Informing Science Special Issue on Multimedia Informing Technologies – Part 2 Volume 3 No 1, 2000

Using a Virtual Room Platform To Build a Multimedia Distance Learning Environment For The Internet

Allen Ginsberg, Dennis Shiau, & Bryan Sampieri
Bell Labs, Lucent Technologies

abg@bell-labs.com

Abstract

Education and training are expected to change dramatically due to the combined impact of the Internet and multimedia technologies. A challenge for internet-based education systems is to allow educators and trainers to work from known models, while simultaneously letting them experiment with innovations without requiring frequent retooling. Work in distance learning, while of interest in its own right, is important to our general research program as a test bed for our ideas. The basic research program is to provide a general foundation, or set of "primitive" classes, for building remote collaboration environments, and then to build toolkits using these classes, and other more specific classes, that will address the particular requirements of various collaboration environments. Our basic premise is that the virtual room concept together with other fundamental concepts provides such a foundation. In this paper we discuss our virtual room platform and a distance-learning environment called PERSYST. PERSYST has been used in several distance-learning trials; we close this paper with a discussion of the trials and what we have learned from them.

Keywords: Distance learning, virtual environments, collaboration, Persyst

Introduction

The rapid rate at which new technologies and related products are invented or updated is certainly one of the defining characteristics of our time. This phenomenon is both a result of, and a driver for, more efficient and effective modes of education. Clearly without a well-educated work force we would not have the kinds of advances that are commonplace today; but it is equally clear that the pace of these advances make education and re-education ever more important to everyone, both as workers as well as consumers.

Just at the time that education and training have taken on increased importance in society due to the pace of technology, we now find that education itself is a prime candidate for reinvention via new technology. The technologies that have brought us the personal computer, the Internet, and multimedia, have made revolutionary change in education a real possibility (Laws, 1996; Lawrence, 1996).

The use of the Internet is expected to dramatically decrease education costs. For example, in the case of corporate training, online education means less traveling for employees, hence lower costs. The Internet should also make it possible for remote students to have access to the renowned instructors in a subject, as well as allow for broader student interactions: instead of just interacting with their local peers, students can interact with their counterparts anywhere in the world. (*Unless otherwise noted, we use the term "Internet" to include intranets as well.*)

Some of the functionality needed for interactive internet-based education is obvious, such as the use of shared audio, video, and objects, such as whiteboards, or applications in a "virtual classroom." Other functionality, such as multimedia recording and indexing (see below), may appear more exotic, but may eventually become commonplace features of such environments.

The system discussed in this paper, Persyst, currently supports all the functionality discussed above as well as additional functions. In this paper, however, we will not discuss video or application sharing at all, since these have been added to the system only recently.

Clearly new technology can impact the ways things are done, and equally clearly, people are often resistant to change. Common sense tells us that users are more likely to accept innovation if they perceive it as arising within a familiar context. For education this means trying to preserve activities such as live interaction among students and instructors, the role of the instructor as lecturer, etc. Therefore, the challenge for internet-based education systems during a transitional era is to allow educators and trainers to work from known models, while simultaneously letting them experiment with innovations without requiring frequent retooling.

Providing non-programmers with the power to easily create and customize instances of distance learning environments is just as important as providing them with such environments at all. Indeed, distance learning is just one example of a remote collaboration environment using multimedia technologies for which the notion of providing a flexible platform for user customization makes a lot of sense. The same considerations apply to areas such as online customer service, technical support services, and so on. Work in distance learning, of interest in its own right, is also a test bed for our attempt to define a set of “primitive” classes for building remote collaboration environments, and then to build toolkits using these classes, and other more specific classes, that will address the particular requirements of various collaboration environments. Our basic premise is that the virtual room concept (Ahuja, Ensor, and Horn, 1988), together with other fundamental concepts, provides such a foundation. The first step in achieving this goal is to build a software development platform using these basic general concepts, and then show how specific instances of collaboration environments, such as distance learning systems, can be built on top of this platform. Once we have built environments on top of our platform we will deal with the task of providing toolkits that will enable others to do so as well.

