A Peer-To-Peer Architecture for Synchronous Collaboration over Low-Bandwidth Networks

Meletis Margaritis, Christos Fidas, Nikolaos Avouris, Vassilis Komis

University of Patras, 26500 Rio Patras, Greece
Email: {Margaritis,Fidas,N.Avouris }@ee.upatras.gr,
Komis@upatras.gr

Abstract. The need for peer-to-peer collaboration over the Internet is increasing nowadays. There are many factors that make this task particularly difficult. Software designers and developers often come across the challenge of overcoming the raising technical and organizational problems. These problems include mostly the limited bandwidth of networks especially of dialup connections, but also the presence of firewalls and proxy servers in intranets that inhibit peer-to-peer communication. A flexible architecture needs to be defined that overcomes these problems and permits a smooth collaborative environment that would satisfy the needs of end users. This paper describes a study on these problems and provides solutions accordingly by defining the architecture of a peer-to-peer collaborative system developed and used for real-time collaborative modelling activities.

1 Introduction

Peer-to-peer (p2p) computing applications seem to proliferate recently. Designing such applications involves tackling serious technical and social challenges. According to [10], the technical advantages of such applications are fault tolerance, performance, and security, while the possibility of powerful communication technologies in distributed form, lead to new person-to-person interaction structures. An especially interesting application of p2p technology is that of synchronous collaboration systems, with many uses in work and education [11]. In this paper we discuss the main characteristics of such an environment (ModellingSpace, **MS**) a distributed application, which comprises a suite of interconnected tools to support collaborative modeling activities. MS is an environment that supports individual and collaborative building of various kinds of models. MS includes tools that permit building and editing of primitive multimedia entities, building and exploring models that are constructed using these primitive entities. MS supports synchronous and asynchronous interaction of users, collocated or at a distance who collaborate in building models. The open character of MS means that the users have access to an open set of primitive entities that can be used for building these models.

Synchronous collaborative problem-solving is often based on a shared work surface [6]. As a result communication among partners is done through the constructed

artifact, found in this surface, e.g. a model under construction or the representation of a solution to a given problem. This is done in effect when one users' manipulation of the objects in this surface is observed by the peers. This indirect way of communication can be as important as direct communication [1].

Various architectural decisions are related to the design of the shared work surface. One possibility is to apply a strict WYSIWIS (what you see is what I see) approach in the main work surface window. As a result the activity in this area is faithfully reproduced in all users' workstations. So most of communication and reasoning of users is based on this shared viewpoint, which becomes the main grounding mechanism of dialogue and through which eventually common understanding can occur. However additional operations outside this shared workspace may also be performed independently by partners involved, a model-level coupling approach according to [14]. This way a more relaxed coupling of partners is achieved.

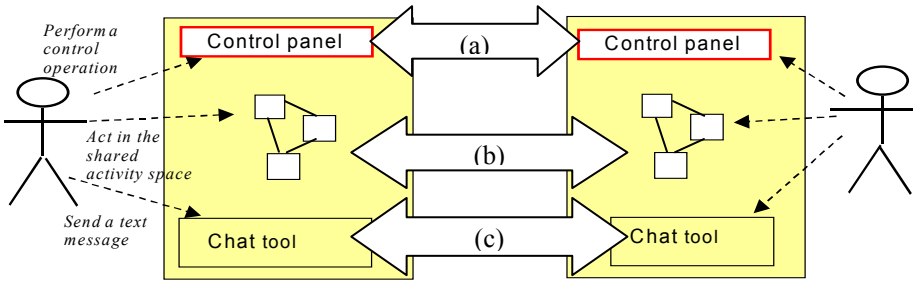

Fig. 1. Peer-to-peer Collaborative environment. The exchanged information concerns (a) coordination control messages (b) shared workspace state-change control messages and (c) chat messages.

