An Ad-Hoc Wireless Network Architecture for Face-to-Face Mobile Collaborative Applications

Gustavo Zurita[1] and Miguel Nussbaum[2]

[1] Universidad de Chile, Departamento de Sistemas de Información y Auditoria, Escuela de Economía y Negocios, Diagonal Paraguay 257, Santiago 9227, Chile
gnzurita@facea.uchile.cl
[2] Pontificia Universidad Católica de Chile, Departamento de Ciencia de la Computación, Escuela de Ingeniería, Casilla 306, Vicuña Mackena 4860, Santiago 22, Chile
mn@ing.puc.cl

Abstract. An architecture for building an ad-hoc wireless network is presented in which various face-to-face, peer-to-peer collaborative applications function simultaneously and the interconnections between group members are highly dynamic and self-organizing. To illustrate how the architecture implements communication, examples of client-server and point-to-point communication are given. An interconnection architecture of a Mobile Computer Supported Collaborative Learning (MCSCL) environment is analyzed in detail. Its communication protocols are showed with sequence diagrams. The paper concludes with an evaluation of the architecture's performance.

1 Introduction

An ad-hoc network [8] is a transitory or permanent association of nodes or mobile devices that do not depend on any fixed support infrastructure to establish intercommunication among them [1]. Connection and disconnection is controlled by the distance among nodes and the face-to-face requirements of the implemented peer-to-peer (p2p) application, which may be educational [3], commercial [7], [11] or collaborative [13].

According to [6], a mobile p2p system inherits many of the features of ad-hoc networks. Specifically, it will be (a) self-organizing: as a side effect of the movement of devices within a limited physical space, the topology of a mobile p2p system constantly adjusts itself, discovering new communication links and managing various ad-hoc sub-networks as required by the application; (b) fully decentralized: each peer is equally important and no central node exists; and (c) highly dynamic: communication endpoints can move and change frequently and independently of one another.

The mobile nodes in these systems can function in any location and change their configuration and/or membership in various sub-networks within a single network to adapt to the face-to-face social interactions that users engage in and that the network must support. The disadvantages of wireless data transmission systems are that they have relatively less bandwidth, more latency, less connectivity stability, and less predictable availability [2]. Additional constraints are a) decentralized control, to have synchronization even when a node fails, b) fault tolerant, when a node fails the other have to be operational, and c) dynamic reconfiguration, sub-networks are formed on demand.

This study presents an architecture for building an ad-hoc wireless network in which various face-to-face, peer-to-peer collaborative applications function simultaneously and the interconnections between group members are highly dynamic and self-organizing.

2 MCSCL Communication Support (MCSCL-CS)

A face-to-face Mobile CSCL (or MCSCL) environment enables several small groups (3 to 5 members) to work collaboratively while moving around freely with handhelds [12], [13]. This capability facilitates flexibility in social interactions and easy management of group composition.

When an MCSCL environment is used in a setting such as a school classroom, the ad-hoc network must not only interconnect all of the collaborative workgroups, but must also simultaneously maintain various sub-networks for each of the 3-to-5-member groups, which function in different collaborative activities that at any given moment are at varying stages of completion.

The proposed ad-hoc network architecture is intended for use with any MCSCL-type p2p application, and enables the interchange of group members in real time. The scenario described here is an environment in which each student in the classroom has a handheld which is used as a support tool for performing collaborative activities together with fellow group members. As well, this environment allows dynamic reconfiguration of the groups.

2.1 Specification of the Proposed Architecture

The specifications of the proposed architecture are:

- Mobility. The application must function anywhere.
- Ad-hoc Network. The network does not depend on any infrastructure beyond that formed by the handhelds themselves. Within a single ad-hoc network, various other sub-networks may be created as required for establishing interconnections between members of the collaborative groups.
- Social and Technological Network. Users can communicate not only over the technological (ad-hoc) network, but also through the “social” network, that is, face-to-face communication between peer groups.
- Configurable. Applications may need to configure different types of interconnection between nodes. In other words, they may need to establish various ad-hoc sub-networks simultaneously as well as configure a variety of intercommunication topologies between nodes, such as client-server, point-to-point, one-to-many or many-to-one.
- Dynamic reconfiguration. The environment must permit reconfiguration of sub-networks in real time.
- Extensible. This feature is necessary to enable the addition of applications not contemplated when the architecture was originally designed.

- Efficient. The architecture's level of performance must be sufficient for the application, meaning that communication times will be undetectable by the user.
- Manageable. Within the same ad-hoc network, one of the handhelds (the teacher's) must be able to reconfigure and manage in real time.

