A Scalable QoS Routing Architecture for Real-Time CSCW Applications*

IBRAHIM MATTA
College of Computer Science
Northeastern University
Boston, MA 02115
matta@ccs.neu.edu

MOHAMED ELTOWEISSY
Department of Computer Science
Central Connecticut State University
New Britain, CT 06050
eltoweissym@ccsu.edu

Abstract

Computer Supported Cooperative Work (CSCW) has the potential of providing the environment needed for groups of diverse users to cooperate in real-time to achieve their common goals. This potential has been under-utilized due to the lack of cooperation between CSCW systems and the underlying network services to accommodate the dynamic behavior and varying Quality-of-Service (QoS) requirements of CSCW groups and the applications they use. To satisfy such requirements, QoS multicast routing should be employed. Unfortunately, current multicast services are inadequate for the traffic generated in CSCW environments. In this paper, we present a CSCW-specific routing architecture that supports QoS requirements and is both scalable and robust. Scalability is needed to accommodate large number of multicast groups and highly dynamic group membership and behavior, while robustness provides reliability and stable performance for each group during adaptation and in presence of the other existing groups.

In our architecture, a router QoS manager, on behalf of a multicast routing manager, negotiates with host and CSCW-specific QoS managers for efficient resource utilization and guaranteed QoS delivery. The multicast routing manager switches between routing trees and algorithms as warranted by the changes in the characteristics and requirements of the CSCW systems running at the hosts. We present a class-based and a partial view-based method to scalability. We also present a centralized and a distributed method to establishing a group's QoS multicast path while providing robustness. In addition to prototyping our architecture and defining appropriate protocols and mappings between the various QoS and router managers, we are investigating the use of the Internet standard SNMP for information gathering. This necessitates extending its management information bases (MIBs) to include group objects.

*This work was supported in part by NSF CAREER grant NCR-9701988 and NEU grant RSDF-377090, and by a grant from CCSU Academic Computing.

1 Introduction

The potential of Computer Supported Cooperative Work (CSCW) systems to provide the support needed for groups of diverse users to work together in real-time has been under-utilized. One major obstacle is the lack of cooperation between CSCW systems and the underlying network services to accommodate the dynamic behavior and varying Quality-of-Service (QoS) requirements of CSCW groups, the applications they employ and the data they manipulate and exchange [7]. CSCW research has traditionally focused on the design of shared environment applications assuming that the underlying network will provide the needed communication services. On the other hand, network researchers have focused on the development of general services that are not tailored to the needs of individual applications. The lack of a common framework that integrates CSCW and network research impedes the wide spread use of computer-supported collaboration. Our goal is to develop and implement a real-time CSCW-specific multicast routing architecture with the following features: (1) **QoS support:** to effectively support the various service levels as needed by the different CSCW systems; (2) **scalability:** to accommodate large number of multicast groups and highly dynamic group membership and behavior; (3) **robustness:** to provide stable performance for each group in presence of the other existing groups.

To provide *QoS support*, we construct different multicast routing trees (possibly using different algorithms) to effectively support the various service requirements of the different CSCW systems. There has been many algorithms proposed to construct multicast paths (e.g. [9, 10, 6, 12, 4]). Some algorithms build a single (shared) Steiner minimal tree that spans all group members in order to minimize replication and bandwidth cost. Others build source-rooted trees (one for each sender in the group) in order to minimize path length or end-to-end delay. While still others try heuristically to achieve a balance between cost and delay. However, none of the above studies address the need for CSCW-specific multicast routing. Only very recently, a few studies [1, 2, 5] have tackled this problem. In [1, 2], Aggarwal *et al*. present an architecture for multi-party conferencing. Their work, however, does not address robustness as it relates to the performance interdependencies among various groups, nor scalability as it relates to the processing and maintenance of the large amount of QoS information. In [5], Huang *et al*. present a parameterized distributed multicast routing protocol, where a control parameter allows the construction of a range of trees from source-based to shared. Their work does not however present an end-to-end architecture nor does it discuss the varying requirements of CSCW groups and their dynamic behavior.