Figure 1 shows a typical collaborative activity, which involves two partners at a distance. These two partners interact through a reliable TCP connection, using the socket interface for client to client communication. A set of primitives have been developed, implementing the semantics of the protocols described in this paper. MS is based on the concept of shared artifact, represented in a shared work surface. This artifact can be a jointly built diagram like a concept map, a flow chart or an entity-relationship diagram or a model simulating an activity or a phenomenon. In contrary to some other collaboration applications, in which emphasis is in communication (argumentation tools, decision making support tools etc.), in our case the distant partners collaborate mainly by sharing the model in the work surface, which thus becomes a cognitive space. In this case the communication *through the artifact* is important, where one participant's manipulation of shared objects can be observed by the other participants. A key requirement in this context is to support sharing of a view of the model in synchronous modeling activities over low bandwidth connections, as is often the case with individual users' connections to the Internet. In contrary to other shared workspace environments, like Microsoft NetMeeting, in which heavy graphical information is exchanged among partners, in MS we use a replication of the libraries of primitive entities and tools.

1.1 Types of Exchanged Messages

As a result of this architecture, three types of messages may be exchanged:

(i) *Change-of-state* messages, shown as (b) in fig.1, For example the following message concerns move of object barell_2 to a new position on the screen, this is transmitted to the collaborating peers and the local client engines affect the move of the object.

```
<message>
<ID>Move object</ID>
<user>George</user>
<objectID>barell_2</objectID>
<attributes>
      <x>100</x>
      <y>250</y>
</attributes>
</message>
```

(ii) In addition, support of direct communication among the participants is achieved through an instant messaging tool (*chat messages*), shown as (c) in fig.1. This is a communication mode that has been preferred to audio or video, which is used in other synchronous collaboration environments, as it is more effective in low-bandwidth connections. The effectiveness of this text-based communication has been proven through a number of studies involving pupils of secondary education [9], and higher education students [3]. In these studies the use of chat was supplementary to the observation of activity in the shared activity space.

(iii) Finally *control messages* are exchanged which relate to coordination of the activity, like messages concerning locking of the activity space by one partner. These are shown as messages (a) in fig.1. A more thorough discussion of the alternative coordination mechanisms is included in section 3.

The exchanged messages of type (a) and (b) are a few bytes long, so fluent collaboration can be effected even under low bandwidth conditions, while those of type (c) depend on the size of the typed text message.

As a result, this architecture can scale up to large-size groups of synchronous collaborators, something not feasible in other point to point architectures.

In the following section 1.2 we discuss a specific case when the exchanged information between the partners may be higher than that discussed in section 1.1, when the need arises for exchange of primitive multimedia entities.

1.2 Exchange of Multimedia Entities

It should be noticed that there are many kinds of entities in ModellingSpace. Abstract entities can be represented by textual descriptions, as in figure 1, while other entities may be represented on the work surface through multimedia files, e.g. images and video. Interconnection of such entities through quantitative and semi-quantitative relations [5], can result in complex mathematical models.

In case that a complex entity is used by one of the collaborating partners and cannot be found in peers' workstations during modelling, a need arises to transmit this

entity to collaborating peers in order to synchronize the peer applications. This may result in relatively long download times. A solution for this problem is to send only light control messages directly to the peers, including the structure of the new primitive entities, while the multimedia files associated to these entities, to be send in this case through a server to the requesting peers, without creating disruption to the rest of the group. The details of the protocol used are discussed in section 2.

