High Performance Publisher/Subscriber Communication For Adaptive, Collaborative Web-based Learning

Yugyung Lee, School of Interdisciplinary Computing and Engineering, University of Missouri-Kansas City
Markus Junginger, School of Interdisciplinary Computing and Engineering, University of Missouri-Kansas City
James Geller, Computer Science Department, New Jersey Institute of Technology

ABSTRACT

The inability of current Internet-based learning systems to deal with different kinds of learners and the high cost of developing a new learning system are well known problems in e-learning. To overcome these problems, we have designed a learning middleware suite, called the Collaborative and Sharable Learning (CoSL) system. CoSL is a tool set for building e-learning systems and for developing course materials for an e-learning environment. Given the geographic distribution of the components of a Web-based e-learning system, it is crucial to support fast communication between providers and consumers of e-learning course materials. In this paper, a high performance Publisher/Subscriber communication system has been designed for real-time communication between participants. CoSL allows us to build and manage global real-time learning systems in a distributed and heterogeneous environment.

Keywords: Publisher/Subscriber communication, Adaptive learning, Collaborative learning, Component-based system, Learning channel, User model, Knowledge model, Game

INTRODUCTION

The Internet holds out great promise for revolutionizing education at every level. It combines the features of traditional distance learning methods with the best of multimedia technology. Thus, it eliminates the need to travel and the facility costs of a traditional school setting. While the immediacy of a classroom setting is still lacking in most e-learning systems, e-learning can be used to improve the quality of traditional instruction. Thus, students can be given individual access to simulation models and can email comments to classmates during an e-lecture. Technologies such as digital video will allow further improvements of
distance education (Internet2). Many efforts have been made to build Learning Management Systems (LMSs) to act as the core of online learning systems.

The major challenge for an Internet-based LMS is to provide integrated and cooperative means of access, because a global learning system is based on heterogeneous, distributed and autonomous data sources. Our solution to this problem is to provide educational middleware that allows the sharing and redistribution of learning objects, i.e., a sequence of course contents packaged as a module (Wiley), the personalization of delivery across a network and the collaboration with other institutions. Such an LMS allows to plug-in executable contents ("Assignment Units") on the learner's interface and exchange data with that content, further enabling the dynamic presentation of course contents and the collection of student behavior data.

A typical LMS performs the tasks of managing online learning, keeping track of student progress, and recording course completion (Thyfault, 2000). Adaptive learning environments usually support adaptive annotation, hiding and presentation of course contents, and generation of study problems of the appropriate difficulty (Beck, 1997). An adaptive educational system should incorporate pedagogical strategies, and apply them based on a student's psychological profile (Sternberg, 1997). However, our focus in this paper is not on the pedagogical issues, but on the technological issues. No e-learning system will be well received, if its response times are inadequate. We describe an architecture called the Collaborative and Sharable Learning (CoSL) system, which provides a way to build and manage global learning systems in a distributed and heterogeneous environment. The major contribution of this paper is a high performance Publisher/Subscriber communication system, which was developed to support the communication within CoSL. XML-based interactions between components simplify heavy-duty data transformations required in a typical middleware system.

COLLABORATIVE AND SHAREABLE LEARNING SYSTEMS

Our solution to the problems in distributed learning is to provide an educational middleware system. The CoSL system acts as the core of an infrastructure composed of local Learning Component Systems (LCSs), Learning Channel, and Learning Middleware. Figure 1 shows the CoSL framework. Our CoSL system has been designed following the paradigm of component-oriented development (D'Souza, 1999). Each learning component has contents, activities, rules, constraints and operations to deal with extension or modification of generic learning patterns (using composition and substitution). These components are autonomous with no need for a master component.

The objects within a component can be defined using a type model (D'Souza, 1999), which describes the state of the objects or the component(s). Each learning object is described with a type, attributes and associations, which represent relationships to other objects. Learning components are considered to be self-contained and loosely coupled. A Publisher/Subscriber subsystem supports
the communication between the components of this architecture. The control structure of the CoSL system is partitioned into two layers: the Learning Middleware (L-Middleware) layer and the Learning Component System (LCS) layer.

All components and all object types are constructed as modules with interfaces, which are specified in XML. The three important ingredients of our CoSL framework (Learning Channel, LCS, and L-Middleware) will be described in the following sections.

**Learning Channel**

All the communication between LCSs or LCSs and L-Middleware in CoSL is done through the Learning Channel. The Learning Channel supports advanced communication mechanisms such as registration and broadcasting (Figure 1). Thus, collaborative interactions between participants can be established through the Learning Channel. An LCS can be either a Publisher or a Subscriber (Figure 2). In our model, a topic can be a motivation to form a virtual learning group. The group is a collection of cooperating participants providing and using shareable learning services.