2.2 General Architecture

MCSCL Communication Support (hereafter MCSCL-CS) is derived from DACIA [10] and includes certain aspects of its group communication design. Thus, collaborative groups are defined as closed groups because they develop activities independently of those of the other groups and so do not need to be aware of the latter's external messages. Communication has been modeled as a hierarchical group, which does not limit the different forms of communication a particular activity may be required to establish so that group members can carry out the roles that activity defines. As for membership control, this is handled inside the group. The management of the collaborative groups has therefore only one point of access, making possible the Dynamic Reconfiguration of Groups (DRG). Maintaining consistency of the messages exchanged by different groups hosts is accomplished through a consistent ordering, given a hierarchical structure that facilitates message management, thus rendering global ordering unnecessary. Finally, as regards the scalability of the system, it must be ensured that the system works independently of the number of groups created and the number of group members. Since the scalability of distributed systems can be negatively affected by design decisions that tend to centralize them, MCSCL-CS was conceived for use in classrooms, whose numbers of system users are known and finite, thereby reducing the adverse impact of the groups' hierarchical structure.

MCSCL-CS is made up of a variable number of components referred to as Comp-CS that provide the necessary functionality for performing an MCSCL activity, which requires diverse structures and models of intercommunication (Fig. 1). Each Comp-CS combines the *Display* of the user interface and the logic of the collaborative activity (*Application's logic*). In Fig. 1, each system application is composed of *n* Comp-CS's, one for each node used by a specific peer.

Communication among Comp-CS's is carried out at two levels. Among components residing in the same host it is executed using an adaptation of the design pattern Events Notifier [5]. Thus, a component is subscribed to the events that other components publish, connecting and disconnecting the components of the p2p system. Communication among components, which is necessary for the collaborative performance of the activity, occurs via the exchange of messages through ports. The components can dynamically request a number of variable ports in real time. All the ports activated in a host are administered by an *Operator* that resides in each client application. Each application's Comp-CS has *n* associated ports to communicate with *n* remote components (Fig. 1).

Communication among hosts within the collaborative group and the connections between pairs of components that reside in different hosts is administered by a *Coordinator* (which may reside in any handheld). In similar fashion to the telephone system, the *Coordinator* is responsible for the wireless connections among the members of a given group of users (sub-network). That is, the *Operator* of a given node asks the *Coordinator* to establish a connection with another node *Operator* (Fig. 1).

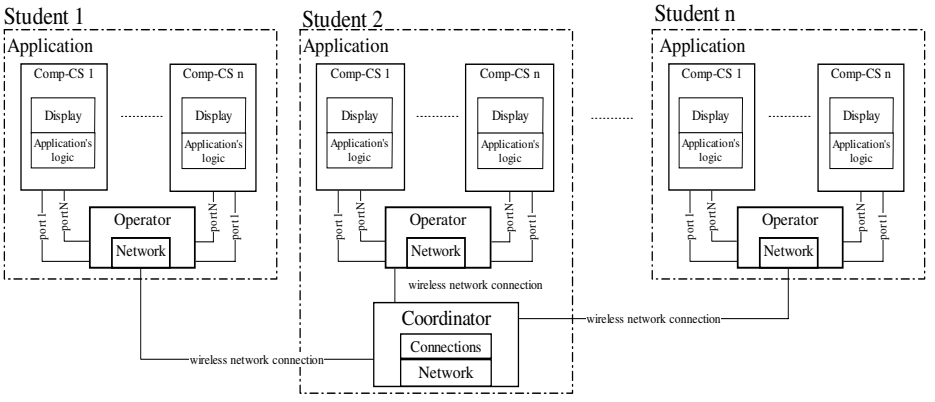

Fig. 1. Generic architecture of a system based on MCSCL-CS

2.3 Architecture Examples: Client-Server and Point-to-Point Communication

The architecture described above is general enough to be able to implement any type of communication. In what follows, two basic types of communication are presented: client-server and point-to-point.

Scenario A of Fig. 2 is an example of client-server communication with three clients, one of them acting as the server and each having its own application logic. The server application has as many ports (named PStudent 1, PStudent 2 and PStudent 3) as there are clients in the group (including its own client), and provides communication service to the other clients. To communicate with the server (PortStudent 3) the clients only need one port. If, for example, Student 1 wants to communicate through its Comp-CS Client with Student 2, the requirement is sent through PortStudent 3 ports that have been established by the Server Comp-CS (resident in Student 3).

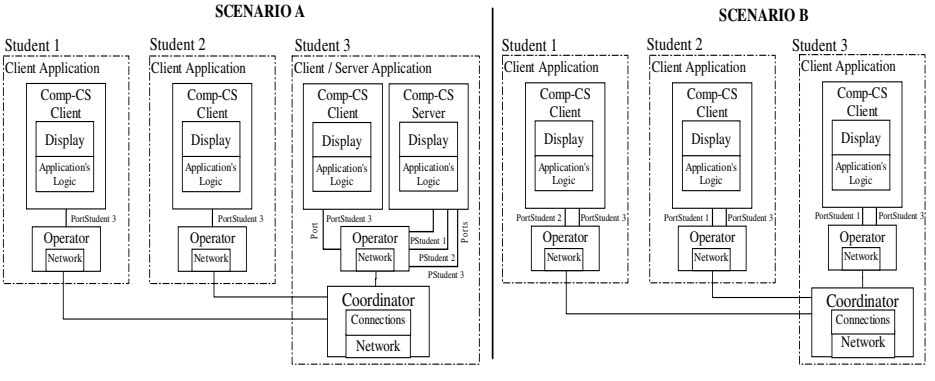

Fig. 2. Architectures for client-server (A) and point-to-point communication (B)

In scenario B of Fig. 2, an example of point-to-point communication is shown. Here, each Comp-CS component will need as many ports as there are partners in the group (excluding itself) in order to communicate with the others, which in this case are two. So, for example, when student 1 wants to communicate with student 2 (using his/her Client Comp-CS) s/he does so through student 2's port (PortStudent 2).