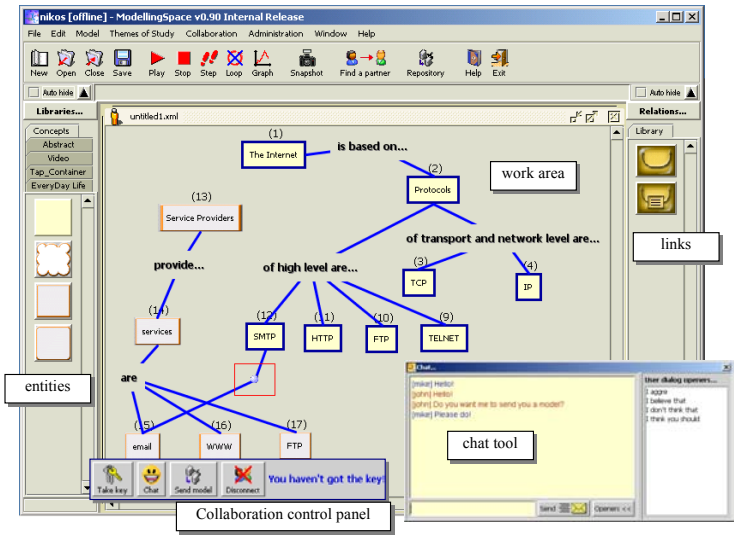

Fig. 2. The ModellingSpace environment

The typical work surface of ModellingSpace is shown in Figure 2. On the left-hand side column of Fig. 2 a library of entities is shown, while on the right hand-side a library of available relations (links) is included; these are the building blocks for modelling. The items included in the main window of Fig.2 (a concept map of the Internet in our example) are reproduced in all collaborating partners windows, through the replicating architecture discussed in this section, which maintains the content of the libraries in all partners sites.

The design of the MS environment has been a challenging process. In particular we have been concerned with mechanisms for coordinating the activity and with mechanisms for overcoming the problems imposed by firewalls and proxy servers, which make establishment of point-to-point connections difficult. In the following we describe the main characteristics of the architecture of the system that has overcome these challenges.

2 Mechanisms for Effective Peer-to-Peer Interaction

The MS architecture is based on a thick client component, which contains a number of interoperable tools. Synchronous collaboration is effected through peer-to-peer interaction. However the proposed architecture contains also a server node (the *Com-*

munity server), which is used as a common repository of information and as a central means for registration and authentication of users participating in collaborative interaction. Many issues related to security and asynchronous interaction can be solved through this server, as proposed by many collaboration support systems architectures, e.g. see [4]. Additional functionality of the server involves support for asynchronous collaboration (asynchronous exchange of messages and files through the tray mechanism, logging of asynchronous interaction), tracking of physical address of online users, information on presence support, i.e. inform users on availability of their peers for synchronous interaction and support for smaller communities (the groups), where most of the activity takes place, by providing them with private space in the repository and private asynchronous interaction support. Finally, these Community Support Tools provide services like group management, session management, registration and login of users, etc, see also [2].