The virtual room platform consists of core classes that include ***User, Room, RoomObject***, and ***Service*** classes, as well as other classes, and classes derived from these. A *room* is a place wherein a collaboration takes place, *users* are the people who participate in it, *roomObjects* are the objects that the users need to use in their work and therefore are present in a room, and can be moved from room to room. S*ervices* are whatever enables tasks to be done.

Various classes are derived from the core classes. Consider two types of collaboration domains: education and customer care. While some derived classes are derived directly from core classes - for example, *student* and *customer* derive from *user* - others are derived from some common intermediate class, e.g., *instructor* and *agent* from *employee*. Thus while developing differing collaboration environments will indeed entail “extra” work, the fact is that, as more and more derived classes are created, greater opportunities for reusing classes will emerge. It is also important to remember that the “extra” work of deriving specialized classes is something that we expect the relevant users to perform *themselves* using toolkits that will be made available to them.

PERSYST

In this paper we use the name “PERSYST” to refer to the distance learning system we developed using a virtual room platform. PERSYST supports online courses that are accessible anytime and anywhere with only a Web-browser at the client side (except for plugins required for various third-party services, such as Internet audio and video that we discuss below). PERSYST supports real-time, synchronous interactions, such as lectures, as well as asynchronous activities, such as collaborating on team assignments. A course in PERSYST consists of a collection of 'electronic places' including a classroom where a teacher can electronically share a presentation, a library that is a repository of all lectures as well as other reference materials, breakout rooms - a collection of rooms for students to break into teams, a student lounge to allow students to chat, as well as others.

A user can be registered in a number of courses, either as a student or an instructor. Upon logging into the system, the courses to which the user belongs are displayed, as shown in Figure 1. The user then selects the course environment that he would like to enter. Upon entering this environment, the user sees a representation or map of the overall structure of the

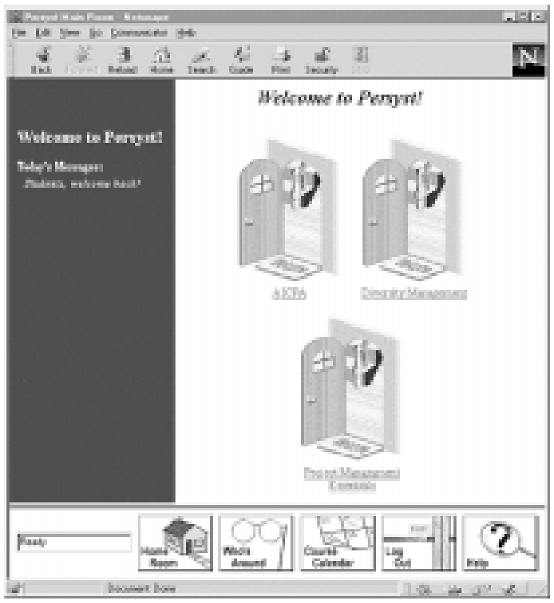


Figure 1: Available Course Rooms

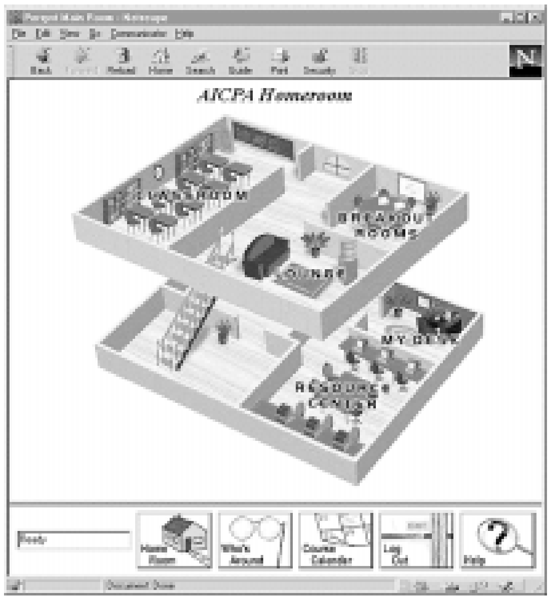


Figure 2: A Course Room

course, as shown in Figure 2. This map shows the available sub-rooms and services for the course. Users can enter the sub-rooms and use any of the services to engage in whatever activities are scheduled or are appropriate for that sub-room. In this paper we focus on the area of the course called the *classroom*. Aside from constraints on the length on this paper, we focus on the classroom because it is the center of live interaction and multimedia technology in PERSYST. First we briefly describe some of the other course areas.

