From CSCW Applications to Multicast Routing: An Integrated QoS Architecture

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

Karl Lieberherr†
College of Computer Science
Northeastern University
Boston, MA 02115
lieber@ccs.neu.edu

Abstract

Computer Supported Cooperative Work (CSCW) has the potential of providing the environment needed for groups of diverse users to cooperate to achieve their common goals. This potential 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 of CSCW groups. Multicast routing presents a strong case where current network services are inadequate for the traffic generated in CSCW environments. In this paper, we demonstrate the need for CSCW-specific multicast routing and present an integrated QoS architecture where 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 or algorithms as warranted by the changes in the characteristics and requirements of the CSCW systems running at the hosts. We discuss an aspect-oriented programming approach to the efficient description and flexible implementation of adaptive application and network protocol behavior.

1 Introduction

In Computer Supported Cooperative Work (CSCW) environments, group members collaborate by sharing applications and data and exchanging messages to jointly accomplish their goals. Computer conferencing, concurrent engineering, and interactive distance learning are typical examples of CSCW environments [6, 4, 3]. Collaborative activities, by and large, are distributed in nature and thus rely heavily on the underlying network infrastructure.

In CSCW environments, traffic from one or more senders is targeted to a selected set of recipients with varying Quality-of-Service (QoS) delivery constraints. This requires the underlying network to provide multicasting and QoS capabilities for efficient CSCW support.

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 approach is to provide an integrated architecture where both CSCW applications and network control entities actively share information and mutually adapt to changes in the environment. In this paper, we focus on the important issue of multicast routing support for CSCW environments. Specifically, 1) we discuss the characteristics of CSCW environments that impact the underlying network, and identify their networking requirements; 2) we demonstrate the unsuitability of existing multicast routing services to CSCW traffic; 3) we present a flexible integrated architecture where the application and the network cooperate to achieve an efficient multicast routing support for CSCW traffic; and 4) we highlight several implementation issues showing the use of Aspect Oriented Programming (AOP) as a tool to conveniently and flexibly implement adaptive behaviors of applications and protocols.

The paper is organized as follows. In Sections 2 and 3, we discuss the characteristics and networking requirements of CSCW environments. In Sections 4 and 5, we review and assess current multicast routing algorithms. In Section 6, we present our integrated architecture. In Section 7, we discuss implementation

*Supported by NSF CAREER grant ANIR-9701988 and NEU grant RSDF-377090.

†Supported by DARPA and Rome Laboratory under agreement F30602-96-0239 and NSF under grant CCR-9402486 and a grant from Xerox PARC.

issues. We conclude in Section 8 with related and future work.

2 Characteristics of CSCW Systems

CSCW systems have several characteristics that, taken together, distinguish them from other systems requiring networking support. Most groups are dynamic in nature, they may toggle between interaction modes and employ more than one application simultaneously. The goal is that the dynamic behavior exhibited in CSCW environments should not be restricted by the underlying network, at the same time, CSCW environments must expend every effort to adapt to the changes in the underlying network infrastructure. To take into account and exploit the different characteristics of CSCW environments, we must recognize the interplay between these characteristics and the underlying network infrastructure.

The demands on the networking infrastructure varies with the variations in CSCW environments. For example, message systems which support asynchronous communication between groups of users, in general, generate less traffic and do not require the same responsiveness as conferencing systems that provide synchronous multimedia interactions between users. When the rate of join/leave is high or when participants form a heterogeneous group each with his/her own networking demands or when participants frequently change roles between being senders and recipients, the communication structures built by the network services might need to be modified, sometimes even replaced. Moreover, multi-media traffic calls for bandwidth guarantees and bounded end-to-end delays between senders and recipients. In addition, to avoid distracting users, bounding the delay jitter (variation)[1] is strongly desirable.

3 Requirements of CSCW Systems

As a result of the aforementioned characteristics, CSCW environments have their own networking support requirements, including bandwidth, responsiveness, reliability, scalability, and QoS guarantees. To satisfy these requirements, several performance and quality indices should be communicated between the CSCW system and the underlying network. These include: throughput, end-to-end delay, delay jitter, error rate, link/tree setting overhead (setup time, state storage requirements, etc.), and partition failures.

