Extending the Scope of Asynchronous Collaboration: a Matter of Being Autonomous and Self-sufficient

Joan Manuel Marquès
Universitat Oberta de Catalunya
jmarquesp@uoc.edu

Leandro Navarro
Universitat Politècnica de Catalunya
leandro@ac.upc.es

Thanasis Daradoumis
Universitat Oberta de Catalunya
adaradoumis@uoc.edu

Abstract

Asynchronous collaborative applications and systems have to deal with complexities associated with interaction nature, idiosyncrasy of groups and technical and administrative issues. Inclusion of requirements derived from them is costly (in time, resources and economically). Existing solutions addresses asynchronous collaboration via simplification of requirements and by using centralized models. In this paper we present LaCOLLA, a fully decentralized infrastructure for building collaborative applications that provides general purpose collaborative functionalities. The provision of those functionalities will avoid applications deal with most of complexities derived from groups and its members, what will help inclusion of collaborative aspects.
The implementation of LaCOLLA follows the peer-to-peer paradigm and pays special attention to autonomy of its members and to self-organization of the components of the infrastructure. Another key aspect is that resources (e.g. storage) and services (e.g. authorization) are provided by its members (avoiding dependency from agents not belonging to group).

Keywords*: collaborative infrastructures, peer-to-peer collaborative systems, peer-to-peer middleware.*

1.Introduction

One of the most significant benefits of Internet is the improvement on people interactions and communication. E-mail, Usenet News, Web and Instant Messaging are four of the most well-known and successful examples of this. Internet has allowed the creation of asynchronous virtual communities in which its members interact in a many-to-many basis. Many-to-many interaction is not something we typically experience in the physical world. That has transformed the way people organize to realize a task, the way people with common interest gather and share them, the way people learn, ... But after 10 – 15 years of great excitement the pace of this transformation is slowing down because collaboration is much more than e-mail, instant messaging and discussion tools. Among the reasons for this slow down, in this paper, we outline that Internet technology is designed on a one-to-one basis and that applications with collaborative necessities have to deal with complexities derived from:

- *Interaction nature*: participants are dispersed at Internet, many-to-many collaboration, people participate in the collaboration at different times, same person connects from different locations at different times (home, work, notebook) of the day, ...
- *Idiosyncrasy of groups*: flexibility, dynamism, decentralization, autonomy of its participants, groups exist while its members participate in group activities and while they provide necessary resources, few participants, ...
- *Technical and administrative issues*: guarantee the availability of information generated in the group, interoperability among applications, security aspects (authorization, access rights, firewalls), ... participants belong to different organizations or departments with different authorities that impose rules and limitations in order to facilitate administration, internal work and individual use. [1]

Development of applications that take into account all those requirements are costly (in time, resources and economically). This has provoked that applications include minimal collaborative functionalities. Even collaborative applications focus only in a few of those aspects (the key aspects for the application) and neglect other aspects. In that way, most of the solutions are centralized and use resources administrated by a third authority (what mean that some of the resources not belong to members of the group).

From the architectural point of view, most of those systems are based on client/server models. Even the

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

systems and applications that claim to be peer-to-peer, like Groove [2] or some Instant Messaging systems, have some centralized functionalities (like rely servers, or authentication authorities).

In this paper we present LaCOLLA, a fully decentralized infrastructure for building collaborative applications that provides general purpose collaborative functionalities. The provision of those functionalities will avoid applications deal with most of complexities derived from groups and its members. This simplification will help inclusion of collaborative aspects into applications.

LaCOLLA is implemented following the peer-to-peer paradigm and pays special attention to autonomy of its members and to self-organization of the components of the infrastructure. Another key aspect is that resources (e.g. storage) and services (e.g. authorization) are provided by its members (avoiding dependency from third party agents).

The rest of the paper is organized as follows: Section 2 presents the requirements that should satisfy an asynchronous collaborative infrastructure. Section 3 describes the functionalities and architectural aspects of LaCOLLA. In this section, is also presented what we have named virtual synchronism, a key functionality that provides all occurred events to members of the group and allow them to have access to last version of any object in a time that they perceive as immediate. Section 4 presents experimental results. In section 5 our present work is presented (LaCOLLA prototype) and sketches future work. We conclude in Section 6.