The area labeled *My Desk* in Figure 2 leads to a personal work space that is available to each user. Instructors can populate students' My Desk rooms with materials to read, assignments to complete, and tests to take. Students could also use this space as a personal storage area for things relating to the course.

Breakout Rooms are sub-rooms designed for groups of students to interact, synchronously or asynchronously, to complete group activities assigned by the instructor. The instructor assigns each student to a group (room) and the system automatically places each user in the appropriate room when he/she clicks on the Breakout Room part of the course map. A message board allows students to send comments to the group's leader or Internet audio can be used in the room in the same manner as in the Classroom (see section 2.1).

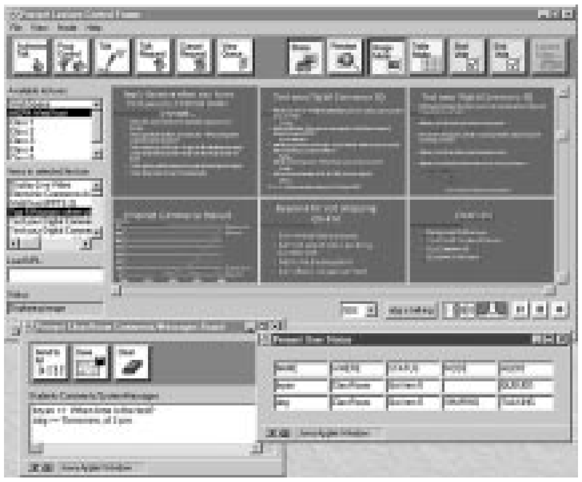


Figure 3: Instructor's view of the Classroom.

The *Lounge* is simply a text chat room that lets members of the course chat among themselves at any time. *Who's Around* is a service that lets users see who else is in the course or one of its sub-rooms at the current time. The *Resource Center* is used to hold a number of different sub-rooms and services. The *Library* is a repository of files that members of the course can access and write to. Users can place various content, such as a course syllabus and a list of relevant readings or stored lectures and lecture items, here for access at any time. *Groups Administration* is a service that is only accessible to users with administration level permissions, such as instructors. This service allows the user to dynamically determine which students are assigned to which Breakout Room. A particular user can be assigned as the group's leader, who would be responsible for over-seeing the activities of the group.

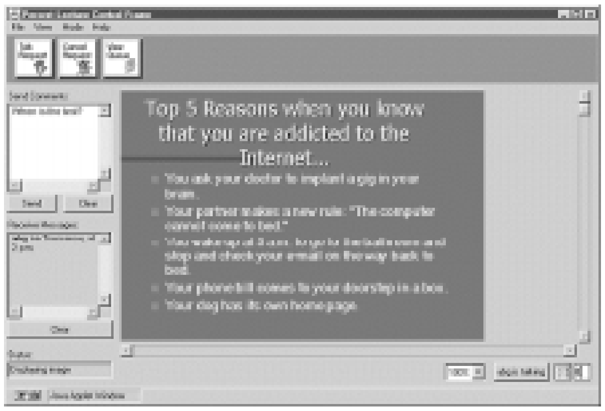
The PERSYST classroom and its tools

The *Classroom* [figures 3 – 4] is the sub-room that is used for live instruction and interaction between instructors and students. It contains "blackboard," text chat, and audio services. The blackboard service refers to the ability of the instructor to "force" items to be displayed on everyone's view of the classroom. This is used to show images, video, graphics, etc. during a lecture. The audio service has three functions: first, to provide live audio for class instruction, second, to allow the instructor to play recorded audio material to the class, and third, to record all the audio data in the session. The reason for doing this is discussed below..