4 Multicast Routing

Current multicast routing protocols differ in the kind of traffic they can support and QoS they can provide. The routing of multicast traffic requires the construction of a tree or a group of trees, referred to as the *multicast path*. There has been many algorithms proposed to construct multicast paths (e.g. [12, 14, 9, 15, 2]). Some algorithms build a single (shared) Steiner minimal tree that spans all group members in order to minimize replication/bandwidth cost and, in general, delay jitter. Others build source-rooted trees (one for each sender in the group) in order to minimize path length or end-to-end delay and, in general, better distribute traffic over the network. While still others try heuristically to achieve a balance between cost and delay. Kompella et al. in [8] attempt to construct a shared tree subject to a given upper bound on end-to-end delay.

5 Need for CSCW-Specific Routing

To effectively support collaboration, we need multicast routing that *adapts* to the dynamics of the characteristics and requirements of CSCW environments. This adaptation could be triggered by a QoS manager signaling the routing subsystem on behalf of the CSCW system because of changes in semantics (i.e. group, application and data characteristics and requirements). Adaptation could also be initiated by the network itself in response to changing network load conditions. To illustrate the need for adaptation, 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 1(a). 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 1(b) and 1(c) reflect the maximum link delays in the presence of this CSCW group. With a shared tree (Figure 1(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 1(c)), however, maximum end-to-end delay is 10 and delay jitter is zero. In this 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 2(b) and 2(c). In this case, with source-rooted trees (Figure 2(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 2(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.

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

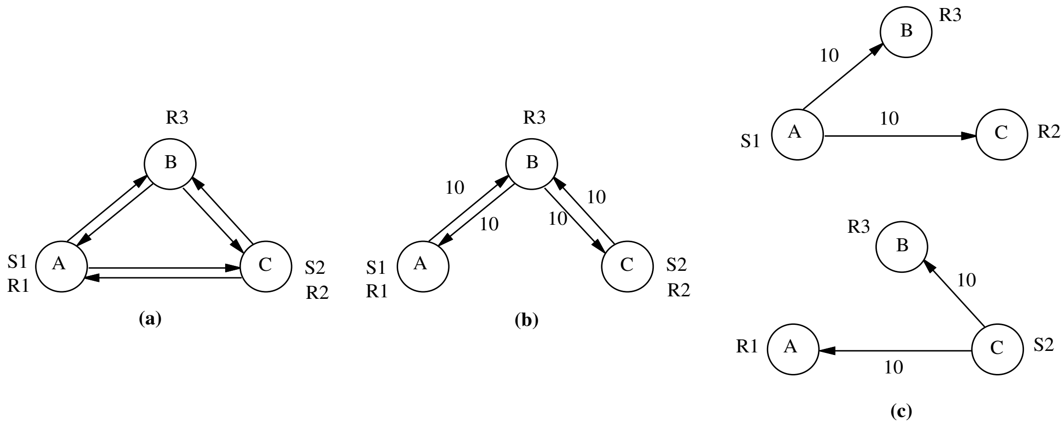

Figure 1. Example network.

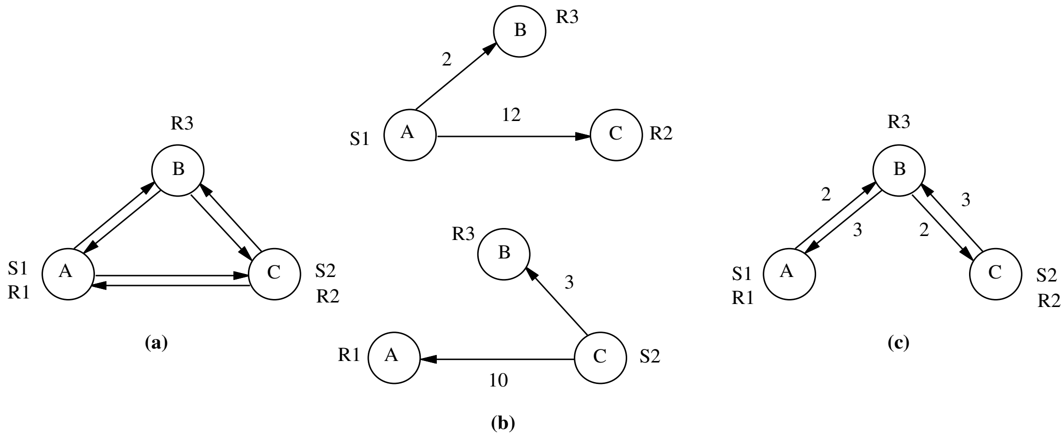

Figure 2. Different maximum link delays.