2. Requirements for an asynchronous collaborative Infrastructure

As mentioned in the previous section, asynchronous collaborative applications have to deal with many aspects to achieve collaboration. The basic requirements an infrastructure should satisfy to facilitate the development of this kind of applications are:[3]

- *Oriented to groups*: group is the unit of organization.
- *Internet-scale system*: formed by several components (distributed). Members and components can be at any location (dispersion).
- *Universal and transparent access*: participants can connect from any computer (or other digital device), with a view independent from the connection point. E.g. web provides it.
- *Decentralization*: No component is responsible of coordinating other components. No component is the only component that possesses a certain information. Centralization leads to easy solutions, with critical components, that condition the autonomy of participants.
- *Self-organization of the system*: system has the capability to function in an automatic manner without requiring external intervention. Has the ability of reorganizing its components in a spontaneous manner in presence of failures or dynamism (connection, disconnection, or mobility).
- *Group availability*: Capability of a group to continue operating with some malfunctioning or not available component. Replication (of objects, resources or services) can be used to improve availability and quality of service.
- *Individual autonomy*: members of a group decide freely which actions perform, which resources and services provide, and when will be connected or disconnected.
- *Group's self-sufficiency*: group is able to operate with resources provided by its members (ideally) or with resources obtained externally (public, rent, interchange with other groups, ...)
- *Allow sharing*: information belonging to group (e.g. events, objects, presence information, ...) can be used by several applications.
- *Security of group*: guarantee the identity and the selective and limited access to the shared information (protection of information, authentication).
- *Availability of resources*: provide mechanisms to use resources belonging to other groups (public, rented, interchange between groups to improve availability, ...)
- *Transparency of location of objects and members*: applications don't have to worry about where are the objects or members of the group. Applications use an identifier independent from location.
- *Scalability*: in number of groups is guaranteed because each group uses its own resources.
- *Support disconnected operational mode*: work without being connected to group. Very useful for portable devices.

3. LaCOLLA

LaCOLLA is an infrastructure that follows the requirements presented in the previews section. Three main abstractions have lead all the design process of LaCOLLA: is oriented to groups, all members know what is happening in the group, all members have access to latest versions of objects. Those abstractions are concreted in the following functionalities:

3.1 Functionalities

LaCOLLA provides to applications the following general purpose functionalities: [3]

- *"Immediate" and consistent dissemination of events*: information about what is occurring in the

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

group is spread among members of the group as events. All connected members receive this information right after it occurs. Not connected members receive it during the re-connection process.

- *Virtually strong consistency in the storage of objects*: components connected to a group can access the latest version of any object. Objects are replicated. When an object is modified, if an application asks for the object, LaCOLLA guarantees that the last version of it will be provided (even thought all replicas are not consistent and it will require some time to have all of them consistent).
- *Presence*: know which components and members are connected to the group.
- *Location transparency*: applications don't have to know location (IP address) of objects or members. LaCOLLA resolves it internally (similar to domain name services like DNS).
- *Instant messaging*: send a message to a subgroup of members of the group.
- *Management of groups and members*: add, delete or modify information about members or groups.
- *Disconnected mode*: allow applications operate offline. During re-connection, the infrastructure automatically propagates the changes.

3.2 Architecture

The architecture of LaCOLLA [3] is organized in three kinds of components. Components coordinate through mechanisms grouped in nine categories.

3.2.1 Components

Component behaves autonomously. Peers decide to instantiate one, two or three of the following components:

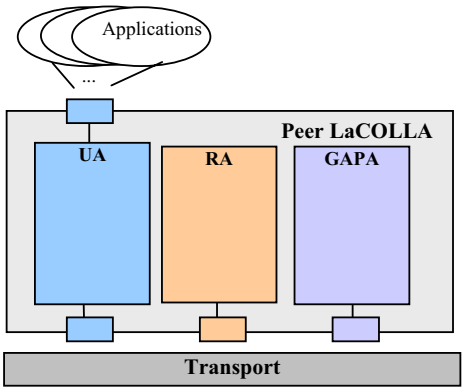

Figure 1. LaCOLLA peer.