As can be seen by comparing Figures 3 and 4, instructor and student views of the classroom are different. For example, the classroom contains a view of the course library for accessing lecture materials that will be used to provide instruction, but only the instructor can access it. The instructor and student views of the text and audio services also differ. The audio service, for example, which is a one-to-many service (one person talks, all others listen), is under the control of the instructor. As can be seen in Figure 4, students can request to be allowed to speak. It is up to the instructors to enable a student to talk, and they can do so whether or not that student has requested to speak. However, the instructor can "grab the mike," so to speak, from any student at any time. There is also an asymmetry with regard to the text chat in the classroom: students can only send comments or questions to the instructor, whereas the instructor can send messages to the class as a whole. However, an instructor can turn over full control of the classroom to another participant, be it a second instructor or teaching assistant, or a student in the class.

This classroom control regime may seem unnecessarily restrictive. However, there are a number of justifications for imposing this structure on the classroom. First of all, with regard to the Internet audio and its one-to-many nature: this is partially due to technological and platform limitations that we encountered at development time. Some of these limitations, e.g., high quality bridging of multiple compressed speech endpoints, have now been overcome. (*Elemedia*, a wholly owned software venture with Lucent Technologies, recently announced a number of packet-based audio bridges; see http://www.elemedia.com) However, the main platform limitation still exists – lack of full-duplex audio cards on end-users' platforms.

The main reason, however, for imposing this structure on the classroom is that it reflects the way classes are usually conducted in most education and training settings. The instructor generally does the talking, and presents his/her prepared lecture materials to the class. Students can raise their hands (request to speak), and the instructor can "call on them" whether


Top 5 Reasons when you know that you are addicted to the Internet...

- You ask your doctor to implant a gig in your brain.
- Your partner makes a new rule: "The computer cannot come to bed."
- You wake up at 3 a.m. to go to the bathroom and stop and check your email on the way back to bed.
- Your phone bill comes to your doorstep in a box.
- Your dog has its own home page.

Figure 4: Student's view of the Classroom.

or not they have raised their hands. When one person talks to the class everyone else listens, except that the instructor can cut someone off. Students are *not* supposed to talk to one another during the lecture.

An interesting way in which a PERSYST classroom differs from its traditional counterparts is the ability of students to send text comments or questions to the instructor *at any time*. The instructor can ignore these remarks or respond as warranted by sending a text response, or by using the audio channel. This enables students to get the instructor's attention immediately regardless of the state of the audio queue. It also lets the instructor know of any problems or issues that need to be addressed without disrupting the lecture. The impetus for and refinements of this idea came from trial participants (see below) - an example of people coming up with an innovation within the context of a familiar environment.

In the classroom, an instructor uses the blackboard service to get students to see text, images, graphics, and web sites. For the most part, lecture items will be prepared in advance using third-party content authoring toolkits; the two exceptions will be discussed below. The instructor, or course designer, supplies a name for the lecture, a name for each item in the lecture as well as a pathname or URL for its location, and a default sequencing of the lecture items. The actual sequence in which items are presented is under the control of the instructor. As shown in Figure 3, the lecturer can preview the items in a lecture, and use "thumbnails" of the items so that more of them can be previewed on the screen at the same time. This use of thumbnails is also an example of a feature that was suggested by a trial participant.

A critical requirement of our second major trial was enabling the instructor to send students a table of options that might need to be altered in real-time, and then allowing students to select several options from the table in order of priority. The results needed to be immediately displayed in another table sent to everyone in the classroom. To meet this requirement we developed a general *table* class in Java that can be specialized in certain ways, e.g., a *voting table* is a table whose cells are labeled buttons that are pressed to indicate a vote. Tables can be created from scratch or altered during class time, or prior to class time, and saved for future use in the *WorkSpace* that PERSYST automatically creates for every classroom.

This experience with tables shows that requirements for specific courses can lead to features that are of general use across a wide variety of domains.