3 Design of an MCSCL Environment

In an MCSCL environment there are two types of actors, teachers and students. The teacher's handheld (MCSCL-Tch) configures and manages the p2p collaborative group activities. The students' handhelds (MCSCL-Stu) run the collaborative educative activities, communicating through the ad-hoc network [12], [13] and [14]. Interconnections and communication must be established with the students' applications so that they can form collaborative groups, start the activity and, when necessary, modify the group configurations.

The specific requirements of the MCSCL-Tch application are: a) management and selection of students' handhelds during the MCSCL activity; b) management and configuration of the groups that develop the activity; and c) management and configuration of the specific MCSCL activity. The requirements for the MCSCL-Stu application are: a) student handhelds assignment information; b) student group assignment information; and c) Rules and roles for the MCSCL activity.

3.1 Architecture of the System

Fig. 3 shows an MCSCL-Tch teacher's application and three groups with a total of nine MCSCL-Stu student's applications, the latter represented by solid-line circles each with an *Operator* component. The *Operator* of the MCSCL-Tch application

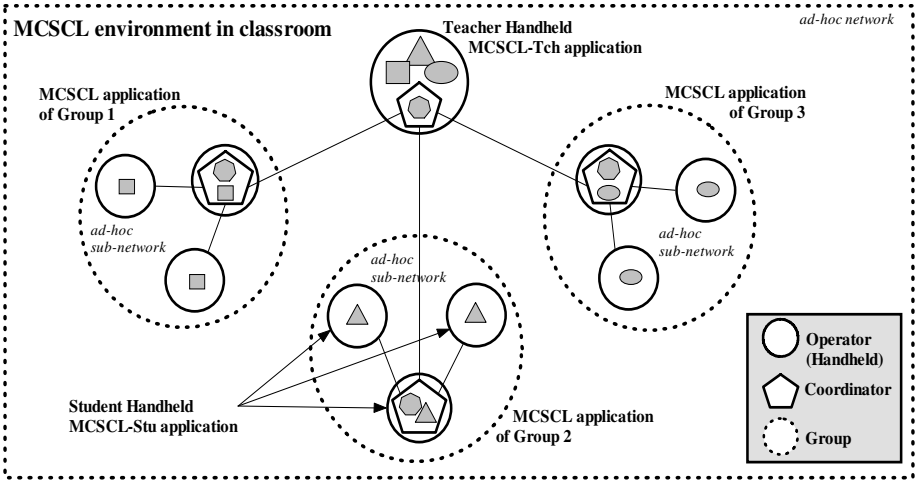

Fig. 3. Characteristics of an MCSCL environment

creates the three classroom groups containing the students, and through the group *Coordinator* (see hexagon inside pentagon) invites the *Operators* of each MCSCL-Stu application to form a part of the specific groups. Once the students are identified, the MCSCL-Tch application configures and manages three components from the nine MCSCL-Stu applications (in the ad-hoc network), that contain a *Coordinator* (the handhelds with a pentagon) which in turn configure and manage two other MCSCL-Stu applications, to form three ad-hoc sub-networks: Group 1, identified by a square; Group 2, by a triangle; and Group 3, by an ellipse.

The lines without arrowheads joining all the circles (handhelds) constitute the ad-hoc network's interconnections. Each of the dotted circles corresponds to an ad-hoc sub-network, whereas the dotted square represents the ad-hoc network. In each sub-network the same or some other application may be executed. The MCSCL-Tch application and one of the MCSCL-Stu applications in each group has a *Coordinator* component that is responsible for establishing communication between MCSCL-Tch and each group formed in order to coordinate activity development in the latter.