In addition to supporting the use of various algorithms for finding appropriate QoS multicast trees, our architecture also supports the use of various methods to achieving *scalability*. Our approach is based on: (1) the separation of the architecture and mechanisms for information gathering from those for route computation, and (2) reducing the amount of QoS information maintained at the routers. In one method, we classify CSCW systems into a few number of classes according to their characteristics and requirements and use class-based multicast routing instead of individual group-based approaches. In another method, we reduce the amount of QoS information maintained by reducing the network map (view) maintained by each router. The two methods or a hybrid of them can be deployed simultaneously. Finally, *robustness* is achieved in our architecture by constructing a QoS multicast path for a group without violating the QoS guarantees promised to existing groups. This avoids oscillations that would arise because of the performance interdependencies among the various groups. We next present our end-to-end QoS routing architecture. We conclude with future research and implementation issues.

2 Proposed Architecture

Our design decisions are motivated by the desire to evolve an architecture that is scalable and flexible, offers high performance, and supports real-time multimedia and diverse group requirements. Figure 1 shows the architecture. We now discuss its main features.

2.1 QoS Support

A CSCW application communicates its characteristics and requirements, such as its communication topology (peer-to-group, group-to-peer, etc.), delay, etc., to the CSCW QoS manager via a QoS Application Programming Interface (API). This CSCW QoS manager understands application-specific attributes, and maps them to application-independent attributes. These attributes are passed to the host QoS manager, which in turn maps them to lower-level attributes. The host QoS manager communicates with other components to make sure that the application's requirements are satisfied and that the network is aware of changes to those requirements. Peer QoS managers at the source and destination hosts may need to cooperate requesting the allocation of necessary resources from their operating systems and network subsystems. The host QoS manager communicates the requirements of the various applications to the router QoS manager, which is responsible for allocating necessary resources within the network by interacting with its peers.

It is the job of the multicast routing managers at the routers to construct appropriate multicast trees to accommodate requests from router QoS managers. A request coming from a router QoS manager can be the result of a user requesting to join an ongoing multicast group or a change to an application's QoS. The following example illustrates the need for different tree types under different QoS requirements or network conditions. Consider a CSCW system with end-to-end delay requirement of 13 and delay jitter of 7 running on the simple 3-node network shown in Figure 2(a).[1] Two senders $S1$ and $S2$ are located at nodes A and C respectively, and three receivers one at each node. Assume the link costs indicated in Figures 2(b) and 2(c) reflect the maximum link delays in the presence of this CSCW group. With a shared tree (Figure 2(b)), maximum end-to-end delay is 20 and delay jitter is 10, which do not satisfy the system's delay requirements. With source-rooted trees (Figure 2(c)), however, maximum end-to-end delay is 10 and delay jitter is zero. In this

[1] Delay jitter is the difference between the minimum and maximum source to destination end-to-end delay.

