires further investigations.

4.2.2. Performance Comparison with QCSA

We evaluate Cover-Merge and QCSA with respect to the percentage of rejected group members and the number of selected cores. Due to some differences between Cover-Merge and QCSA, a number of modifications were applied in order to fairly compare the two techniques. We modified our Cover-Merge to select only one core that covers the largest number of group members and call this modification *Cover-Merge-One*. Since the original QCSA only guarantees core-to-end delays, we modified the original QCSA to guarantee end-to-end delays. This modified technique is named *QCSA-modified*. We also improved QCSA-modified further to handle the case that no candidate core can construct QoS-assured paths to all other candidates. In this case, the candidate with the largest covering set is chosen as the core. We call this *QCSA-modified-cover* since it is influenced by our cover heuristic.



(a) Number of rejected members



(b) Number of selected cores

Fig. 3. Effect of routing with a core set.

Fig. 3(a) shows that Cover-Merge-One rejects 77.09% less group members than QCSA-modified and performs as well as QCSA-modified-cover. QCSA-modified has a very large percentage of rejected members because it ignores the case when not every candidate core can construct QoS-assured paths to all other candidates. The core selection fails and all the group members are rejected in that case. The similar performance between QCSA-modified-cover and Cover-Merge-One is expected since both schemes use the same concept in selecting one core. The rejected group members in those two cases are both insatiable members and the group members whose requirements cannot be satisfied by the single selected core. Fig. 3(a) also indicates that Cover-Merge reduces the percentage of rejected members by about 97.59% compared to QCSA-modified. Only the insatiable members cannot be satisfied by Cover-Merge. The plot labeled as *insatiable* in this figure indicates the average number of insatiable members in all the simulations. Fig. 3(b) demonstrates that Cover-Merge uses more cores when the group size increases. Although Cover-Merge uses about 3 cores when there are 50 group members, it can satisfy about 48% more group members compared to QCSA-modified.

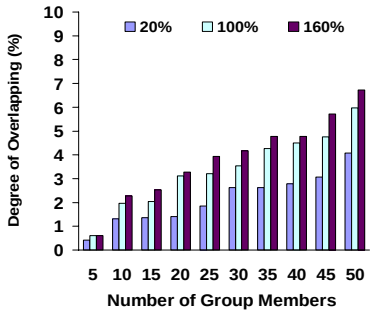
4.2.4. Performance of the Tree Construction Protocol

To evaluate the performance of our tree construction protocol, we measure the degree of overlapping, routing overhead, and stress of multicast trees constructed using the protocol. In the simulations, the number of senders was ranged from 20% to 160% of the group sizes. The results are shown in Fig. 4.

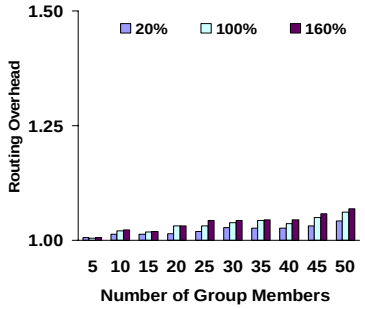
Fig. 4(a) shows that for any number of senders, multicast trees of a group are not overlapped much. In the most extreme scenario when the group size is 50 and the number of senders is 160%, the degree of overlapping is less than 7% on average. This indicates that not many of the on-tree routers are required to maintain multiple routing entries per group. The figure also demonstrates that for any number of senders, multicast trees of a group are overlapped more as the group size increases. This is because there are more group members to satisfy. Fig. 4(b) also show that for any number senders, multicast trees of the group do not cause much routing overhead to the on-tree routers. From the figures, the number or routing entries per group at an on-tree router in the most extreme case is about 1.07. The protocol incurs on average 1.05 duplicate packets forwarded on a tree branch (i.e., Stress). We omit the results here due to limited space.