3.2 MCSCL-Tch Application

Fig. 4 shows the MCSCL-Tch application. The *Connection Manager* component establishes the links with the classroom group *Coordinator* and with each *Coordinator* of the students' collaborative groups (in one of the group's handhelds). When the application is executed, each component is created and performed, as are the application *Operator* and the group *Coordinator* in the classroom. Finally, the *Operator* is subscribed to this group, as explained in section 2.2

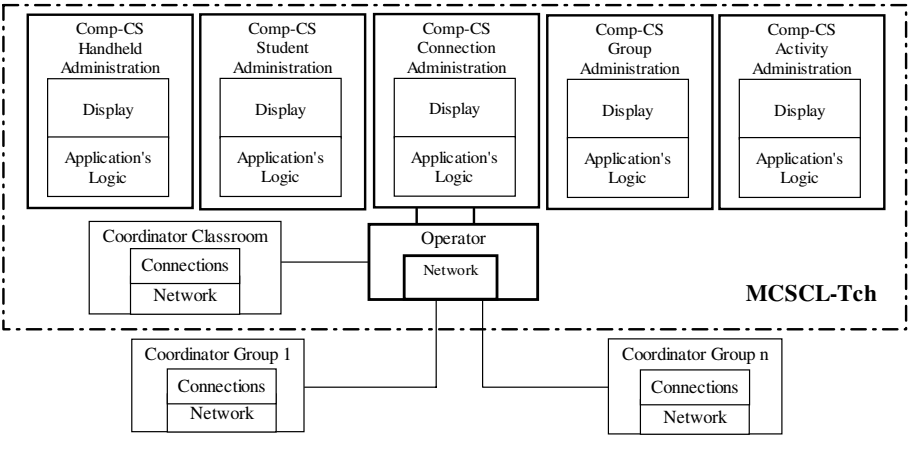

Fig. 4. Scheme of the MCSCL-Tch application architecture

3.3 Group Formation

The system's networking is exclusively p2p. This means that all users will have the same program running on their handhelds and there is no central service. The program recognizes the presence of other participants and establishes a secure communication

with them in order to transfer data for synchronizing the applications. This is done via multicasting, peer discovery and synchronization via point-to-point data communication. Group formation is a basic function of the p2p MCSCL application. It is composed of three clearly distinguished stages:

- The *Operator* of the MCSCL-Tch sends connection invitations to the *Coordinator* so that connections are established with the *Operators* of the MCSCL-Stu applications residing in the students' handhelds.
- The *Operator* of the MCSCL-Tch application requests its *Coordinator* to create a new *Coordinator* among the MCSCL-Stu student application *Operators* that were previously connected.
- A group must be formed with the *Coordinator* just created. This *Coordinator* receives the connections from the *Operators* of the MCSCL-Stu application of each student that belongs to the group. The messages indicating that those *Operators* are to connect again to the new *Coordinator* are sent to the *Operator* of the MCSCL-Tch application through the teacher *Coordinator*.

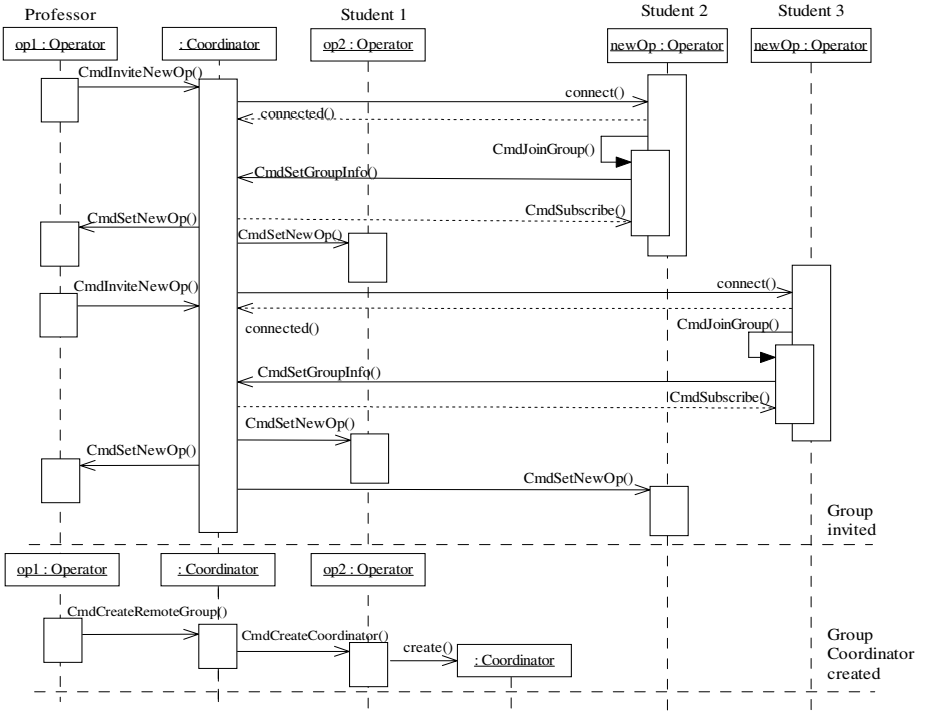

Fig. 5. Sequence diagram for connecting three students

Fig. 5 is a sequence diagram illustrating how the teacher connects three students and asks each one to create a group *Coordinator*. The *op1:Operator* of the MCSCL-Tch application sends a *CmdInviteNewOp()* command connection invitation to its *Coordinator* to establish a connection with a new *Operator* of the MCSCL-Stu

