# On Distributed Systems and CSCL[1]

Leandro Navarro, Joan Manel Marquès, Felix Freitag
*Computer Architecture Department*
*Politechnic University of Catalonia (UPC)*
*{leandro,marques,felix}@ac.upc.es*

## Abstract

*The design of distributed systems for collaborative work or learning requires that both humans and machines be organized such that the potential and limitations of the human-machine interdependence is optimized. Current distributed architectures like Peer-to-Peer and Grids have not yet been optimized to support the needs of collaborative applications. In this paper, we investigate the requirements of collaborative applications for groups and discuss current Grid and P2P architectures with regards to this need. We suggest possible extensions described in a "function dispersion architecture". Finally, we present as case study our recent work on the design of a middleware to provide services for distributed collaborative applications that has been validated by simulation and prototyping.*

## 1. Introduction

The world of humans and machines which collaborate summing their capabilities, resources and work is complex and with a sensitive equilibrium state. The design of a system, in which people collaborate by means of software applications requires that all agents (both the persons and applications) are organized in order to benefit from the potential of technology and human collaboration, but also to respect its limitations.

Collaborative applications can provide to a group of persons a highly productive environment, achieving its goals efficiently. In the contrary, an application can also represent a highly ineffective tool, imagine situations of intensive collaboration which lead to the saturation of the machines and the network.

It is therefore important to design collaborative applications taking account how people collaborate, and taking account how the machines and the network work, on which these application run. Both the components (humans and machines) of this social-technical framework have limitations, which need to be considered in the design, in order find a balance in the capacities and to avoid disequilibrium.

The efficiency of collaborative applications will benefit if underlying concepts of both group interactions within humans and performance of distributed architectures are considered. Aspects from human interactions include for instance, how groups coordinate and where the participants are located. Aspects from distributed system architecture include the characteristics of the user interface, the machines used, network characteristics, and the price and availability of computational resources.

In the domain of cooperation for the development the term “appropriate technology” is coined, which basically suggests to design and build systems, which adapt to the available resources. This strategy fits to the current resources and avoids additional costs, taking into account the capacity of humans and machines. In our view, we share this goal in the sense of optimizing on the available resources the benefits of collaborative applications.

From a technical point of view, the trend of software design to high abstraction may hide important details of the infrastructure to the designer, which may lead to inappropriate systems for certain situations. This may concern general concepts of distribution, the presence of faults, and the scale of the system.

In our view, we believe that systems proposed for collaborative work have to take into account the characteristics of the interactions of such a complex distributed system formed by humans, computers and

[1] This work has been supported by the Spanish Ministry of Science and Technology under TIC2002-04258-C03-01.

networks, which are organized to perform a collaborative task.

The remainder of the paper is structured as follows: In section 2 we identify the requirements a system designed for collaborative work should provide. Section 3 provides a discussion of current distributed architectures with regards to using them for collaborative tasks. In section 4 we propose a "function dispersion" architecture which contains some of the elements identified for collaborative applications. Section 5 presents as case study the middleware LaCOLLA targeting to provide services for collaborative applications. In section 6 we conclude the paper.

## 2. Requirements for systems to support collaborative tasks

The collaboration and learning of humans by means of computers can lead to difficulties in several situations, starting with the interconnection when the participants of a collaborative tool belong to different organizations or departments, located in separate locations with less than ideal computers and networks. In such situations, the resources are managed by distinct authorities, which often target to facilitate the internal work, but hinder collaborative tasks over different domains, like for instance through firewalls. These "extreme" and complex situations offer an excellent opportunity for computer-based instructional systems, but the architecture of current systems fails.

There are several additional difficulties, which designers face when using current environments for collaborative tasks. In the following we indicate these difficulties as requirements to be solved by a system for building collaborative applications in settings where educational applications with traditional centralized or server-based architectures fail.

- **Internet Scale of the system**: The collaborative system can be formed by various distributed components and as such both its members and components can be in any place (function dispersion).
- **Universal and transparent access**: The participants can join the system at any place while the user interface should be independent of the location.
- **Decentralization**: None of the components should neither be exclusively in charge of coordination, neither have unique information. Centralized approaches often lead to simple, but more critical solutions, affecting the autonomy of the participants, the potential growth of the system and its fault tolerance.
- **Self-organization of the system**: It is desirable that the system is able to work without external intervention, to organize its components spontaneously, have fault tolerance and adapt to dynamics (i.e. mobility, disconnection).
- **Individual autonomy**: Each member of a group should have the facility to decide which resources are contributed (shared), when to connect and disconnect.
- **Self-sufficiency**: Each group should be able to work with limited resources, like for instance those contributed by the members of the group, or the group should be able to obtain additional resources externally.
- **Sharing**: Different participants or components should be able to use the same group information (i.e. events, objects, structures), both for access and for representation.
- **Security**: The system should guarantee the identity of the participants and provide access control to confidential shared information (data protection, authentication).
- **Resource availability**: The system should provide the resources such that a group is able to carry out a collaborative task. The resources include both the resources contributed by the group members (Peer-to-Peer) and external resources (Grid).