5. Concluding Remarks

This paper presents new core-based routing with QoS support for multisender multimedia applications. Our contributions are the distributed core selection protocol and the multicast tree construction protocol. The core selection protocol employs the core selection algorithm that utilizes as many cores as necessary to maximize the number of group members with satisfied QoS requirements. Our simulations confirm that about 70% more members can be served with the desired service quality compared with a recent QoS core selection algorithm using a single core. The proposed core selection protocol is distributed, which helps to alleviate the hot spot and the single point of failure problems. Our multicast tree construction protocol utilizes path information gathered during the core selection protocol to construct a multicast tree rooted at each core of the group. The tree construction protocol ensures loop-free routing. The simulations demonstrate that the protocol does not incur much overhead on the network links and routers. Our ongoing research focuses on extending the new core-based routing for application-level multicast.



(a) Degree of overlapping



(b) Routing Overhead

Fig. 4. Performance of the multicast tree construction protocol.

Acknowledgments

This work is partially supported by the U.S. National Science Foundation grant No. 0092914. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Tree (CBT): An architecture for Scalable Inter-Domain Multicast Routing", Proc. of ACM SIGCOMM, Ithaca, NY, USA, pp. 85-95, Sept. 1993.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast", Proc. of ACM SIGCOMM, Pittsburgh, PA, USA, pp. 205-217, Aug. 2002.

[3] M. Berkelaar, "LP_SOLVE: Linear Pgmming Code", 1996.
S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Service", RFC 2475, IETF http://www.faqs.org/rfcs/rfc2475.html.

[4] L. Blazevic and J. L. Boudec, "Distributed Core Multicast: a multicast routing protocol for many groups with few receivers", ACM SIGCOMM CCR, vol.29, no.5, pp. 6-21, Oct. 1999.

[5] K. Calvert, M. Doar, and E. Zegura, "Modeling Internet Topology", IEEE Communications magazine, vol.35, no.6, pp.160-163, Jun. 1997.

[6] K. Calvert and E. Zegura, "Georgia Tech Internetwork Topology Models", 1996.