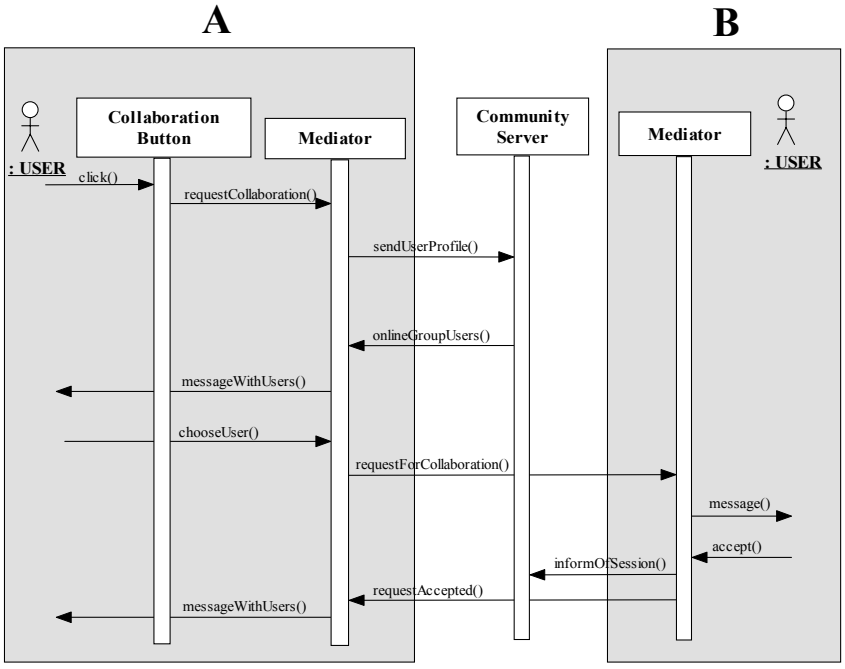

Fig. 3. Initiation of Collaboration session between peers A and B

The collaborative session is established as follows: The user activates *request for synchronous collaboration*, selecting an individual user or a group of users from the on-line users in the server, as shown in the interaction diagram of fig.3. The system checks if a model is in the process of creation in the activity space, in such case the system informs the users that the activity space should be cleared before collaboration can be initiated. The system sends the request to the user(s).

The reply of the user(s) is either acceptance of the request or rejection of the request, if no reply is provided within a time limit a “reject collaboration” is assigned to

the particular user. If the request is accepted by some of the users, the collaboration panel is activated and a chat window is created, as shown in fig.2. If the collaboration request is done in the frame of an existing group, then a collaboration session is established (logging parameters, continuation of previously suspended collaboration session). If the collaboration session is generated by a group coordinator, the coordinator can decide on the collaboration protocol (round robin, key passing mechanism, role playing protocol). Once a collaboration session is initiated, more users can join in or leave the group at any time. This is acknowledged to the other partners.

In the following, two alternative communication mechanisms that have been implemented are described.

2.1 Direct peer-to-peer interaction

As discussed in section 2, the *Community Server* plays a role only during initiation of collaboration. Peer workstations' synchronization is achieved without intervention of the server in this case. The mechanism is based on a set of reactive agents, which try to achieve synchronization with the corresponding agents of the peer host based on a stimulus–response model. So in a joint problem solving activity each object and each relation introduced, act as reactive agents. The behavior of each agent depends on whether it is on the active user's side or on the passive user's side at a specific point in time. If it is on the active user's side it monitors user events that are related to the particular object (move, change of properties, delete, etc.), and sends these events to the equivalent agent on the passive user's side. This is achieved through the *Mediators*, shown in figure 4. The size of these change-of-state messages is variable and depends on the kind of actions of the active user. However in most cases it remains very small, permitting good run-time performance. When the Mediator of the passive user's side receives the message, it decodes it and informs the equivalent agent who acts accordingly.

This necessitates that the objects present in the Activity Spaces of two collaborating partners are identical. However, as discussed in section 1.2, there is a possibility that two users are in possession of different primitive library objects, as a result of the open architecture of the environment. So there can be a case when the active user A adds an object into the shared activity space, which does not exist in the library of user B. In this case it is necessary to update the library of user B at run time with the missing object before proceeding any further. This is done transparently from the users as follows: When user A inserts the new object O_i in the Activity Space, Mediator A informs Mediator B about the addition of the new object, sending the appropriate message with the object's unique ID (shown as GUID in fig.4). Mediator B searches the local Entity Library for O_i If this object does not exist on host B then Mediator B asks A to send a copy of object O_i before proceeding any further. Mediator A sends the object, and waits. During this activity the user actions in the shared Activity Space are suspended and a message is displayed that the peer library is updated. After the sending is complete Mediator B informs Mediator A that it has received the object and the activity can proceed. The object multimedia attachments can be send either directly as shown in figure 4 or through the server if the size of the multimedia files are too large and can disrupt activity for both partners for too long. In the latter case

the message is send to the Community Server with the ID of the object, the server sends the object to the user. If the object does not exist in the server, it is downloaded, transparent to the two users from the library of user A.