## 3. Current distributed architectures

In order to assure that the requirements for collaborative applications are met, the architecture of the system is critical. It is important to define the components of which the architecture consists and which distributed mechanisms or algorithms are used for the management and organization of the system.

### 3.1 Peer-to-Peer

Peer-to-Peer (P2P) networks are formed by the PCs of the participants. All machines share their resources: computation, storage and communication (although the network is an external paid service). They act both as servers and as clients. P2P networks are self-sufficient and self-organizing, applying protocols in a decentralized way to perform search and location, and to share the burden of object transfer. Due to the fact that coordination is not made by a central authority, all participants have similar functionalities. P2P networks are characterized by high fault tolerance, tolerance to disconnections and to attacks.

Some Peer-to-Peer networks, however, have problems when increasing its size, since search protocols can lead to a heavy load on the network and in the machines. Often, neither guarantee on the duration and success of the search is given, nor on the access to the information [7].

Overlay networks are a type of Peer-to-Peer network, which offer a higher level network service. Using distributed hash tables (DHTs) or network measurements, its topology is built taking into account the content of the peers. DHTs or network measurements can achieve better load balancing, the routing for search mechanisms reduce the time to find an object, and reduce the transfer time of the object. Search becomes deterministic, which significantly improves the performance of such P2P networks [5].

### 3.2. Grid architectures

The computational Grid is an infrastructure for distributed computing targeting at applications, which require a large amount of computational power or data processing capacity. The name is a metaphor of the electrical network (power grid), which is a complex distributed infrastructure but of easy use for the user and which provides power on demand.

The Grid offers access to resources for data processing and other functions not available in a single machine. This is achieved by a Grid middleware, which integrates computational resources of different geographic locations and organizations: this leads to the concept of virtual organization, an alliance of people working together and sharing resources from different organizations. The grid middleware aims at providing organizational transparency, hiding the complexity of unifying resources and crossing organizational boundaries.

The existence of problems which require a huge amount of processing power (for example experiments in physics in CERN), the availability of a concrete architecture (currently based on Web Services) and an open toolkit as de-facto standard (Globus) has made the Grid a success in the high performance computing and related communities.

The current architecture of Globus based on Web Services, Open Grid Services Architecture (OGSA), offers functions for the integration and management of services, including the creation, management of the life cycle and grouping of services, security, policies, access, integration of data, and the management of the workflow.

AccessGrid [10] is an application to create environments for collaboration of research groups. It offers a multi-user work-space and connections of audio and video with other spaces. This is achieved using large screens for projection (1.2 x 5 meters) and bidirectional sound. Interestingly, AccessGrid uses concepts from P2P: It is based on an overlay network, in which audio and video channels circulate by diffusion (IP multicast). Currently, AccessGrid 2.0 uses Globus for managing security in transport (SSL) and authentication (PKI) of the multimedia and multipoint sessions.

### 3.3. Information architecture

Standardization in the structure of information allows that in addition to persons, programs will be able to learn sufficiently about the significance of data in order to process and organize it. This is known as the semantic Web [2]. In the future the web and other applications like collaborative applications need to have information about objects, like having a global database.

The architectural components include semantic (meaning of the elements), structure (organization of the elements), and syntax (communication). The standardization of these components in general terms and within the area of learning and collaboration should make group activities in collaborative applications easier.

### 3.4. Programmable networks

A fundamental difference of the web compared with radio or TV is that the power of the emission determines the area which is covered. Differently, in the web, the resources to serve a document are proportional to the audience, which usually is not predictable [1].

A programmable infrastructure is formed by a large number of machines distributed on Internet, on which by means of a single interface new programs are installed. This allows in an easy way the deployment of new services in the network. The model of programmable networks is close to that of the Grid.

Programmable networks allow automating the installation, deployment, and maintenance of applications and services in different machines in the network. They can adapt to the demand and dispersion of the users.