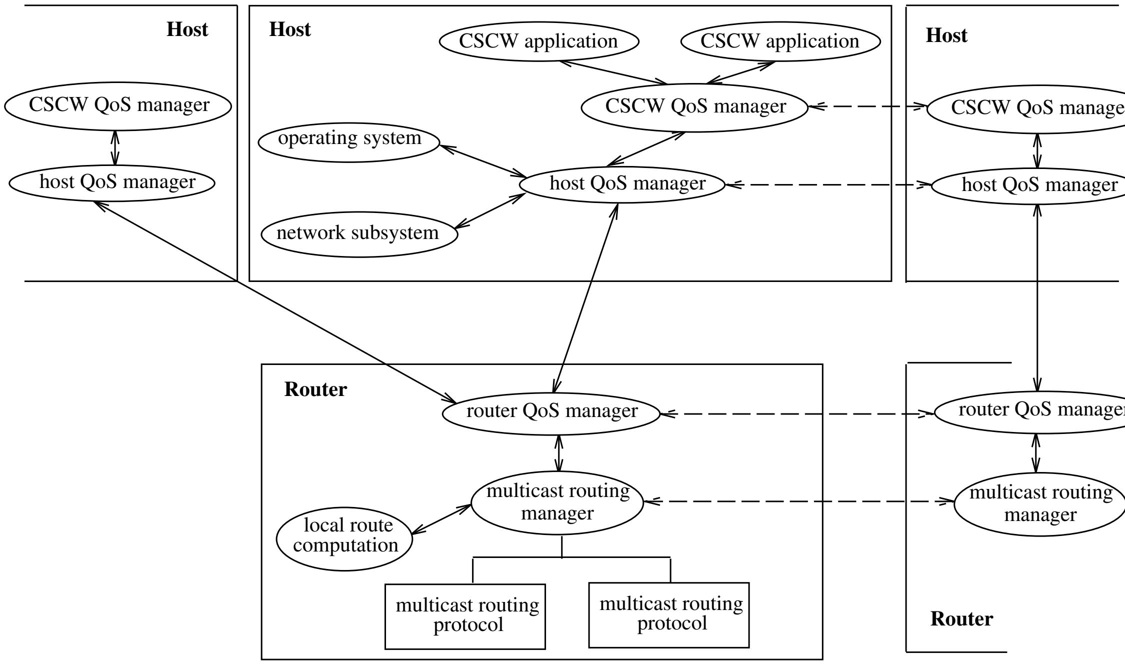

Figure 1. Integrated QoS architecture.

case, the routing algorithm should construct source-rooted trees.

Consider the case where the current network load results in the maximum link delays shown in Figures 3(b) and 3(c). In this case, with source-rooted trees (Figure 3(b)), maximum end-to-end delay is 12 and delay jitter is 10. This violates the system's jitter requirement. Then the routing algorithm should construct the shared tree shown in Figure 3(c), where maximum end-to-end delay is 6 and jitter is 3 (both are within the system's delay requirements). In any case, if the network cannot find a suitable tree that satisfies the requested QoS, then the network informs the application, which may choose to adapt to the (lower) QoS that the network can deliver.

Switching between multicast trees/algorithms

Our architecture supports multiple (parameterized) multicast routing protocols. This allows a group's router agent, which is responsible for processing all requests for join or QoS change made by members of its group, to flexibly initiate the switching to a different multicast tree(s)/path to satisfy those requests. We consider two approaches to this switching:

- **Centralized:** similar to the approach taken in [1]. Here the group's router agent sends routing information to all routers on the new multicast path so as to update their routing entries for this group. Once the new multicast path is successfully established, the old multicast path is deleted.

- **Distributed:** where a group's router agent only initiates the execution of the distributed multicast protocol. This has the advantage of building a multicast path using the *current* (instantaneous) state of the network, as opposed to the delayed (usually outdated) state maintained by the router agent as in the centralized approach. Switching to a *different* multicast routing protocol requires an additional mechanism to secure its initiation at *all* routers. Our approach is to use a "broadcast-echo" mechanism that ensures that messages sent by the new routing daemon will be received by its peer daemon running on a neighbor router. This mechanism works as follows: a group's router agent initiates the switch by broadcasting a "switch" message to all neighbor routers, and enters a "switching" state. This process is repeated as the switch message travels. Each router receiving a switch message broadcasts it to all its neighbor routers except the sender to which it sends back an "echo" message. The router then enters the switching state. In a switching state, a router accepts and buffers any new routing messages (corresponding to the new multicast routing algorithm).