The Learning Channel is inspired by the Event Channel (Bartlett, 2001) and is based on a high performance Publisher/Subscriber communication model. The Publisher/Subscriber topology applies the Publisher/Subscriber messaging concept at the network level. Each Subscriber machine is connected through specific Topics to Publisher machines. The communication topology can become semi-centralized (a few Publishers serving many Subscribers) or decentralized (many Publishers serve a small number of Subscribers) on demand. One of the advantages of the Publisher/Subscriber topology is that no servers are involved. Because messages are delivered directly

Figure 1. Collaborative and Shareable Learning System
from the source, this approach can be more efficient than indirect ones.

**Figure 2. Publisher/Subscriber Communication**

The idea behind our CoSL communication subsystem was to build a lightweight but extensible protocol, which is optimized for performance, is easy to use (i.e., no additional message infrastructure is required) and can be integrated seamlessly with other applications. **Learning Channels** are asynchronous communication channels for sending and receiving messages on a specific learning topic. **Learning Channels** are virtual in that a message channel’s endpoint can be bound to one or more **LCS** endpoints. A **Learning Channel** is dynamically bound to each **LCS** at runtime, so different **LCSs** can be bound at different times. A **Learning Channel** connects multiple **LCS** endpoints together, from one output topic channel to one or more input topic channels. In CoSL, any number of unicast and multicast protocols and their combinations can be used. In fact, one **Learning Channel** can be created by chaining several sections together, such that each section of the chain uses an entirely different transport protocol.

Our communication model is similar to other message systems in that Publishers are associated with Subscribers through Topics. However, CoSL has some unique features.

- CoSL is server-less, while most other systems rely on Message Servers or other infrastructure, like ORBs or COM services. Any Java application can use the CoSL protocol without the need of additional administration.
- CoSL allows multiple connections (e.g., TCP, UDP, local, etc.) at the same time. While the connection initialization process uses higher-level techniques like RMI, the actual message delivery is done differently. Our experimental results (Junginger and Lee, 2002) show that communication using TCP and UDP is more efficient than using standard Middleware protocols.
- The CoSL architecture is independent from how the communication is implemented (a connection type can be any Java class, which implements the connection interface). Thus, it is extensible.
- CoSL has a generic concept of connection selection. The optimal connection is dynamically selected for specific needs. For this selection, a number of QoS parameters can be considered: message priority, essentiality, and security. Message essentiality allows a sender to choose a connection type according to its needs. Thus, non-essential messages can be transferred by an unreliable protocol (e.g., UDP), which may be more efficient. Similarly, a secure but slow connection can be used for sensitive data.
• Besides the standard Internet connection implementations (e.g., TCP and UDP), CoSL has an optimized implementation for local connections. A local connection requires LCSs to be on the same Java virtual machine. In this case, no networking overhead is involved. This makes CoSL very suitable for local communications. A Return Channel enables Subscribers to communicate back to Return Channel Listeners.

• To support interactions through the Learning Channel, each component has an interface, called a "port." This interface includes typical facility services that can be used to access the knowledge component, learner component and presentation component within any LCS. These ports function as sockets and a local LCS can be plugged into the Learning Channel to become a part of the global learning system. In order to build and reuse ports, we use templates and categorize ports. With a template, the properties of a port are specified. These include the port name, the local learning system name, the LCS name and release number, the host machine name and network address, the learning topic and its presentation modes. For the purpose of reuse, all existing ports are categorized by their properties. Thus, a port can be reused as the basis for building a new port.

In summary, the CoSL features include lightweight, server-less, high-performance, highly extensible, multiple and multi-type connections including TCP, UDP, and local connections. The lightweight CoSL communication subsystem eases deployment of distributed learning applications. Further, it does not

Figure 3. Use Case diagram for the Publisher/Subscriber Communication System Learning Middleware

Copyright 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
rely on additional services (e.g. CORBA), except for a Java virtual machine.

The Communication System

The communication subsystem of CoSL is based on the Publisher/Subscriber model (Figure 3). A Publisher provides methods such as creating a Topic, publishing to a Topic, deleting a Topic, publishing to one specified Subscriber, enforcing Message delivery, etc. The Subscriber can subscribe to a Topic, receive Messages on a Topic, and unsubscribe from a Topic. For the specific needs of LCS, the Publisher offers the communication functionality needed for the server. The clients could be Subscribers to the LCS-Topic. But they also need to send Messages to the Publisher, e.g. ask for addition material.