## 4. Function Dispersion architecture for collaborative work

We propose a system model and architecture, which incorporates the majority of the previously described ideas. The system should integrate persons, who collaborate in a task by means of distributed computers interconnected with a network. Our approach considers light user agents (thin clients), which access to entities and use computational resources without imposing any significant constrain on the capability of computational resources (i.e. mobile appliances, simple devices, PDA,

etc). This approach is similar to the concept of Ubiquitous Computing [11].

In fact, our approach applies to many existing applications, which use a browser as the user agent and a server structured in one or two units, often based on the Model-View-Controller paradigm [3]. By means of web pages or small programs (applets) the browser provides the view and the controller. The server provides the model in terms of a program. This program processes the requests according to the model, which is stored persistently in the data base. The systems called two-tier and three-tier can also be classified to be in this category.

Other applications, which are executed in any machine in a network and view in a terminal using protocols for graphic representation like X, IC, and RDP, also follow this scheme.

We have studied VNC [9], a tool for graphic representation in light clients, which uses the Remote Frame Buffer protocol (RFB). This protocol allows visualizing remotely the user interface of any application on the Internet and from different types of machines. Characteristics of this protocol are its simplicity, requiring only moderate network and machine resources, and being independent of the characteristics of the machine where the visualization is carried out: it imposes minimal requirements on client devices, and it has been implemented successfully on very simple devices.

An extension of this protocol has been developed for a collaborative application [8], in which a group of students visualizes a networked computer lab. Functionalities to facilitate the collaboration between groups of students and teacher have been implemented.

### 4.1. Collaborative interaction levels

In order to classify the requirement for collaborative functionalities we define a number of collaborative levels in the user interface (all have been implemented in a prototype [8]):

a) **Screen**: The content of the screen is shared with several persons. In this scenario, members of a group visualize in each screen the same content of a selected screen. In this screen collaboration is provided like for example by using different cursors or sharing a unique cursor, which is assigned to the different users according to some turn taking or floor control mechanism. In such a scenario, for instance, each student could write a text when its turn is activated, which can be read by the other students.

b) **Window**: The interface of an application is shared with several persons. As example of this scenario we can consider web browsing in groups (a web "guided tour") with browser windows with a multi-user interface (obeys to user interface events from several users). Each group member has a browser window which obeys to user interface events from the local user and those (load of a new URL) sent by the group leader. One of the members of the group could control which pages are visualized in all browser windows. Each member has also local control of his browser window and can read and interact with the page autonomously.

c) **Application**: This scenario considers multi-user applications, which interacts with each user according to the user requirements. Each user can have a different view of the information shared by the participants. For example, we can consider a shared text editor, in which all the participants work on the same document, but each user can visualize and modify a different part of the document.

This model with light user agents, which we suggest, allows that participants from a wide range of devices, like PCs connected to Ethernet or PDAs with wireless access can collaborate. Having light clients the requirement of mobility appears feasible: The user interface of the participants may be the same and migrate from one device to another, or continue to be active even if the participant changes the location and machine.

Light clients appear as a reasonable choice as it can be assumed that in a collaborative scenario the participants of a group change frequently their location. This requires finding solutions to handle such technical and organizational changes, which finally fulfils the requirement of universal and transparent access.

### 4.2. Computational resources

Traditionally, the computational resources are provided previously by "our own" organization in the view of the requirements of the application and the persons using them.

Depending on the application, a server provided by the organization may carry out the collaborative application. This may lead to a problem in the case that the server is administered by a policy which does not adapt well to the application, or if the collaborative application is used by participants which belong to different organizations.

Following the requirements of decentralization, self-organization and self-sufficiency described previously, we propose to apply mechanisms and functionalities applied in P2P networks, in order to avoid centralized components.

There are also cases that groups which collaborate do not have the necessary computational resources. In order to fulfill the criterion of self-sufficiency, systems which execute collaborative applications, should provide

mechanism which allow using external resources as a service in a multi-organizational environment (this is the aim of the Grid, and overlay networks (P2P) and programmable networks).

## 5. Case Study: Middleware LaCOLLA

We have designed, simulated and implemented LaCOLLA, a decentralized and distributed infrastructure (or middleware) offering a set of common services for handling objects, events and groups management to collaborative applications [6]. This infrastructure aims to facilitate the creation of collaborative applications that allow disperse groups to collaborate through the Internet.