[7] K. Calvert, E. Zegura, and M. Donahoo, "Core Selection Methods for Multicast Routing", Proc. of ICCCN, Las Vegas, NV, USA, pp. 638-642, Sept. 1995.
[8] K. Carlberg and J. Crowcroft, "Building shared trees using a one-to-many joining mechanism", ACM SIGCOMM CCR, vol.27, no.1, pp. 5-11, Jan. 1997.
[9] S. Chen, "Routing Support for Providing Guaranteed End-to-End Quality-of-Service", Ph.D. thesis, UIUC.
[10] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions", IEEE Network magazine, Special Issue on Transmission and Distribution of Digital Video, pp. 64-79, Nov./Dec. 1998.
[11] S. Chen, K. Nahrstedt, and Y. Shavitt, "A QoS-Aware Multicast Routing Protocol", IEEE JSAC, vol.18, no.12, pp. 2490-2498, Dec. 2000.
[12] Y. Chu, S. G. Rao, and H. Zhang, "A case of end system multicast", Proc. of ACM SIGMETRICS, Santa Clara, CA, USA, pp. 1-12, Jun. 2000.
[13] S. Chung, and C. Youn, "Core Selection Algorithm for Multicast Routing under Multiple QoS Constraints", Electronic Letters, vol.36, no.4, pp. 378-379, Feb. 2000.
[14] T. Cormen, C. Leiserson, and R. L. Rivest, "Introduction to Algorithms", McGraw Hill, 2000.
[15] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "An Architecture for Wide-Area Multicast Routing", Proc. of ACM SIGCOMM, London, UK, pp. 125-135., Aug./Sept. 1994.
[16] S. Deering, B. Fenner, D. Estrin, A. Helmy, D. Farinacci, L. Wei, M. Handley, V. Jacobson, and D. Thaler, "Hierarchical PIM-SM Architecture for Inter-Domain Multicast Routing", Internet Draft.
[17] M. Donahoo, and E. Zegura, "Core Migration for Dynamic Multicast Routing", Proc. of ICCCN, Washington, DC., USA, Oct. 1996.
[18] M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of service sensitive multicast internet protocol", Proc. of ACM SIGCOMM, Vancouver, Canada, pp. 144-153, Sept. 1998.
[19] E. Fleury, and Y. Huang, "On the performance and Feasibility of Multicast Core Selection Heuristics", Proc. of ICCCN, Lafayette, LA, USA, pp. 296-303, Oct. 1998.
[20] P. Francis, "Yoid: Your Own Internet Distribution", http://www.icir.org/yoid/.
[21] Y.-H. Chu and H. Zhang, "Considering Altruism in Peer-to-Peer Internet Streaming Broadcast", Proc. of ACM NOSSDAV, Cork, Ireland, pp. 10-15, Jun. 2004.
[22] Y. Huang, E. Fleury, and P. McKinley, "LCM: A Multicast Core Management Protocol for Link-State Routing Networks", Proc. of IEEE ICC, Atlanta, Georgia, USA, pp. 1197-1201, Jun. 1998.
[23] I. Mas, V. Fodor, and G. Karlsson, "Probe-Based Admission Control for Multicast", Proc. of IWQoS, Miami Beach, FL, USA, pp. 97-109, May 2002.
[24] W. Putthividhya, W. Tavanapong, M. Tran, and J. Wong, "A Novel Core Selection with End-to-End QoS Support for Multi-sender Multimedia Multicast Applications", TR-03-07, Dept. of Comp Science, Iowa State University, 2003.
[25] W. Putthividhya, W. Tavanapong, M. Tran, and J. Wong, "Core Selection with End-to-End QoS Support", Proc. of ACM SAC, Nicosia, Cyprus, pp. 328-333, Mar. 2004.
[26] W. Putthividhya, W. Tavanapong, M. Tran, and J. Wong, "Distributed Core Selection with QoS Support", Proc. of IEEE ICC, Paris, France, pp. 2132-2137, May 2004.
[27] D. Thaler, and C. Ravishankar, "Distributed Center-Location Algorithms", IEEE JSAC, vol.15, no.3, pp. 273-276, April 1997.
[28] H.-Y. Tyan, C.-J. Hou, and B. Wang, "A Framework for Provisioning of Temporal QoS in Core-based Multicast Routing", Proc. of IEEE Real-Time Systems Symp., Phoenix, AZ, USA, pp. 168-178, Dec. 1999.
[29] VRAC, "Virtual Reality Applications Center", Iowa State University, http://www.vrac.iastate.edu.
[30] D. Zappala, A. Fabbri, and V. Lo, "An Evaluation of Shared Multicast Trees with Multiple Cores", Journal of Telecommunication Systems Kluwer Academic Publishers, vol.19, no.3-4, Mar. 2002.
[31] R. Zimmermann, and L. Liu, "ACTIVE: Adaptive low-latency peer to peer streaming", Proc. of ACM MMCN, San Jose, CA, USA, Jan. 2005.

Wanida Putthividhya received the B.S. degree in Computer Science (first class honor) from Thammasat University, Thailand in 1996. She received the M.S. degree in Computer Science from University of Southern California in 2000. She is currently a Ph.D. candidate at the department of Computer Science, Iowa State University.

Wallapak Tavanapong is an Associate Professor of Computer Science at Iowa State University. She received her Ph.D. degree in Computer Science from the University of central Florida in 1999. Her current research interests include network support for quality of service, medical video analysis, and multimedia databases. She is a recipient of the U.S. National Science Foundation CAREER award. She has served as an editorial board member for ACM SIGMOD Digital Symposium Collection, a program committee member for several international conferences, and a referee for several conferences and reputable journals.

Johnny Wong is a Professor & Associate Chair of the Computer Science Department, Iowa State University. He received his Ph.D. in Computer Science from University of Sydney, Australia. His research interests include Operating Systems, Computer Networks, Multimedia Systems, and Peer-to-Peer Systems. He has published over 100 papers in peer reviewed journals and conferences.