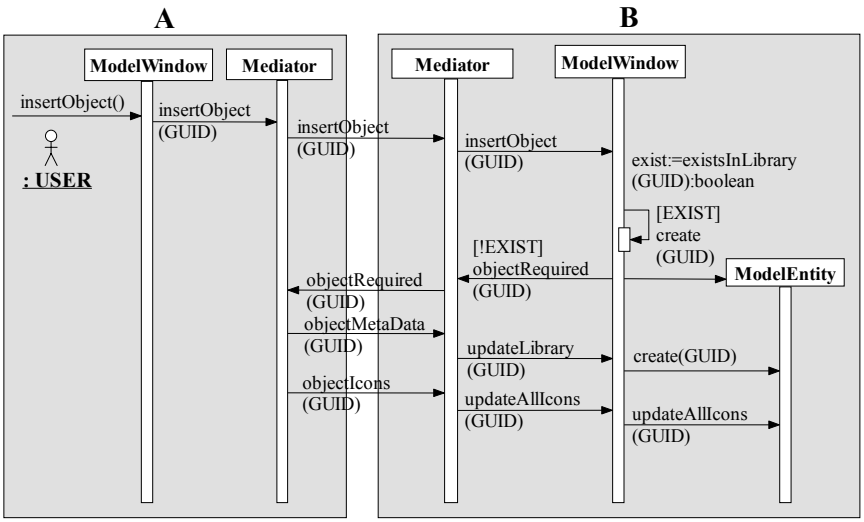

Fig. 4. Synchronization of the peer workstations in direct peer-to-peer interaction.

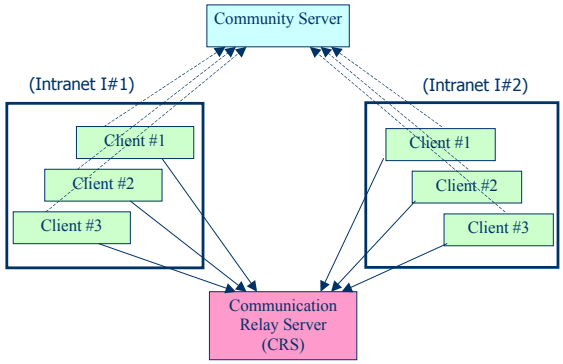

Fig. 5. Use of Communication Relay Server , Case A: collaboration across intranets

2.2 Communication through Proxy Servers

An alternative communication scheme is described in this section, which has been designed in order to overcome common problems in p2p protocols. These are the restrictions imposed by intranet proxy servers who do not allow point-to-point connections to not-trusted sites, while dynamic allocation of IP addresses creates difficulties in establishing reliable connection across Intranet boarders. A solution proposed to

this problem has been the definition of a trusted Communication Relay Server (CRS), residing in a host with public IP address. The role of this server is to relay the exchanged control messages to collaborating partners. This component of the MS architecture has been used effectively overcoming the above problems, permitting control message tunneling, traffic coordination, improving client security, since the communication is done only towards the trusted CRS node.

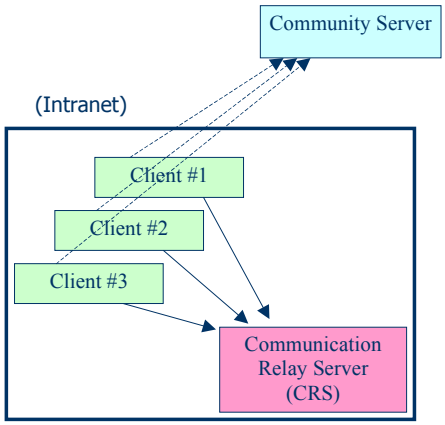

Fig. 6. Use of Communication Relay Server, Case B: collaboration within intranet

While the introduction of the CRS component solved these problems, in effect lead to an implementation of the p2p protocol through a client-server mode which defeats some of the advantages of the p2p approach, presented in section 2.1. For instance the existence of a central CRS server creates a bottleneck in communication and does not scale up. A more flexible approach to this problem, that has been lately used in MS, has been to let the final user decide on the CRS to use for collaboration. In effect in every installation of MS a copy of the CRS was included, so any host running MS software can become a relay server. A default relay server resides in the Project Community server (www.modellingspace.net), however if a user decides to start a collaboration session using his/her own host as relay server, this can be done by setting up the appropriate parameter in the MS environment. This is the case when the collaborating partners are located in a local area network, so that it is more effective to communicate using one of the local hosts as a relay server, as shown in case of figure 6. Finally the possibility of overcoming completely the Community Server and use just a local Communication Relay Server for synchronous collaboration is also allowed by this flexible architecture. This is the case of a group of users in a local area network with no connection to the outside world, who wish to collaborate using the p2p protocol. In the latter case, however some of the services of the Community Server are missing, i.e., the history of group collaboration cannot be retrieved, while presence info about group members is not available.