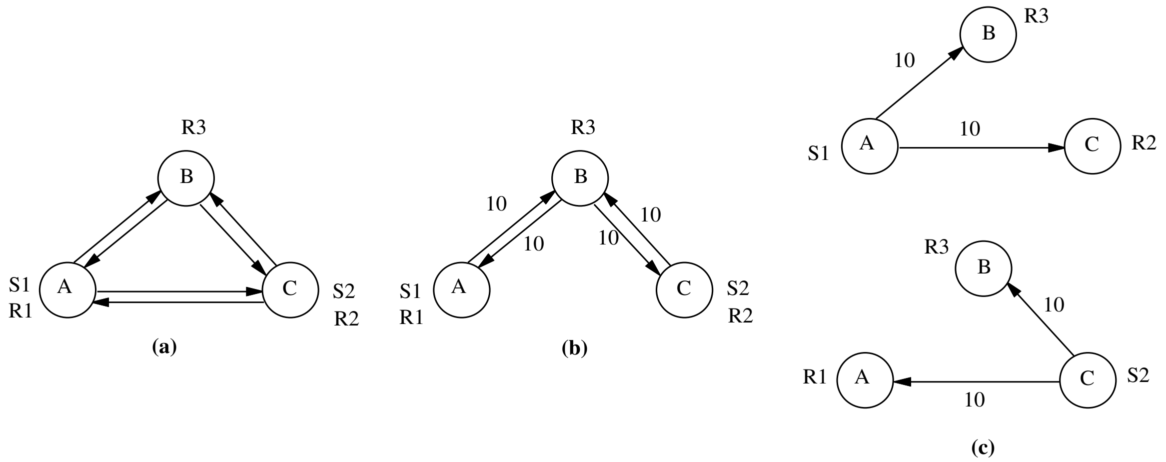

Figure 2. Example network.

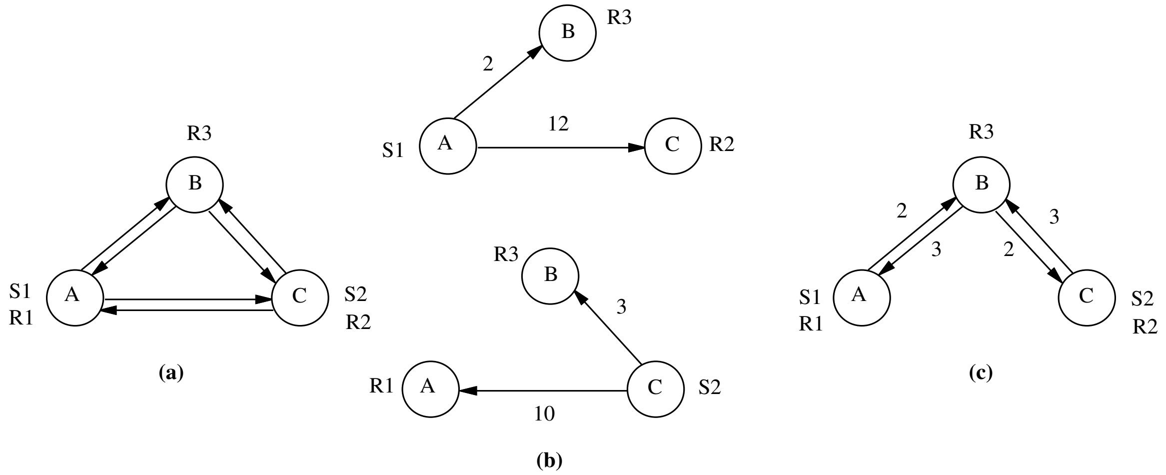

Figure 3. Different maximum link delays.

Once a router receives an echo message from each of its downstream neighbor routers, it leaves the switching state and starts processing the new routing messages that are buffered. Note that receiving an echo message from each downstream neighbor means that all neighbors have been notified of the switch and are now running the new routing daemon.

2.2 Scalability

In our architecture, we separate the architecture and mechanisms for information gathering from those for route computation. We consider two approaches to information gathering:

- **Class-based:** where groups are classified into a small number of classes according to their characteristics and requirements. The QoS parameters (e.g. maximum end-to-end delay, maximum distance between a sender and a receiver, etc.) of each class are pre-specified. A group's router agent keeps the whole view of the network with class-based statistics associated with links and nodes in the view, for example, number of groups per class on each link. Scalability is achieved by maintaining class statistics rather than individual group statistics at each router, which greatly reduces the amount of QoS information and simplifies the computation of QoS multicast trees.
- **Partial view-based:** maintains individual group information, but a router agent maintains only a *partial* view of the network (as opposed to a full view as in the first approach). We use a view-server (VS) hierarchical approach similar to [3].