student application (*NewOp:Operator*). As can be seen in Fig. 5, *op2:Operator* (of Student 1) is already connected to the teacher's *Coordinator*, and two new *Operators* corresponding to students 2 and 3 are in the process of being connected. Once all the students' *Operators* have been connected (Fig. 5, the teacher's *op1:Operator* asks its *Coordinator* by means of a *CmdCreateRemoteGroup()* command to create a *Coordinator* among of the three *Operators* that were connected. The decision as to who will be the *Operator* that creates the new *Coordinator* is made by the MCSCL-Tch application; in Fig. 5, *op2:Operator* of Student 1 is chosen. The teacher's *Coordinator* asks *op2:Operator* through the *CmdCreateCoordinator()* command. Each group formed has a group *Coordinator*, and the MCSCL-Stu applications of each student who has joined a group have an *Operator* connected to that group *Coordinator* and to the *Coordinator* defined by the teacher.

Fig. 6 is the sequence diagram showing how the teacher's *Operator* and *Coordinator* and the students' *Coordinator* and *Operators* form a three-member collaborative group. The *Operators* created in Fig. 6 are now called *op3:Operator* and *op4:Operator*, corresponding to students 2 and 3. The communication is established through the teacher's *Coordinator* and the group *Coordinator* of the three connected students.

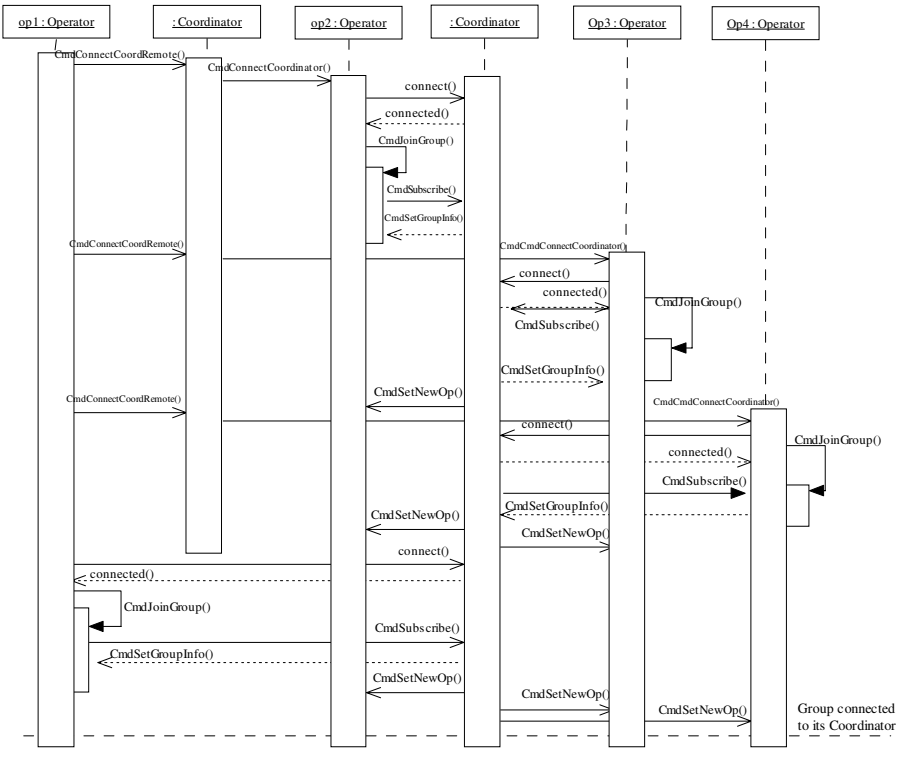

Fig. 6. Sequence diagram for forming a group

To form a group of students, the teacher must send the applications to each member of the group. For example, in order to add student 1 to the group, the teacher's *Operator* asks its *Coordinator* for that application through a *CmdConnectCoordRemote()* command. The teacher's *Coordinator* then asks student 1's *op2:Operator* by means of a *CmdConnectCoordinator()* command to include Student 1 in the collaborative work. The Student 1 *Operator* asks the group *Coordinator* (in this case, contained within itself) for a connection, and the *Coordinator* then joins *CmdJoinGroup*() and subscribes *CmdSubscribe()* Student 1 to the group.

In the scenario illustrated in Figs. 5 and 6, the teacher connects and forms a group of three members only. The procedures followed to connect more students and create new collaborative work groups are similar.

3.4 Starting and Sending Messages

Once the group is formed, a protocol based on the logic of the collaborative activity must be established for communicating among the group members. The sequence diagram in Fig. 7 shows the establishment of communication under a client-server protocol. This protocol, as requested of the collaborative group *Coordinator* by the teacher's *Operator*, must start (*CmdStartComp()*) and update (*CmdUpdateCompInfo()*) with the *ClientMCSCL* information component of the MCSCL application for each member of the group, including the teacher's *Operator*. A new *ServerMCSCL* component of the MCSCL-Stu application must then be created. This way, neither the *ClientMCSCL* component nor the *ServerMCSCL* component knows that they reside in the same place, which allows any MCSCL-Stu application to start that component. Finally, the group *Coordinator* requests and creates communication ports among all the *ClientMCSCL* and *ServerMCSCL* components.