There are two main scenarios for the LCS Publisher to send a Message. The first one is sharing or providing information among LCSs. Another motivation is that a Subscriber sends feedback and the Publisher wants to respond to it in a general way. Due to space limitations, a detailed discussion must be deferred to (Junginger, 2003).

Learning Middleware

In the CoSL architecture we provide distributed collaboration between networked LCSs through our middleware (L-Middleware). The CoSL framework relies on cooperative agents for supporting an adaptive learning environment. Software agent frameworks have gained much attention as alternatives to distributed static object frameworks, because of their social abilities, intelligent decision-making, concurrent execution and mobility (Wooldridge, 1995). Distributed collaboration in the L-Middleware layer is facilitated through the use of these agent types. Agents can communicate locally.

Figure 4. The Learning Component System
through internal communication links and with agents in other LCSs through the Learning Channel.

LEARNING COMPONENT SYSTEM

The lower layer of CoSL is composed of multiple Learning Component Systems (LCSs). In an LCS, multiple agents collaborate to achieve adaptive learning. Agents are activated by a request for a learning episode and the consequent Knowledge Component and Learner Component retrievals. Guided by the current learning situation, they dynamically form an adaptive sharable learning environment that fits the current learning process. There are three types of agents (Learner, Knowledge, and Presentation) in the LCS. Each agent concentrates on its own specialized tasks and resources. A prototype of the Learning Component System (LCS) is currently being implemented using JAVA and XML (Figure 4).

The Learner Component of LCS

The key to enable customizing a course presentation is a model of the learner. The Learner Component provides a Learner Profile representing the learner's generalized objective, behavior, learning performance and cognitive style. A Learner Profile is constructed from a learner's observed behavior. By studying the learner's cognitive behavior the agents can dynamically alter the learning contents and presentation over time. Specifically, the agents participate in a number of important tasks such as analysis of the learning, retrieval of resources, delivery of services to learners and adaptation to new learning patterns and services. Learners differ in their learning objectives. For example, a learner may be a busy manager, interested in a short course, or may be a student who has a whole semester for taking a course. Also, the learner has an associated cognitive style preference. For example, some learners prefer first getting a subject overview, whereas others prefer to focus on a specific topic and to ignore the big picture. These two styles roughly correspond to the global and local cognitive styles and can be applied to customizing lessons.

We have adopted Extensible Markup Language (XML) for specifying interactions between components in CoSL. For instance, an undergraduate student's Learner Profile, written in XML, can be used on different LCSs across departments. Table 1 shows parts of a Learner Profile.

Table 1. Learner Profile in XML (partial)

```xml
<!DOCTYPE DOCUMENT [<!ELEMENT DOCUMENT (LEARNER_PROFILE)>]
<!ELEMENT LEARNER_PROFILE (COGNITIVE_STYLE)>]
<!ATTLIST LEARNER_PROFILE LEARNER_ID ID #REQUIRED>
<!ELEMENT ORDER_ORIENTATION (In-Order|No-Order)>]
<!ELEMENT PROBLEM_SOLVING_ORIENTATION (Top-Down|Bottom-Up)>]
<!ELEMENT DIRECTION_ORIENTATION (Global|Local)>]
<!ELEMENT VIEW_ORIENTATION (Internal|External)>]
<!ELEMENT INTERACTION_ORIENTATION (Collaborative|Individual)>]
<!ELEMENT FOCUS_ORIENTATION (Parallel|Sequential)>]
<!ELEMENT FLEXIBILITY_ORIENTATION (Liberal|Conservative)>]
```

The Knowledge Component of LCS

The domain knowledge to be covered needs to be efficiently organized for quick
retrieval and update. In this research, a concept hierarchy is constructed to represent content to be covered in Web-based learning. The hierarchy is a directed acyclic graph (DAG) of nodes and links. A node represents a content element and a link represents a relationship between content elements. The hierarchy represents concepts at different abstraction levels and contains multiple relationships including Is-a, Part-of, Contained-in, Assoc-with, Related-to, Example-of, Applicable-to, Easier-than, etc. in a single hierarchy (Lee and Geller, 2002).

The hierarchy is used to determine the level of abstraction for a given topic, to interpret relationships between topics, etc. Pointers to lecture notes, tests and homework problems are attached to topic nodes. The Knowledge Component manages level of difficulty, type of knowledge and problem solving behavior. Knowledge is declarative, procedural, or structural. There are three levels of difficulty (hard, medium and easy). The problem-solving behavior is top-down or bottom-up. The KCA (Knowledge Component Agent) determines the topic and type of knowledge (declarative, procedural or structural) to be presented to the learner.