A requirement that became apparent in our first trial was the need for lecturers to know when all the students had received the last sent lecture item on their blackboards before proceed-

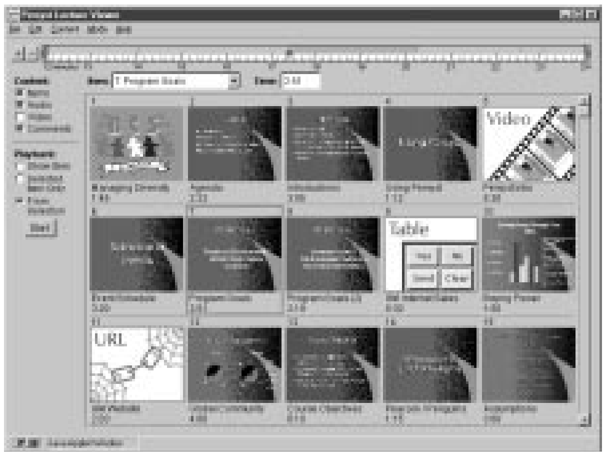


Figure 5: Multimedia index of recorded lecture.

ing to talk about it. Our first trial was conducted using a telephone bridge and while it was easy for the instructor to ask the students whether or not they had received the current lecture item, it was annoying. With the one-to-many regime of our Internet audio service this would be excruciating.

We dealt with this issue by developing the user-status panel shown in Figure 3. This panel is specifically designed for use in the classroom. For every participant it shows and dynamically updates the item number of the last item that they have received. The instructor can wait until everyone's last received item number matches. An item can be resent if a student does not receive it after a reasonable amount of time.

If a classroom contains a large number of students (anything over 15 to 20), it is unlikely that the instructor will be able to manage these kinds of issues herself. We expect some kind of technical and administrative support to be present on these occasions. For this, and other reasons, the user status tool described above should be generalized into a general administrative tool that will show what is happening, relative to any service or media, in a classroom or elsewhere in a course, or in

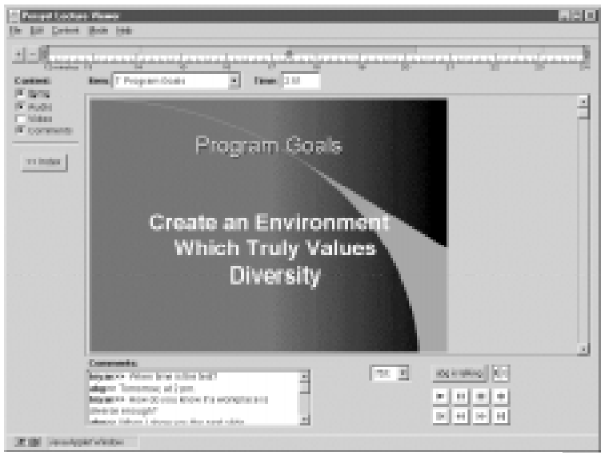


Figure 6: User interface for lecture replay.

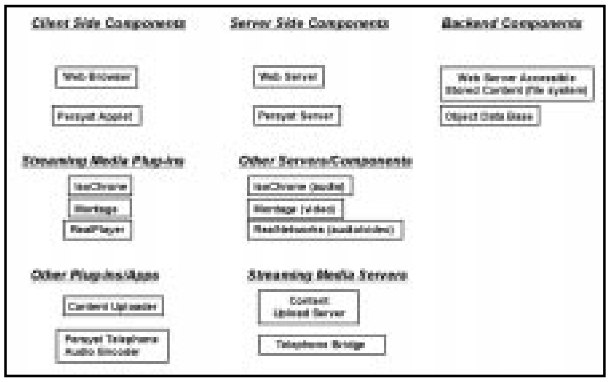


Figure 7: PERSYST platform overview.

any other course in the system. In the case of a classroom, once a problem has been identified, and is determined to involve a single user or a small number of users, one would like to be able to handle it without halting or disrupting the lecture.

Multimedia recording and indexing

The notion of recording related simultaneous media streams for subsequent replay is one that we explored in earlier papers (Gabbe, Ginsberg, & Robinson, 1994; Ginsberg & Ahuja, 1996). In those papers we also introduced the notion of *multimedia indexing*, which refers to a process whereby recorded media streams are automatically indexed by events such as a user's starting/stopping to speak, or a user's sharing an image, and so on. Such indexing makes it possible to create a visual map of the recorded streams that enables informed random access of the recording.