- *User Agent (UA)*: Interacts with applications. Through this interaction, it represent users (members of the group) in LaCOLLA.
- *Repository Agent (RA)*: store objects and events generated inside the group in a persistent manner.
- *Group Administration and Presence Agent (GAPA)*: in charge of administration and management of information about groups and its members. Also is in charge of authentication of members.

3.2.2 Mechanisms

Components interact one to each other in an autonomous manner. The coordination among the components connected to a group is achieved through internal mechanisms. Internal mechanisms have been grouped in: events, objects, presence, location, groups, members and instant messaging. They are implemented using weak-consistency optimistic protocols [4] and random decision techniques. Table 1 presents which components are involved in each category of mechanisms.

Table 1. Shows which categories of mechanisms implements each kind of component.

Categories of Mechanisms	UA	RA	GAPA
Events	X	X	-
Objects	X	X	-
Presence	X	X	X
Location	X	X	X
Instant Messaging	X	-	X
Groups	X	X	X
Members	X	-	X
Security	X	X	X
Disconnected operational mode	X	-	-

Our objective was to prove that the proposed architecture can provide the presented functionalities in a way that satisfies the requirements. We never had as objective to implement the best choice for each mechanism. Table 2 presents a brief summary of algorithms used to implement presence, events and objects mechanisms, which are the key mechanisms to provide virtual synchronism.

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

Table 2. Functional description of mechanisms related to presence, events and objects.

Mechanisms		Description
Presence	Connection	Component authenticate in any of the GAPAs connected to group. The GAPA returns to connecting component the components and members that it knows that are connected. Connecting component sends a message to all components that knows that are connected to inform of its connection.
	Disconnection	Component sends a message to all connected members informing about its disconnection.
	Dissemination of presence information	All messages include a summary of all components that sender knows that are connected. If a component has not sent any message for a long period of time, sends a message to other members saying that still is alive.
	Consistence of presence information	Periodically, components do consistency sessions* with other components to guarantee that all components know who is connected to group.
	Detection of components no more connected	When a component realizes that a component has not done activity for a long period of time, sends to it a message asking if is still connected. If no answer is received, takes it out of its connected-members information.
Events	Dissemination of events	When a new event is generated, the component where the event was generated sends it to all connected components.
	Consistency of events	Periodically, components do consistency sessions* to guarantee that all components have all events.
	Storage of events	RAs store all events generated in the group.
	Storage of events at UA	UAs store events during a session. Members connect to group from different UAs; for this reason, UAs only have to guarantee that have events during the session.
Objects	Storage	When a new object (or version) is created, the UA sends it to an RA to be stored in a persistent manner. The RA sends an event to all connected members informing about the new object.
	Obtain	Due to events mechanisms, all connected members know all objects that are in the system. Then, when an UA requires an object, ask it to any of the RA that has a replica of the object.
	Replication	Periodically, each RA tries to replicate all locally stored objects that are replicated a number of times lower that a replication factor defined to group. Those replicas are done sending the object to any RA belonging to group that doesn't has the object.

* A variant of weak-consistency sessions proposed by Golding[4].

As can be seen in table 2, LaCOLLA combines push, pull and autonomous decision behaviors. Some examples of each kind are: a) push: when a component has an information that interest other members (e.g. object modified, member connected, etc.), sends it to other connected components; b) pull: periodically, components do consistency sessions to get information about presence, events, ... that they don't have; c) autonomous decisions: when an RA has an object that is replicated less times than the replication degree fixed to group, it chooses another RA and copies the object. When a component realizes that another component has been inactive for a long period of time, takes him out of its connected components (if the component is still connected, he will learn about him in a near future).

Even though push behavior is very used, neither components nor network are saturated because groups are small. Another key characteristic is that each component knows at any moment (or discovers them in a short time) which components are connected to group.

This combination of autonomy of components and direct communication among them (in a peer-to-peer manner) along with the ownership of resources provides a flexibility that suits idiosyncrasy of groups.

Among the aspects that characterize LaCOLLA one that deserves special attention is what we have named virtual synchronism.

3.3. Virtual synchronism

LaCOLLA guarantees to applications that all events delivered to LaCOLLA will be received almost immediately by the rest of connected members. This guarantee provides them the feeling that they know what is happening in the group while it is occurring. No connected members will receive the events during the re-connection process.