The Presentation Component of LCS

The Presentation Component responds to the learning proclivities of the learner as shown by the learner's profile. We implemented two Presentation Views: E-book View and E-Game View. A typical learning episode may require a combination of different Presentation Views. Presentation Views consider the instruction mode (guided, self-learning and game-oriented), presentation mode (audio, video, and text), presentation strategy (top-down, bottom-up, local, global, hierarchical, and flat) and instruction mode, (model-based, problem-solving-based or theory-based).

The E-Game View (Figure 5) has a dual purpose: to quiz a student about class material and to let him play a game he will hopefully enjoy. Essentially, the game is the well-known arcade game PacMan with integrated quiz questions. When the game is over, the quiz score will be returned to the Learner Component.

We have initiated a set of experiments using our CoSL framework with about 60 students in a Discrete Mathematics course in Computer Science. As the first outcome of these experiments, we were able to collect data for learner teaching profiles. These will enable the CoSL system to customize course materials to fit the learning preferences of individual students and also to optimize its own teaching.
strategy. Further experiments are planned to confirm whether our CoSL system can be used to reinforce the concepts covered in a traditional class.

COMPARISON OF THE COSL COMMUNICATION SUBSYSTEM

For our purposes, middleware is a layer of software, which exists between the Internet and multiple software applications, to provide the connectivity between applications running on different systems (Emmerich, 2000). Today's most popular examples of middleware include the Common Object Request Broker Architecture (CORBA), Microsoft's Component Object (COM) and Enterprise Java Bean (EJB). Services provided by middleware include communication, identification, concurrency, fault tolerance, and security. In this section, we will compare CoSL with COM+, CORBA, JAVA Message Service (JMS) and EJB.

COM+ offers a distributed event system (MSDN, 2001), which is based on the Publisher/Subscriber model of communication ideally decoupled between Publishers and Subscribers. COM+ provides the efficient use of communication mechanisms: local method call, inter-process call and remote call (Emmerich, 2000). CORBA provides synchronous and asynchronous communications through the Notification Service (Bartlett, 2001) and the Event Service (OMG, 2001). The Event Service is a layer on top of the ORB (Object Request Broker) architecture. An event channel can support consumers and suppliers of an event channel by producing events on demand and the consumers by discovering the event types offered by the suppliers. It determines QoS according to message attributes, such as reliability, priority, expiration times, earliest delivery time, maximum events per consumer, order policy, and discard policy. The Notification Service is powerful but it is very complex.

Within Java Message Service (JMS), Publishers and Subscribers create connections and sessions explicitly. Publishers and Subscribers communicate through message objects. Subscribers have a “Message selector” which allows filtering of messages for Publishers. Similar to the COM+ and the CORBA services, Publishers and Subscribers communicate indirectly, but JMS is not as complex as the CORBA Notification Service. The mechanisms of connections and sessions allow more efficient networking than RMI or ORB calls, e.g. via TCP.

Many implementations of JMS (Gore, 2001) are available. The basic EJB communication method is RMI-IIOP. Recent enhancements of EJB introduced message-driven beans, which integrate JMS into the EJB component platform. The EJB platform can benefit from JMS’s features like asynchronous delivery, exactly-once reliability and QoS parameters. Because of the need of an application server and increased overhead, EJB might not be ideal for high performance communication in distributed computing.

The communication systems are a central factor in determining overall performance and efficiency of a message system. The COM+ Event Service lacks the transparent distribution capability.
Only CORBA and JMS support multicast, which is particularly important when dealing with multiple Subscribers. Only COM+ and CoSL offer an optimized communication mechanism, focused on increasing the efficiency of communication. Lightweight communication (Floyd, 1995) means that the system uses the networking layer directly. COM+ and the CORBA Services are tied to their standard communication methods, RPC and ORB calls, which are far from a lightweight communication implementation.

JMS extends Java-based explicit connections with sessions, but does not define the connection type. Our CoSL communication subsystem is the only system that is offering lightweight TCP and UDP connections by definition, and which is also open to new connection types. The CoSL communication subsystem allows multiple connections and uses a server-less approach (Table 2).

<table>
<thead>
<tr>
<th>Table 2. Comparison of Communication Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM+ Event Service</td>
</tr>
<tr>
<td>Transparent Distribution</td>
</tr>
<tr>
<td>Multicast</td>
</tr>
<tr>
<td>Optimized non-remote communication</td>
</tr>
<tr>
<td>Lightweight TCP communication</td>
</tr>
<tr>
<td>Lightweight UDP communication</td>
</tr>
<tr>
<td>Plug-in points for custom connection types</td>
</tr>
</tbody>
</table>

¹ It depends on the implementation whether this feature is available.