Once communication between the ports is established, each member of the collaborative group is ready to send and/or receive information.

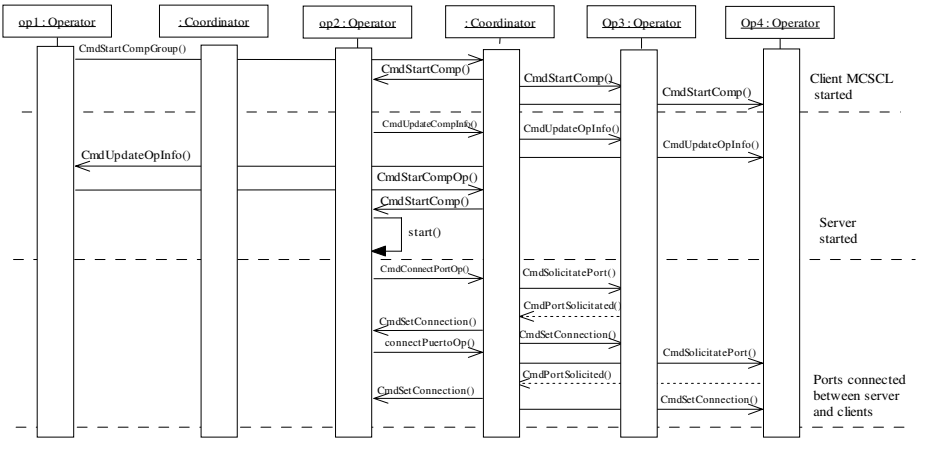

Fig. 7. Sequence diagram for communication between ports under server protocol

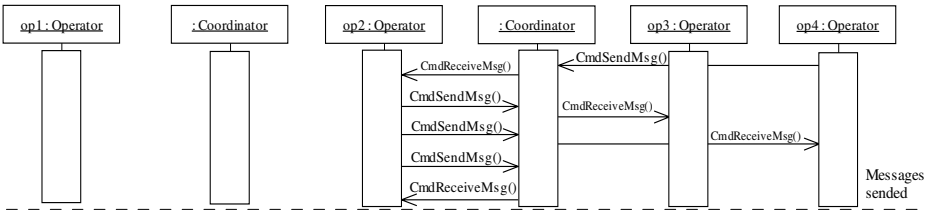

Fig. 8. Sequence diagram for the delivery of messages among the members of a collaborative group

The sequence diagram in Fig. 8 is an example of how the Student *op4:Operator* sends a message (*CmdSendMsg()* command) to the *op2:Operator*, who then forwards the same message (or a different one) to the other two students *op3:Operator* and *op4:Operator*. Note that the request to send a message is delivered to the group *Coordinator*, who redirects it to the *Operator* that needs the information (*CmdReceiveMsg()* command). The MCSCL-CS architecture is such that this procedure is independent of the logic and requirements of the MCSCL collaborative activity.

Any other communication protocol that an MCSCL activity might require can be designed based on the described functionality of MCSCL-CS, which demonstrates its flexibility, extensibility and adaptability.

3.5 Dynamic Group Reconfiguration (DRG)

For a DRG to be carried out, there must be at least one group to be reconfigured. Fig. 9 shows the sequence diagram for dismantling (disarming) the group formed in Fig. 6. The student group *Coordinator* is eliminated by the *delete()* command. After this operation, the students remain connected to the group defined by the teacher (Fig. 5).

To execute a DRG, the teacher chooses the new members of the collaborative groups who are to work on a given activity through the MCSCL-Tch application. Once all the members of the groups have been selected, the teacher's *Operator* (*op1:Operator* in the case of Fig. 5) must create the new *Coordinators* for each collaborative group again. In the case shown in Fig. 5 the *Coordinators* should be started again since the students are already connected to the teacher's *Coordinator*, the only remaining task then being to create a new group *Coordinator* and connect the students' *Operators* to it. Recall that Fig. 5 shows the case of one teacher and only three students; if there were more students connected, the DRG would choose other components of the MCSCL-CS as group *Coordinators* in the students' handhelds.

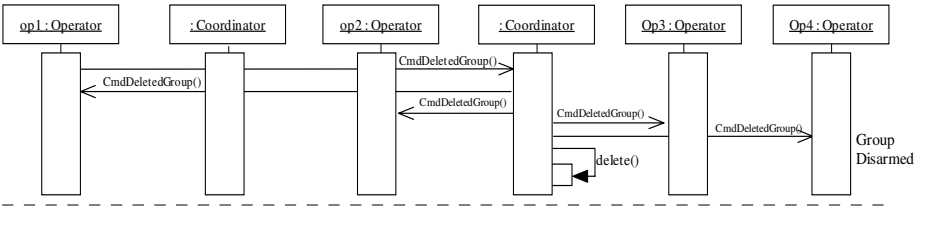

Fig. 9. Sequence diagram to dismantle a three-student group already formed

Finally, the MCSCL groups must be formed with their respective *Coordinators*. An example of this may be seen by referring back to Fig. 6, which shows the sequence diagram for creating a three-student collaborative group.