LaCOLLA also guarantees that all objects belonging to group will be available immediately by all members.

The sum of both guarantees is what we named virtual synchronism. Apart from the up-to-date perception that group members have at any moment, the virtual synchronism has an interesting side effect. This side effect is very useful in an autonomous, decentralized and dynamic storage system: because all components know the location of all objects (and their replicas), components access them directly (without requiring a

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

resolver that informs about location of last version of the object). This allows LaCOLLA to have an autonomous and decentralized policy to handle objects and its replicas at the same time that it guarantees immediate access to last version.

4. Validation

We have a prototype of LaCOLLA. Now we are implementing some collaborative applications to study the performance of LaCOLLA in real situations. The results presented here come from a simulator implemented to validate the proposed architecture. The simulator uses J-Sim [5] as network simulator and implements the three kinds of components and the internal mechanisms necessary to prove that LaCOLLA self-organizes even though it operates in an autonomous, decentralized and self-sufficient manner. The mechanisms implemented are: events, presence, objects and location. We also proved virtual synchronism.

Several experiments were done with different degrees of dynamism (failures, connections, disconnections or mobility), with different sizes of groups (from 5 to 100[**] members) and with different degrees of replication (different number of RAs and GAPAs). All components are affected by dynamism.

Experiments showed that, in spite of the dynamism and the autonomous and decentralized behavior of components, LaCOLLA requires short amount of time to have up to date the information referring to internal mechanisms in all components. Experiments also showed that members know what is happening in the group and that they have access to latest versions of objects in a time they perceive as immediate.

Simulation was done in two phases. First phase simulated a real situation, whit all internal mechanisms operating. In this phase events and objects were generated depending on different activity degrees (simulating different kind of members). Also failures and changes (connections, disconnections and mobility) occurred according to different degrees of dynamism. This first phase was used to show the tolerance to failures and changes (if system is able to reach a consistent state after failures and changes occurred during the first phase, it means that LaCOLLA has been able to recuperate from them). Duration of first phase was different for each simulation.

Second phase was used to show the ability to recover automatically. During this phase, only were active internal mechanisms (no failures, connections, disconnections, mobility, new events or new objects). Time to reach a consistent state was calculated (by consistency we mean: all components have same information about presence, location, available objects, events, and GAPAs belonging to group). More precisely, we evaluated how much time requires LaCOLLA to achieve:

a) self-organization: all connected components have consistent the information about all internal mechanisms.
b) virtual synchronism: all connected components have all events and have consistent the information about available objects.
c) presence+location: all connected components have consistent information about presence and location.

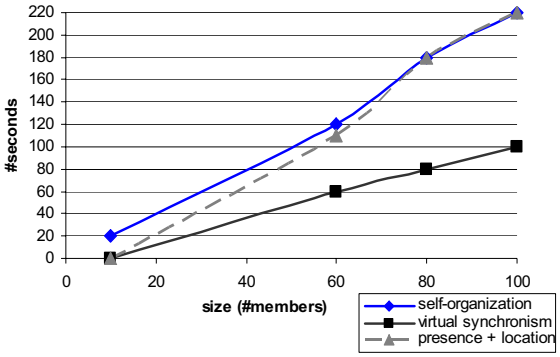

Figure 2. Time required by LaCOLLA to reach self-organization, virtual synchronism and presence+location consistency after being active during a period of time. (Failures, connections, disconnections, mobility occurred, and members did actions that generated events and objects).

Figure 2 shows the time required by LaCOLLA (depending on group size) to be self-organized, to provide virtual synchronism, and to have consistent information about presence and location. Note that, for groups of usual size (10 members), LaCOLLA has a good performance: requires 20 seconds to self-organize, and less than 10 seconds to provide virtual synchronism. Deserves special attention the fact that, even though all components don't have consistent all information about internal mechanisms (self-organization), connected members know all what is happening in the group and have access to last version of objects (virtual synchronism) in a time that they perceive as immediate. This is due to the decentralized implementation of internal mechanisms and to the fact that non-key mechanisms have long-term consistency policies. In this figure is also plotted the required time to have consistent presence and location mechanisms because they have a great influence in the achievement of self-organization.