As can be seen in Figure 3, the instructor has the option of creating an indexed recording of whatever happens in the classroom. Thus a lecture can be recorded and saved in the library for students to review at their leisure. At the time of the PERSYST trials only the audio and video recording features were available. The complete indexing capability will be available in subsequent trials. Figures 5 and 6 show the nature of the index created by this service and the interface that a user will employ to replay the recordings.

PERSYST Architecture

Over the course of several trials, a large number of services have been used in conjunction with PERSYST. The platform components are shown in Figure 7. Some of these components are not likely to be used together in the same course. For example, the Montage video server and plugin (Gaglianello & Cash, 1995) would be used for higher-bandwidth environments, while a RealNetworks server and plugin would be used for low-bandwidth situations. There is no single optimal solution for every case. It is therefore important that the system

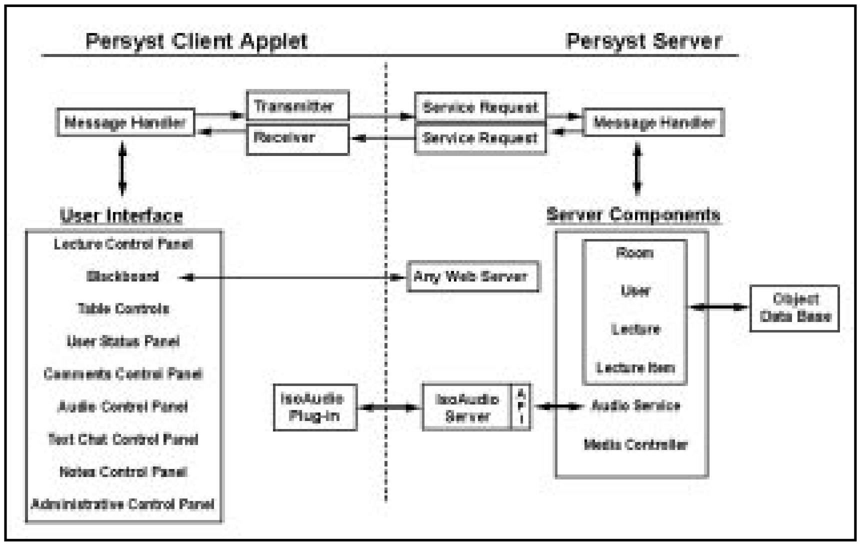


Figure 8: PERSYST Java-based architecture.

architecture be designed in a manner that makes it easy to add or remove components and design interfaces for them.

As a result of accommodating all these services, the current architecture is really three separate architectures. There is a Java-based architecture, shown in Figure 8, a client-server model that uses applets and other dynamically downloaded classes on the client side. This is our most recent architecture and represents the core design upon which we will build in the future. There is also RPC based architecture, shown in Figure 9. This is also a client-server model, but it can come in one of two flavors: one in which the client is a plugin or standalone application that is installed prior to course-startup-time or one in which the "client" is nothing more than a browser. Finally there is a "vanilla" architecture that requires nothing but a browser and plugin at the client side, with the server talking directly to the plugin.

In this paper we limit our discussion to the Java-based architecture.

As we mentioned earlier, PERSYST can be used anytime, anywhere by any client consisting of a web browser that is Java enabled. Using the browser a client contacts a PERSYST server and logs in. This causes an applet and several Java classes to be downloaded to the client. As a user navigates through the various areas in a course additional Java applets and classes are downloaded as needed. As can be seen in figure 8, these classes fall into two sets: those that deal with the user interface and controls, and those that deal with the fundamental client-server request-reply communication structure. We discuss the later first.