The features of this infrastructure are:

- **Groups**: The group is the basic unit of an organization. All actions occur inside a group boundary. A group is considered as a set of people or applications doing a collaborative activity. Collaborative applications will have the group management service by the middleware.
- **Group storage:** Transparency of availability, replication, location, and selection of the best location. LaCOLLA hides to the applications and users the complexity of managing objects in a decentralized, distributed and replicated environment. It guarantees the availability of objects by replicating them and placing the replicas at best locations depending on the users demand.
- **Group awareness:** Awareness information & events dissemination. To facilitate the coordination, the members of a group need to know how the group is evolving (awareness). In LaCOLLA this awareness is achieved by disseminating all the actions done inside the group as events. For the collaborative applications, this dissemination is done in a transparent manner. Apart from informing group members about the evolution of the group, events dissemination helps storage units to decide the best location of objects and to have a virtually strong consistency in the replicated object storage.
- **Dynamism**: Due to dynamic nature of groups (i.e. members are not working full time in the group), LaCOLLA must deal with the connection and disconnection of group members.
- **Interoperability and sharing**: To deal with the diversity of group members the management of groups, objects and events is independent of the applications. This facilitates the access to data from different applications (interoperability) and that different applications share resources related to a group (i.e. two applications use the same presence information provide by LaCOLLA).
- **Self-Organization**: The term "self-organization" is not defined precisely in the literature. Intuitively, it describes the ability of a system to organize its components into a working framework without the need of external help or control. For our purposes we shall understand self-organization as the capability of adding and removing system parts without the need for reconfiguration or the need for human intervention.
- **Decentralized management**: Each member of the group handles the resources provided to the group at their wish. This approach without central authority or coordination influences the system design.

LaCOLLA is based on a peer-to-peer architecture. Each node implements one, two or three of the following functionalities: User agent (UA), repository agent (RA) or group administration and presence agent (GAPA).

The user agent represents users in the system. It is in charge of being notified of all actions done by the user. Once notified, the user agent interacts with other nodes to get the action processed or to get the event distributed to other members. It is also in charge of receiving events about actions done by other members.

Repository agents are dedicated to the storage of the information (events and objects) generated by the group. To facilitate the availability and the accessibility on a potentially large scale, information can be replicated in different storage components.

Group administration and presence agents contain the information about users and groups. They are also specialized on presence information.

The current implementation of LaCOLLA provides the basic mechanisms. They are grouped in the following categories: Events (what is happening), objects (which objects, where), presence (who is connected), and location (where are located the connected members, RA and GAPA). Future work which is developed within [4] includes extending the infrastructure to cover additional aspects like security, disconnected operation, instant messaging.

The results from simulation and initial evaluation with a prototype implementation being developed show how collaborative applications can be constructed with a Function Dispersion architecture using a decentralized, distributed (P2P) middleware which is autonomous and self-organized, and that can be extended to support the requirements presented in this paper.

## 6. Conclusions

The design of systems formed by persons and machines, which interact intensively, requires taking account on the complexity of both sides: the aspects of technology and how human groups behave.

Technology offers opportunities and limitations, which once identified can be exploited to allow new possibilities for collaboration between persons, while at the same time the respecting the autonomy of groups and the integration of persons in disperse locations.

The proposed architecture fulfils the majority of the identified requirements for building distributed collaborative applications, including distributed components, light clients, and resource provision on demand.

Towards this architecture we presented as case study the middleware "LaColla", which provides services for collaborative applications with the goal to simplify the design of real-world distributed collaborative applications.

## References


[1] Ardaiz, O. Application Network Deployment in the Internet, PhD Thesis, 2003.

[2] Berners-Lee, T.; Hendler, J.; Lassila, O. The Semantic Web, *Scientific American*, May 2001.

[3] Burbeck, S. Applications Programming in Smalltalk-80: How to use the Model-View-Controller (MVC), 1992.

[4] CRAC, Proyecto MCYT, 2002-2005, http://research.ac.upc.es/crac/

[5] Doval, D. Overlay Networks, *IEEE Internet Computing,* v7 #4, 2003.

[6] Marquès, J. M. LaCOLLA: una infraestructura autònoma i auto-organitzada per facilitar la col·laboració, PhD Thesis, 2003.

[7] Menascé, D. Scalable P2P Search, *IEEE Internet Computing,* v7 #2, 2003.

[8] Navarro, L. Canal de Aprendizaje, Internal report, 2003.

[9] Richardson, T.; Stafford-Fraser, Q.; Wood, K.R.; Hopper, A. Virtual Network Computing, *IEEE Internet Computing*, Volume 2, Number 1, January/February 1998.

[10] Stevens, R.; Papka, M. E.; Disz, T. Prototyping the workspaces of the future, *IEEE Internet Computing*, v7 #4, Julio 2003.

[11] Weiser, M. Some Computer Science Problems in Ubiquitous Computing, *Communications of the ACM*, July 1993.