4 MCSCL-CS Performance Evaluation

The architecture proposed here has been employed with the MCSCL applications discussed in [12], [13] and [14], and implemented on the eMbedded Visual Basic (eVB) Runtime for Windows Mobile-based Pocket PC 2002 platform. The applications are executed over a wireless p2p Wi-Fi network and TCP/IP on Compaq iPAQ handhelds. For the permanent storage of configurations and results of the groups, Microsoft SQL Server CE 2.0 was used. TCP Sockets from the WinSock 3.0 eVB library provided the necessary elements to create, eliminate, connect and manage the socket connections established among the handhelds [4]. Since the eVB development environment is only a subgroup of Visual Basic, it cannot support dynamic object creation, i.e., at runtime. To solve this problem for the applications developed here, the socket-time creation was simulated based on a defined number of static objects created in implementation time. In this way, the objects needed at runtime were handled as an array of objects.

To measure the architecture's performance, the teacher's handheld (MCSCL-Tch application) interconnection delay before formation of the defined groups (MCSCL-Stu applications) was timed. MCSCL-Tch will form an ad-hoc network with MCSCL-Stu, which in turn will form ad-hoc sub-networks for each group. The group formation evaluation was conducted on a teacher's handheld for 1, 2, 9, 12 and 15 groups, each group consisting of 3 handhelds, thus forming ad-hoc networks of up to 45 students (the typical Chilean classroom size). For each of the five different quantities of groups, time performance was evaluated for delivery of 3 different information package sizes: 128 bytes, 256 bytes 512 bytes (MCSCL applications transfer small volumes of information).

As shown in Fig. 10 the time taken for all the groups to form their ad-hoc sub-networks depends on how many groups there are. According to the protocol, this time should be independent of their number, but the more groups there are the more acknowledgements must be sent to their handhelds, all of whom share the same bandwidth. Furthermore, time is needed to form the groups' ad-hoc network (controlled by MCSCL-Tch) once they have all created their ad-hoc sub-networks.

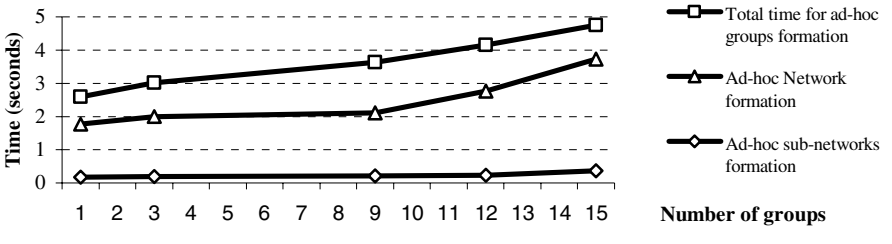

Fig. 10. Formation time of ad-hoc network and sub-networks

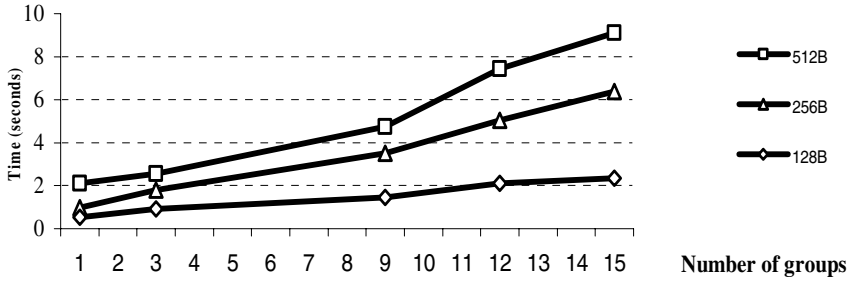

Fig. 11. Time taken for sending packages of various sizes (512B, 256B, 128B) to different number of groups (1, 3, 9, 12 and 15)

The chronometers measured only the time the teacher's handheld took to form the groups plus the time taken to send the information through the communication ports to all members of the group. In both cases, the MCSCL-Tch application receives acknowledgments of group formation (ad-hoc sub-networks) and of sent information for each group.

Fig. 11, shows the time taken for sending packages of various sizes to different numbers of groups. Sending time increases as the number of groups to which information is sent thought the ports increases. This occurs because each sub-network replicates the information, thereby overloading the wireless interconnection. In Fig. 10 the results obtained with the different numbers of groups reveals how group formation times increase linearly as the quantity of groups to be formed increases. The chronometers measured only the time the teacher's handheld took to form the groups and the ad-hoc network. This does not mean that all the collaborative groups wait for that number of seconds before continuing with their activity; rather, the number refers to the time the MCSCL-Tch application takes to execute the last phase of the protocol for the last group serviced. For example, with 15 collaborative groups, at worst each one will have to wait an average of 9.102/15 seconds to receive 512 bytes of text information before being able to return to its activity. The total number of 9.102 is explained by the fact that MCSCL-Tch must wait until the last group acknowledgement of information received has arrived and the information has been replicated to the group's ad-hoc sub-network. Once the ad-hoc network and the ad-hoc sub-networks of each of the groups have been formed, the sending and/or receiving of information does not result in heavy loads on the environment because the communication ports between group members, and the ports between them and the teacher's handheld, have already been created.