6 Integrated QoS Architecture

Figure 3 shows our architecture. 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 oversees and 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 adapt their behavior in response to network load or requests from router QoS managers. A request coming from a router QoS manager can be the result of an application entity requesting the computation of a local path or alternate path. In this case, the multicast routing manager invokes a route computation algorithm that locally updates the multicast tree as in [16].

6.1 Switching Multicast Tree/Algorithm

Our architecture supports multiple (parameterized) multicast routing protocols. This allows a group's router agent, which is responsible for processing all requests (e.g. for QoS change) made by members of its group, to flexibly initiate the switching to a different multicast tree(s)/path to satisfy those requests. 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 vi-

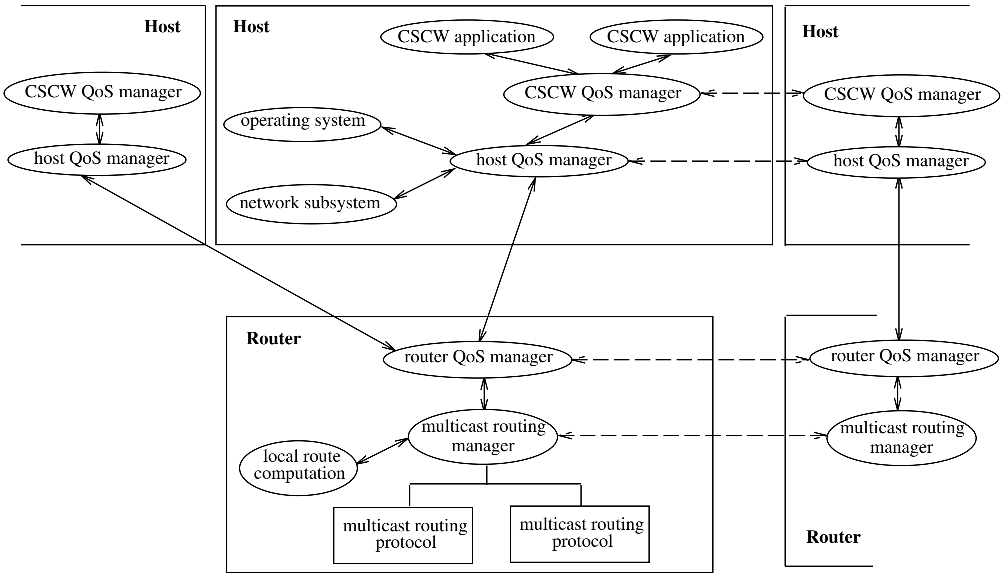

Figure 3. Integrated QoS Architecture.

olating the QoS requirements of other existing groups. 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. Switching to a *different* distributed multicast routing protocol requires an additional mechanism to secure its initiation at all routers [11].

7 Using Aspect Oriented Programming

CSCW environments often involve multimedia applications which impose QoS delivery constraints on the hosts and network. However, such applications can usually operate within an acceptable QoS region and adapt their behavior to accommodate the current QoS. The latter may be observed by the application itself or by the QoS manager and be communicated to the application. For example, an MPEG video application can adapt its compression strategy (e.g., ratio of I frames to P and B frames) based on currently delivered QoS. An application might also adapt because its own objectives change within the same session [7].

For applications to be able to express their QoS demands and adaptive behavior in an abstract, unambiguous and flexible way, Aspect Oriented Programming (AOP) [5] could be used. An AOP application consists of components expressing the core behavior (using, for example, C++ or Java or a procedural language) and several aspectual descriptions dealing with issues relevant to the application, for example, QoS, communication or synchronization. AOP has several roots, one of them is adaptive programming (AP) [10]. To illustrate, we show in Figure 4 a simplified QoS aspect language for a generic video application, which adapts its coding parameters based on currently delivered QoS.