3 Coordination of Collaborative Activity

The coordination of partners' activity in the shared activity space is a very important aspect of the architecture. In general, the coordination mechanism of the activity in the shared workspace can take many forms, see [6] for a survey and a discussion of alternative approaches. Some of these approaches impose no particular control, i.e. any member has his/her own pointing device and can manipulate objects in the activity space or write on the whiteboard. This is claimed that may result in chaos with participants ending up in writing one on top of the other and cancelling each other's actions. Other approaches propose floor control mechanisms, involving the existence of a coordinator, various floor control protocols, like round-robin etc, or protocols of explicit request/ concession of the floor with time constraints. For instance, inactivity of the floor owner for more than a certain time can release the floor.

Two alternatives have been provided in relation to coordination mechanisms for ModellingSpace design. The first mechanism involves a token, the Action Enabling Key, which is owned by one of the participants at any given time. This key owner imposes a lock on the shared activity space. The owner of this token can act in the shared workspace, while the other participants just observe this activity. This mechanism is supported by *key request, key accept, key reject* functions. These coordination control messages are shown as connection (a) in fig.1. The effectiveness of this approach has been studied in various experiments, see [7, 8].

An alternative that has been also implemented, proposed especially for small groups of partners, involves lack of such floor control mechanism. The partners can manipulate parts of the model at any time during problem solving. For reasons related to distributed data consistency, only a temporary locking mechanism of objects selected by one partner is imposed during an operation, as shown in fig.7. The coordination of activities is left to the partners to decide in this case. So, the activity of a partner cannot be inhibited and no conflicts can occur over key possession. Nevertheless, in this case, implicit social protocols of organization need to be established by the users themselves, as discussed in [3], in order to facilitate coordinated group activity, since explicit coordination is not imposed.

Early experiments with the explicit floor control mechanism have indicated that it may improve reasoning about action, as partners need to reason and negotiate during key requests. In the experiment reported in [3] the effect of this mechanism on problem solving was studied, by comparing the performance of two groups of users, one of which used this mechanism while the other used no explicit floor control. A side-effect of the *no-floor control* case is observed when two users attempt at the same time to handle the same object. In this case the final state of the specific object dependents on the order of release of the lock on the object by the partners involved, as shown in fig.7.

3.1 Direct Communication

In the work surface, a text dialogue tool has been integrated, which is based on an instant messaging protocol, using the same point-to-point connection and protocol of the shared activity space. Through this, text messages are exchanged during collaborative problem solving, as shown in fig. 8.