5 Final Remarks

When an MCSCL environment is used in a setting such as a school classroom, the ad-hoc network must not only interconnect all of the collaborative workgroups, but must also maintain various sub-networks, which function in different collaborative activities at various stages of completion at any given moment. The proposed ad-hoc network architecture is intended to be used with any MCSCL-type p2p application,

and enables the interchange of group members in real time necessary to achieve high levels of student communication and motivation [9], [13], [14], necessary to achieve learning objectives.

Initially the MCSCL environment recognizes the presence of other participants and establishes a fault tolerant communication to transfer data for synchronizing the applications. This is done via multicasting, peer discovery and synchronization via point-to-point data communication.

Using the MCSCL-CS architecture that has been proposed here, the number of ad-hoc network nodes in a p2p collaborative system can vary up to 45 or more without causing network instability.

With MCSCL-CS, a teacher's handheld can (a) manage all other ad-hoc sub-networks, (b) configure the formation of new network nodes without having to reboot, (c) simultaneously maintain sub-networks working with different collaborative applications, (d) maintain sub-networks with 3, 5 or more nodes per collaborative work group, and (e) reboot and reconnect a collaborative work group when a group node crashes.

Once initiated by the teacher's handheld, each ad-hoc sub-network can function independently while always maintaining an open interconnection with the teacher in case rebooting or a change in the membership of a work group is necessary.

Acknowledgements

This work was supported by FONDECYT #1050601.

References

1. Bartram, L., Blackstock, M.: Designing Portable Collaborative Networks. Colligo Networks, ACM Queue 1(3) (2003) 41-49
2. Buszko, D., Lee, D., Helal, A.: Decentralized ad-hoc groupware API and framework for mobile collaboration. GROUP 2001 (2001) 5-14
3. Danesh, A., Inkpen, K.M., Lau, F., Shu, K., Booth, K.S.: Geney™: Designing a Collaborative Activity for the Palm™ Handheld Computer. Proceedings of the Conference on Human Factors in Computing Systems (CHI 2001) Seattle, USA (2001) 388-395
4. Grattan, N.: Pocket PC Handheld with Microsoft embedded Visual Basic. NJ: Prentice Hall PTR (2001)
5. Gupta, S., Hartkopf, J.M., Ramaswamy, S.: Event Notifier, a Pattern for Event Notification, Java Report, SIGS Publications 3(7) (1998) 19-36
6. Kortuem. G.: Proem: A Peer-to-Peer Computing Platform for Mobile Ad-hoc Networks. In Advanced Topic Workshop—Middleware for Mobile Computing, Heidelberg, Germany, Nov. 2001. Banavar, G. Editor. Retrieved on December 2005 from http://www.cs.rizona.edu/mmc/10%20Kortuem.pdf
7. Malloy, A., Varshney, U., Snow, A.: Supporting mobile commerce applications using dependable wireless networks. Mobile Networks and Applications archive 7(3) (2002) 225 - 234

8. Murphy, A.L., Roman, G.C, Varghese, G.: An Exercise in Formal Reasoning about Mobile Communications. Proceedings of the Ninth International Workshop on Software Specifications and Design, IEEE Computer Society Technical Council on Software Engineering, IEEE Computer Society Ise-Shima Japan (1998) 25-33
9. Pintrich, P.R., Schunk, D.H.: Motivation in education: Theory, research, and applications, Prentice Hall Merrill, Englewood Cliffs NJ (1996)
10. Radu, L.: Providing Flexibility in Distributed Applications Using a Mobile Component Framework. Ph.D. dissertation, University of Michigan, Electrical Engineering and Computer Science, Sep. 2000. Retrieved on December 2005 from http://www.eecs.mich.edu/~aprakash/papers/radu/dissertation_radu.pdf
11. Tarasewich, P.: Designing mobile commerce applications. Communications of the ACM. SPECIAL ISSUE: Mobile commerce opportunities and challenges, 46(12) (2003) 57 - 60
12. Zurita, G., Nussbaum, M.: A Constructivist Collaborative Learning Environment supported by Wireless interconnected handhelds. Journal of Computer Assisted Learning, 20(4) (2004) 235-243
13. Zurita, G., Nussbaum, M.: Computer Supported Collaborative Learning using Handheld Computers. Computer & Education, 42 (2004) 289-314
14. Zurita, G., Nussbaum, M. Sharples: Encouraging face-to-face collaborative learning through the use of handheld computers in the classroom. Human Computer Interaction with Mobile Devices and Services. Springer, Verlag Lecture Notes in Computer Science 2795 (2003) 193-208.