In this example, to make the application QoS-sensitive, we have added a QosAspectDescription object into the Receiver object. The meaning is that the QosComputation object in the QosAspectDescription will update the current QosProperty object, which will then update the CodingBehavior of the application according to the information in the selection table. The selected CodingBehavior will then be used by the application to do the encoding. The join points between the aspect description and the application behavior are the coding behavior names.

This simple example illustrates the ease of maintenance when AOP is used to describe adaptive applications. To deal with QoS, we just have to add a QosAspectDescription object, implement the coding behaviors referenced in the selection table and use the selected CodingBehavior object in the application to do the appropriate coding in response to changing QoS. To change the coding selection mechanism, we only have to update some data: the selection table. AOP could similarly be used to obtain flexible implementation of adaptive network protocol behavior.

8 Conclusions

We presented a general integrated QoS architecture to effectively support CSCW systems. A few recent studies relate to our work. Aggrawal et al. in [1] present an architecture where a near-optimal shared tree is constructed and periodically recomputed to re-

```
// aspect classes
QosAspectDescription =
    QosComputation
    <current> QosProperty // initially undefined,
                          //updated by QosComputation
    SelectionTable
    CodingBehavior.

SelectionTable =
    <selections> List(Mapping).
Mapping = "if" QosProperty "in" Range "then" BehaviorName.
List(S) ~ {S}.
// end aspect classes

// application classes
Application =
    Display
    Camera.
Camera = Sender CodingBehavior.
Display = Receiver.
Sender = Frame Rate.
Frame = SendingTime Info.
Receiver = QosAspectDescription Sender.

CodingBehavior : Coding1 | Coding2 *common* BehaviorName.
Coding1 = .
Coding2 = .
```

Figure 4. A simplified QoS aspect language for a generic adaptive video application.

store its near optimality. The architecture does not however address the need for multiple multicast algorithms or issues related to major characteristics and QoS requirements of CSCW environments. Nahrstedt et al. in [13] describe a QoS architecture and the required mappings from application-level requirements to system-level and network-level requirements. The paper does not address network control protocols, specifically routing. Zinky et al. in [17] apply an AOP style approach to QoS in a "Quality of Service for CORBA Objects (QuO)" architecture. [17] only describes QoS support at the CORBA layer. It does not address issues related to network protocols.

As future work, we plan to experiment with a simulation model implementing our architecture, and to study aspect oriented languages to develop adaptive robust code for CSCW applications and multicast routing protocols. We will also investigate the interactions between CSCW systems and other network services, for example, error and flow control.

References

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

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

[3] P. Dewan and J. Riedl. Toward Computer-Supported Concurrent Software Engineering. *IEEE Computer*, 26(1):12–15, January 1993.

[4] C. Ellis, S. Gibbs, and G. Rein. Groupware: Some Issues and Experiences. *Communications of the ACM*, 34(1):38–58, January 1990.

[5] G. Kiczales et al. www.parc.xerox.com/spl/projects/aop/position.html.

[6] I. Grief, editor. *Computer-Supported Cooperative Work: A Book of Readings.* Morgan Kaufman Publishers, San Meteo, California, 1988.

[7] V. Jacobson. Multimedia Conferencing on the Internet. SIGCOMM '94 Tutorial, August 1994.

[8] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos. Multicast Routing for Multimedia Communication. *IEEE/ACM Transactions on Networking*, 1(3):286–292, June 1993.

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

[10] K.J. Lieberherr. *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns.* PWS Publishing Company, 1996.

[11] I. Matta and M. Eltoweissy. A Scalable QoS Routing Architecture for Real-Time CSCW Applications. NU-CCS-97-16, Northeastern University, College of Computer Science, December 1997. To appear in IEEE RTAS '98.

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

[13] K. Nahrstedt and J.M. Smith. The QoS Broker. *IEEE Multimedia*, 2(1):53–67, Spring 1995.

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

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

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

[17] J.A. Zinky, D.E. Bakken, and R.D. Schantz. Architectural Support for Quality of Service for CORBA Objects. *Theory and Practice of Systems*, 3(1), January 1997.