In this approach, special routers act as VSs where each VS maintains a (partial) view of a small area around itself (called VS's precinct). By querying a sequence of VSs, a merged (partial) view containing the group members can be obtained by the group's router agent, which uses it to compute QoS multicast trees.

To make these queries efficient, VSs are organized in a hierarchy such that the address of a node specifies a sequence of VSs whose merged view contains a path from a top-level VS to the node. Top-level VSs are placed such that they can reach each other (*i.e.* the view of a top-level VS contains paths to every other top-level VS). Given this VS hierarchy and VS addresses of group members, the group's router agent can query all VSs in these addresses to obtain a merged (partial) view containing paths among all group members. Actually, the query process can be stopped as soon as the merged view contains all group members. A group's router agent can periodically or on demand obtain this partial view and use it to compute an appropriate multicast tree(s) for its group.

2.3 Robustness

As in [2], we assume that a particular router for each group is responsible for handling requests for join or QoS change. We call this router the group's *router agent*. The functions of this router agent can be replicated for reliability. The router agent uses its current view of the network to choose an appropriate multicast tree(s)/path that would satisfy the group's QoS requirement *while* not violating the QoS requirements of other existing groups. If such a path cannot be found, then the request is rejected and the application is informed (this is an admission control function performed at the group's router agent for nodes along the selected path). A scalable protocol—which is based on the class-based method, the partial view-based method or a hybrid of them—is deployed to collect at every router agent the view and QoS information associated with links and nodes in the view. Nodes include routers as well as hosts so that each router agent can estimate *end-to-end* QoS measures.

After a group's router agent chooses a multicast path that is likely to satisfy its group QoS, the router agent then initiates the construction of the path together with the reservation of necessary resources (buffers, bandwidth, service priority, etc.) at nodes along the chosen path. Resources could be reserved to provide either hard or soft real-time guarantees. If the reservation does not succeed at any of the nodes, the requesting application could be asked to adapt to the reservation level that could be granted.[2] Our architecture also allows the processing of requests to be done locally by allowing an application to request the computation of a local path or alternate path. In this case, the local multicast routing manager invokes a route computation algorithm that locally updates the multicast tree as in [13]. Again, the local multicast routing manager can ask the application to adapt if only a lower QoS can be granted.

An important feature of our architecture is stability. Instabilities could result if join and QoS requests are honored without accounting for the QoS requirements of already existing groups. In such case, oscillations can occur because a request from one group possibly resulting in a tree update may affect the QoS delivered to another existing group, which in turn may request the network to adapt its tree. In our architecture, such instabilities are avoided since existing QoS guarantees are not violated while honoring a new request for join or QoS change. To illustrate, consider the shared tree example in Figure 3(c). Denote its associated group by $G1$. Consider a new group $G2$ with sender $S1$ at node A and receivers $R1$ and $R2$ at nodes B and C, respectively. Assume G2's maximum delay requirement is 20 and jitter is 8. See Figure 4(a). Figure 4(b) shows the maximum link delays if $G2$ is admitted on the same shared tree as $G1$ where the maximum link delays of links (A, B) and (B, C) increase from 2 to 8. But admitting $G2$ on this shared tree would clearly violate $G1$'s requirements (recall, they were maximum delay of 13 and jitter of 7). Hence $G2$'s request should be either denied or admitted on the source tree shown in Figure 4(c) with maximum delays of 8 and 18 on links (A, B) and (A, C), respectively. In this latter case, the maximum delay for $G2$ is 18 and jitter is 10, and so $G2$ is asked to tolerate some extra jitter.

3 Future Work

We presented a general integrated QoS architecture to effectively support real-time collaboration. The implementation of this architecture will allow us to investigate and compare the various approaches to QoS support, robustness and scalability that we have presented. For example, we will investigate class-based information gathering by extending information gathering tools to provide per class parameters for multicast groups. This includes the use of the existing Simple

[2] Alternatively, to make the reservation for the requesting group successful, the QoS delivered to another existing group(s) could be degraded as long as it remains within the acceptable QoS region of that group(s).