Currently PERSYST uses message-based client-server architecture. That is, every class on the client that needs to communicate to some class on the server, or visa versa, does so by means of messages that are sent back and forth from one to the other. The *messagehandler* classes on the client and the server receive the incoming messages and route them to the correct class as determined by attributes in the message. The *transmitter* class on the client is a thread that is used to send requests to a server that involves *sending items to other clients*. Thus the *lecture control panel* class used by an instructor in the classroom uses the transmitter thread to send requests to the server concerning the items that are to be displayed on students' blackboards. The transmitter is also used to send text messages intended for a text chat service to the server. The *receiver* thread on the client performs the inverse function: it receives messages from the server telling some class on the client what it should do next. Note that the receiver sends its own acknowledgements to the server, if requested, and does not rely on the transmitter for that function. Each of these threads on the client side communicates with a corresponding thread on the server-side, marked *Service Request* in Figure 8.

A virtue of this design is that it allows client functions that involve transmission to and control of other clients and services to be managed and allocated completely independently of basic client functions handled by the receiver. It is possible to be a PERSYST client without having a transmitter at all.

Figure 8 also indicates that the blackboard service operates by communicating with web servers. When an instructor in the classroom commands a certain item to be placed on the blackboard what actually happens is that the PERSYST server sends a URL to every client that is the location of that item on some web server. The blackboard service on the client then contacts that web server and retrieves the item using the standard protocols. In other words, most of the "raw bits" of content never pass through the PERSYST server at all. This way of handling

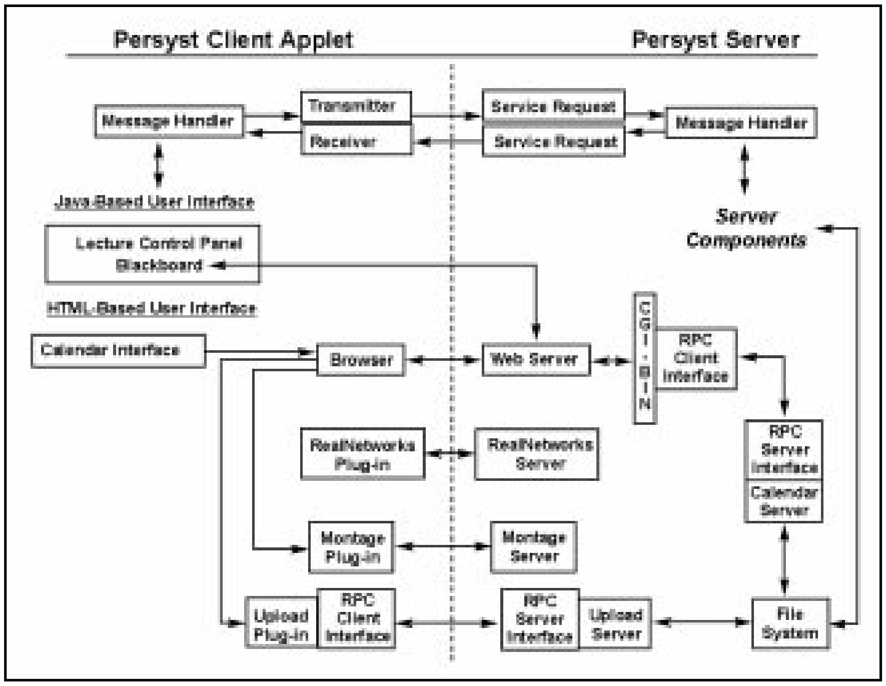


Figure 9: PERSYST 'vanilla' and RPC-based architectures.

access to content is simple and flexible, and it can also remove load from the machine that hosts the PERSYST server.

Figure 8 also shows how we integrated a third-party Internet audio service into this architecture. *Isochrone* (a privately owned company based in Eatontown, NJ, USA; www.isochrone.com) supplied us with three components: a plugin, called *IsoAudio plugin*, for the client side, a server called *IsoAudio server*, and a server side API that allowed us to talk to the IsoAudio server. We had no similar API on the client side for the plugin. From the figure it can be seen that our *Audio Control Panel* class on the client controls the plugin by first communicating with the PERSYST server and then using an Audio Service class to talk to the IsoAudio server which in turn contacts the plugin. While this may seem to be a roundabout way of doing things, the fact is that for the case in which one client (the instructor) wants to control other clients' audio plugins (students), there is no better way.