PERFORMANCE EVALUATION

Two experiments were made with CoSL. The first one compared it with RMI and an implementation of the Java Messaging Service (SwiftMQ). We included RMI, because it is one of the standard Java approaches for distributed computing, however, RMI is not capable of group communication.

The Java Messaging System and the SwiftMQ implementation are more similar to CoSL, because they offer Publish/Subscribe communication. Sun’s Java 1.4 Beta 3 Virtual Machine was running in the Server Hotspot mode on a Pentium III 866 MHz with 256 MB RAM. The results show the communication of two separate Virtual Machines on a single machine, i.e., without network connection.

Copyright 2003, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
Figure 6 illustrates message rates depending on the message size. The results indicate that CoSL provides exceptionally high rates for small messages, while both RMI and SwiftMQ provide only relatively moderate rates. The rate decreases with an increasing message size, but that is rather related to TCP data processing than to the systems themselves. Therefore, the results for higher message sizes differ less.

*Figure 6. Message rates*

The throughput (data rate) is outlined by Figure 7. The results indicate that an efficient data communication can be realized especially for small messages in CoSL.

*Figure 7. Throughput*

Figure 8 confirms the previous results in a network experiment. The test environment consisted of two identical machines, which were connected with a 100 Mbps Ethernet network. Each of the machines was equipped with a 1.4 GHz Pentium 4 CPU, and 256 MB RAM.

Figure 8 shows that CoSL has a better data rate than SwiftMQ, especially for small messages. At a certain point, the network bandwidth is reached, which is revealed by the stagnation of the rates. Both experiments provided evidence for the efficiency of CoSL. In contrast to other systems, it can handle small messages efficiently.
RELATED WORK

Individual or tailored instruction based on learners' needs and background has not been achieved yet (Luchini, 1998). Researchers have investigated psychological variables affecting the quality of online learning. They have identified different variables, e.g., learning styles (Cox, 2000). The importance of user modeling has been emphasized in the learning community. Milne et al. (Milne, 1996) focused on the development of composite learner models, incorporating both domain-related data and information about personal attributes such as capabilities and preferences. Schwab et al. (Schwab, 2000) used machine learning methods to acquire a user-interest profile from observed user behavior. Wesley et al. (Wesley, 1999) used distributed rational agents to manage the acquisition and presentation of multimedia information in a distance-learning environment. El-Khouly, 1999) developed intelligent computer aided instruction (ICAI) software using agent technology. Rickel et al. (Rickel, 1997) introduced two types of agents, students and pedagogical agents, communicating using virtual reality. Some researchers focused on the interactions of multiple agents (Maes, 1996) and user models and their maintenance (Brown, 1998).

The use of XML has rapidly increased, due to its flexible and powerful features, e.g. in MathWeb (Franke, 1999). A study demonstrated the ineffectiveness of totally self-guided learning and suggested efficient ('intelligent') combinations between externally guided and self-guided learning (Schindler, 1986). Gustafson et al. (Gustafson, 1998) built instructional models to provide conceptual and communication tools that were used to visualize, direct and manage processes for generating episodes of guided learning. An example of game-based learning is "dialogue game" (Ravenscroft, 2000). Another study showed that game-oriented learning is more effective than straight tutorial animation (Ford, 1993).

CONCLUSION

Our high performance Publisher/Subscriber communication system supports flexible and efficient communications between agents in CoSL. A prototype of the CoSL system has successfully employed XML-based data exchange and agent-based communication. Experiments with our Publisher/Subscriber communication subsystem show that it is faster than JMS and RMI.
REFERENCES


Yugyung Lee is an Assistant Professor at the University of Missouri at Kansas City. She received a B.S. in Computer Science from the University of Washington in 1990 and a Ph.D. in Computer and Information Sciences from the New Jersey Institute of Technology in 1997. Before joining UMKC, she was working at MCC. Her research interests include distributed data mining, software agents, middleware, Semantic Web, Web Services and Web-based learning.

Markus Oliver Junginger is currently a Computer Science MS student and research assistant at the UMKC. He is also a candidate for graduation (Diplom Informatiker, FH) at the University of Applied Sciences in Augsburg, Germany. His current research interests focus on middleware and group communication in massively distributed systems like peer-to-peer and grid networks.

James Geller received his M.S. and Ph.D. degrees at the CS Department of the State University of New York at Buffalo. He is currently Professor at the CIS Department of the New Jersey Institute of Technology. He has published in a number of areas including knowledge representation, parallel artificial intelligence, medical informatics, and object-oriented databases. His current research interests concentrate on medical vocabularies, and on Web mining.