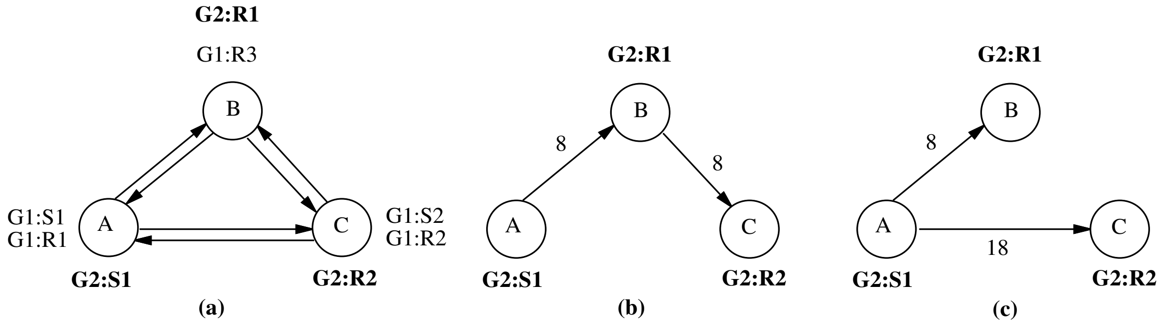

Figure 4. Performance interdependencies among groups.

Network Management Protocol (SNMP) and the extension of management information bases, in particular SNMP MIB-II [8] and the remote monitoring (RMON) MIBs [11] to include group objects. We will also define criteria for group classification based on CSCW characteristics and requirements, and develop policies and mechanisms to resolve class conflicts. Other issues include the mapping of QoS measures at different levels of the architecture, and the selection of QoS multicast trees in the presence of feedback delays.

References

[1] S. Aggarwal and S. Paul. A Flexible Protocol Architecture for Multi-Party Conferencing. In Proc. *IEEE ICCCN*, pages 81–91, 1996.

[2] S. Aggarwal, S. Paul, D. Massey, and D. Caldararu. A Flexible Protocol Architecture for Multi-Party Conferencing: From Design to Implementation, 1997. Draft manuscript.

[3] C. Alaettinoglu, I. Matta, and A.U. Shankar. A Scalable Virtual Circuit Routing Scheme for ATM Networks. In Proc. *International Conference on Computer Communications and Networks - ICCCN '95*, pages 630–637, Las Vegas, Nevada, September 1995.

[4] A. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees. In Proc. *SIGCOMM '93*, San Francisco, California, September 1993.

[5] N-F. Huang, C-Y. Yeh, and C-C. Chiou. PDMRP: A Programmable Distributed Multicast Routing Protocol, 1997. Draft manuscript.

[6] L. Kou, G. Markowsky, and L. Berman. A Fast Algorithm for Steiner Trees. *Acta Informatica*, 15:141–145, 1981.

[7] I. Matta, M. Eltoweissy, and K. Lieberherr. From CSCW Applications to Multicast Routing: An Integrated QoS Architecture. NU-CCS-97-09, Northeastern University, College of Computer Science, May 1997.

[8] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. Request for Comments RFC-1213, March 1991.

[9] J. Moy. Multicast Extensions to OSPF. Internet draft, Network Working Group, September 1992.

[10] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. Request for Comments RFC-1175, November 1988.

[11] S. Waldbusser. Remote Network Monitoring Management Information Base. Request for Comments RFC-1757, February 1995.

[12] B. Waxman. Routing of Multipoint Connections. *IEEE J. Select. Areas Commun.*, SAC-6(9):1617–1622, December 1988.

[13] D. Zappala, B. Braden, D. Estrin, and S. Shenker. Interdomain Multicast Routing Support for Integrated Services Networks. Internet Draft, March 1997.