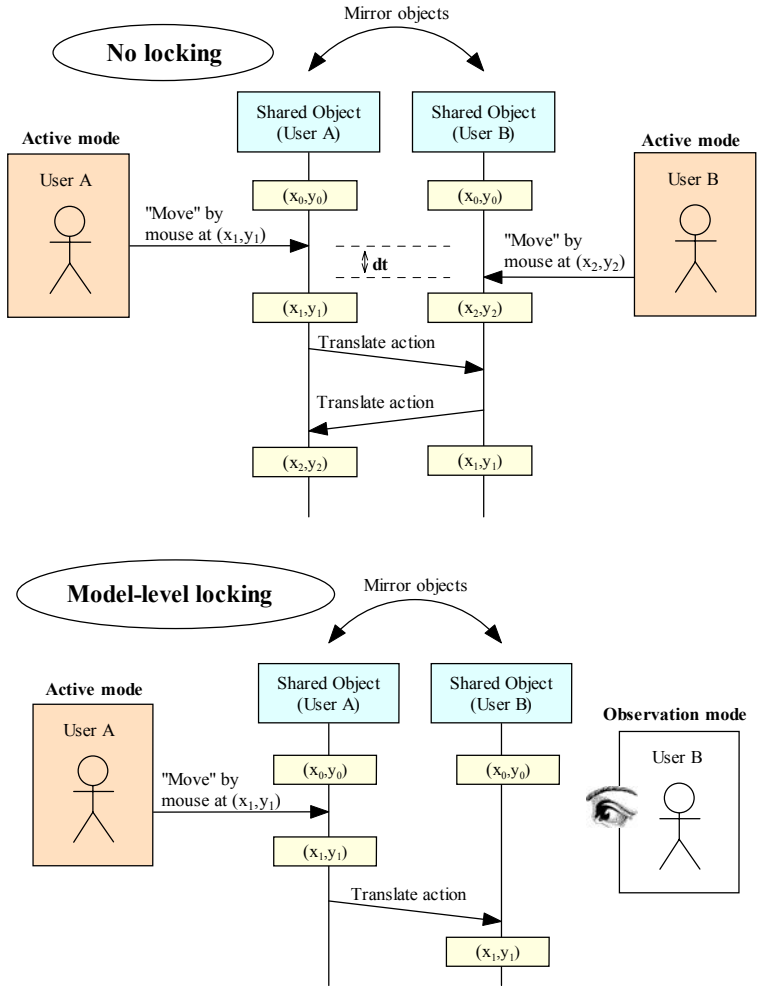

Fig. 7. Locking of objects in the shared space.

This chat tool, which is activated from the collaboration panel, is equipped with dialogue openers, i.e. phrases like “I agree with…”, “I object to…”, “ I think that…”, which can be used to open a chat message, as shown in fig.8. This way the user can select the opening phrase of the message and thus classify indirectly the speech act. There is a lot of controversy associated with structured dialogue mechanisms. Some researchers believe that they interfere with interaction and should be avoided, while others believe that they support development of meta-cognitive skills and in addition they facilitate analysis of communication and collaboration [13].

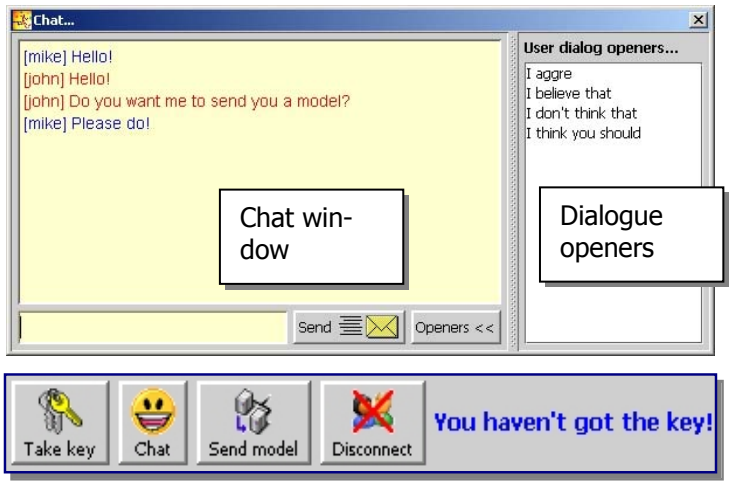

Fig. 8. Chat window and collaboration panel

Other means for exchange of text messages are the sticky notes (text containers positioned in the work space). These are treated, in terms of the architecture, as special kind of entities, with internal properties: *owner, time of creation, text_content*. Through the sticky notes, gestures can be simulated, since a sticky note inserted in the work surface, can be related to an object in this space and through this a comment by one of the partners can have a permanent effect.