Finally, as shown in Figure 8, PERSYST relies upon an "object data base" for persistent storage of objects used in the virtual room environments for online courses, e.g., rooms, users, lectures, and so on. Currently there are two implementations of the object data base, one that actually uses an object-oriented data base manager, and one that simply uses files for long term storage of these objects. While the former is certainly preferable on technical grounds and for research purposes, the fact is that "real-world" uses of a system like PERSYST will almost always take place within a context of legacy systems that will impose constraints on system components, particularly the database. We therefore consider the object database to be internal to PERSYST and how it is mapped to/from long term storage may vary from case to case.

Discussion

PERSYST has been used in several trials. Two of these trials involved teaching a course for credit towards a master's degree in project management. These trials had about 12 students and met two times a week for four weeks.[*] Another trial involved a course in diversity management training that lasted one week, with 8 participants. These three trials took place over a corporate intranet and a telephone bridge was used instead of Internet audio. The last trial, in November 1997, took place on the Internet, had 10 participants, utilized our live audio service for class instruction, and live video was used briefly to introduce the instructor[**]. In all the trials, users accessed PERSYST from distant locations all over the country, and in most cases, users were highly bandwidth limited, i.e., 28.8 kbs.

[*] Here is a quote from the instructor's evaluation of the trial: "After taking the PERSYST course students completed the same post-test given in the traditional classroom version. The average score was 86%, with a range of 80 to 90. This is well into the satisfactory range, and on the high side of classroom outcomes. In informal polling, students liked the delivery method and found it easy to learn how to use it."

[**] The bandwidth required for the combined audio and video streams was around 19 kbs (8 kbs audio, 11kbs video). Theoretically this means that a 28.8 connection would contain sufficient leftover bandwidth to allow for lecture items to be sent as well. In practice, connections through Internet Service Providers rarely reach 28.8 kbs. Therefore we turned off the video once the lecture began in earnest.

In all the trials - with or without Internet audio, on the Internet or an intranet - network congestion and delay always presented a problem to a greater or lesser degree. In our first trial we found that the instructor might continue lecturing even though some students had not received the relevant lecture item. In response we developed a *user status* tool that enables anyone to continually see what item has last been received by everyone else, as shown in Figure 3. This has been a useful tool. But a similar problem for Internet audio is, given relative delays among the listeners, how does the instructor know who has heard what at any given time? This problem is not so easily handled. One of our current research directions involves developing a systematic approach to *visualizing* the state of a meeting in a virtual room so that individual problems can be resolved without disrupting the entire group. We believe that such an approach should be developed using the general virtual room platform, and then specialized as required for particular applications. Finally, the trials so far have involved one course on a server at a time. In practice we expect multiple courses to be handled simultaneously within the same environment. Administrators of such an environment will indeed require an even more powerful set of tools to deal with the problems that will inevitable arise.

Acknowledgements

We thank Ramesh Viswanathan for valuable criticisms of earlier versions of this paper. Tony Haro was instrumental in implementing an earlier version of PERSYST and portions of the current system still bear his mark. Vibeke Jensen, Norman McDonald, Cati Laporte, and Ching I-Kao were all involved with this project in its beginning phases, and their contributions are not forgotten.

Teri Lindstrom and Peter Hodge worked on later versions of the system.

References

Ahuja, S.R., Ensor, J.R., and Horn, D.N. (1988). The Rapport Multimedia Conferencing System. Proceedings of the Conference on Office Information Systems.

Gabbe, J., Ginsberg, A., and Robinson B. (1994). Towards Intelligent Recognition of Multimedia Episodes in Real-Time Applications. Proceedings of ACM Multimedia 94.

Gaglianello, R., and Cash, G. (1995). Montage: Continuous presence teleconferencing utilizing compressed domain video bridging. IEEE ICC95.

Ginsberg, A., and Ahuja, S. (1996). Automating envisionment of virtual meeting room histories. Proceedings of ACM Multimedia 96.

Lawrence, B.H. (1996). Online course delivery: Issues of faculty development. Journal of Educational Technology Systems, 25, 127-131.

Laws, R. (1996). Distance learning's explosion on the Internet. Journal of Computing in Higher Education, 7, 48-64.