4 Conclusions

In this paper we discussed a peer-to-peer architecture that permits real-time collaborative modelling at a distance. The approach involves exchange of just control messages for maintenance of effective WYSIWIS (what you see is what I see) of the shared workspace, as well as text chat messages for direct communication and coordination control messages. These messages are at the most a few bytes long and therefore can be exchanged without disruption of interaction even under low bandwidth peer-to-peer connections. The effectiveness of this approach has been proven through a number of case studies in authentic collaborative problem solving settings, reported in [3, 9, 12], in which alternative cooperation schemes have been implemented.

Two communication schemes have been implemented, one involving direct peer-to-peer communication and a second one through a relay server (CRS). The architecture can accommodate multi-partner collaboration due to the low-bandwidth requirements for both the p2p and CRS versions.

The proposed architecture is characterized by a great degree of flexibility, as it permits use of various coordination schemes and levels of locking of objects in the shared activity space, while the proposed communication relay server can overcome security problems and restrictions imposed in modern intranets.

The solutions discussed in this paper have applicability to a wide range of p2p applications, which can be used for effective collaboration and sharing of resources in communities of various sizes and characters.

Acknowledgement

The reported work has been partly funded by the *ModelsCreator* project in the frame of the Pinelopi Program of the Greek Ministry of Education and the IST-School of Tomorrow Project IST-2000-25385 "*ModellingSpace*" of the European Commission.

References

1. Avouris N.M., Dimitracopoulou A., Komis V., On Analysis of Collaborative Problem Solving: an Object-oriented Approach, *Journal of Human Behavior*, 19(2):147-167 (2003)
2. Avouris N., Margaritis M., Komis V., Saez A., Melendez R., ModelingSpace: Interaction Design and Architecture of a collaborative modelling environment, *Proceedings 6th Conference Computer Based Learning in Science (CBLIS),* Nicosia, Cyprus. (2003) 993-1004
3. Avouris N., Margaritis M., Komis V., Real-Time Collaborative Problem Solving: a Study on Alternative Coordination Mechanisms, *Proceedings IEEE International Conference on Advanced Learning Technologies (ICALT)*, Athens (2003) 86-90
4. Constantini F., Toinard C., Collaborative Learning with the Distributed Building Site Metaphore, *IEEE Multimedia*, July-Sept. (2001) 21-29
5. Dimitracopoulou A., Komis V., Design Principles for an Open and Wide Modellingspace of Modelling, Collaboration and Learning, *Proceedings 6th Conference Computer Based Learning in Science (CBLIS),* Nicosia, Cyprus (2003) 1005-1016
6. Dix A., Finlay J., Abowd G, Beale R., *Human-Computer Interaction*, 2nd Edition, Prentice Hall (1998)
7. Fidas C., Komis V., Avouris N.M. Design of Collaboration-support Tools for Group Problem Solving, *Proceedings PC HCI 2001 Conference*, Patras, Greece (2001) 263-268
8. Komis V., Avouris N., Fidas C., Computer-supported Collaborative Concept Mapping: Study of Synchronous Peer Interaction, *Education and Information Technologies*, 7(2):169-188. (2002)
9. Komis V., Avouris N., Fidas C., A Study on Heterogeneity During Real-time Collaborative Problem Solving, *Proceedings CSCL Conference*, Kluwer, (2003) 411-420
10. Lethin R., Technical and Social Components of Peer to Peer Computing, *Communications of the ACM*, Special issue on P2P Computing, 46(2):30-32, (2003)
11. Lopez P., A. Skarmeta, ANTS Framework for Cooperative Work Environments, *IEEE Computer*, 36(3):56-62, (2003)
12. Margaritis M., Avouris N., Komis V., Dimitracopoulou A., (2003), Real-time Collaborative Modelling over Low-bandwidth Networks, Proceedings CSCL Conference, Bergen, (2003) 138-140
13. Soller A.L. Supporting Social Interaction in an Intelligent Collaborative Learning System. *International Journal of Artificial Intelligence in Education*, 12(1):40-62 (2001)
14. Suthers D.D., Architectures for Computer Supported Collaborative Learning, *Proceedings IEEE International Conference on Advanced Learning Technologies (ICALT),* Madison, WI (2001)