When size of groups increases, the required time grows, but still maintains low enough values for

[**] (Due to human limitations) a group with 100 members is an extreme situation. Collaboration occurs in small groups. Big groups tend to split in smaller groups. Proxy techniques can be used to provide information to members that are not active participants.

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE

asynchronous collaboration (e.g. 60 members: self-organizes in 2 minutes, provides virtual synchronism in 1 minute). This proves that LaCOLLA can be used in situations where big groups require asynchronous sharing capabilities (understand big groups on terms of collaborative groups). With bigger groups, proxy techniques should be used.

5. LaCOLLA prototype and future work

At present time we have a prototype that implements the basic functionalities presented in this paper. We are also implementing some collaborative applications (a chat tool and a document sharing tool) that benefits from LaCOLLA. Those "real" applications will help us to improve the architecture and implementation of LaCOLLA.

The prototype implements mechanisms related to presence, events, objects and location. The first three are described at table 2. The last category of mechanisms is strongly related to presence. More detailed information can be found at [3]. At present time, our prototype doesn't implement object mechanisms related to delete and conflicts. Our next step will be to implement them. The literature has treated widely all those concerns [6][7]. Our approach will be to adapt one of those solutions to our concrete situation.

In parallel, we are working on extending LaCOLLA. Our main effort is focused on defining and implementing the necessary components and mechanisms that will allow members of a group provide computational capabilities in a dynamic and decentralized way to other members of the group. At first step, this will mean that a member could execute tasks using computational resources belonging to group (and provided by group members). But our goal is that members of the group could deploy services using computational and storing resources belonging to the group. The components and mechanisms of the first step are based on the ideas used to design JNGI [8], a decentralized and dynamic framework for large-scale computations for problems that feature coarse-grained parallelization. The components of JNGI communicate using JXTA [9]. In our case we use the communication facilities of LaCOLLA.

We also are working in the extension of the infrastructure to provide a better support for synchronous collaborative applications.

6. Conclusions

Asynchronous collaborative applications have to adapt to group idiosyncrasy and interaction nature. This is achieved by stressing autonomy and self-sufficiency. The architectural paradigm that better adapts to it is peer-to-peer. Infrastructures like the one described in this paper will help the inclusion of asynchronous collaborative functionalities in applications.

In this paper we have described the general characteristics and properties of LaCOLLA, a decentralized, autonomous and self-organized infrastructure for building collaborative applications that operates with resources provided by its members, and that adapts to group's idiosyncrasy and to interaction nature.

The developing and use of applications that are implemented on top of LaCOLLA will help us to improve the architecture and implementation of LaCOLLA. New functionalities (like computational capabilities) will open new possibilities of cooperation.

7. Acknowledgments

This work was partially supported by MCYT-TIC2002-04258-C03-03.

8. References

[1]Foster, I.; Kesselman, C.; Tuecke, S. (2001). The Anatomy of the Grid Enabling Scalable Virtual Organizations. *Lecture Notes in Computer Science.*

[2] Groove: http://www.groove.net

[3] Marquès, J.M. (2003). *LaCOLLA: una infraestructura autònoma i autoorganitzada per facilitar la col·laboració*. Doctoral thesis, <http://people.ac.upc.es/marques/LaCOLLA-tesiJM.pdf>

[4] Golding, R.A. (1992). *Weak-consistency group communication and membership*. Doctoral Thesis, University of California, Santa Cruz.

[5] J-Sim: http://www.j-sim.org

[6] Kistler, J. J.; Satyanarayanan, M. (1992). "Disconnected Operation in the Coda File System". ACM Transactions on Computer Systems. Febrer 1992, 10(1): 3-25, < http://www.coda.cs.cmu.edu/>.

[7] Guy, R. G.; Heidemann, J. S.; Mak, W.; Page, T.; Popek, G. J.; Rothmeier, D. (1990). "Implementation of the Ficus replicated file system". A: USENIX Conference Proceedings, p. 63-71. USENIX, juny 1990.

[8] Verbeke, J.; Nadgir, N.; Ruetsch, G.; Sharapov, I. (2002) *Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment*. Manish Parashar (Ed.): Grid Computing - GRID 2002, Third International Workshop, Baltimore, MD, USA, November 18, 2002. Lecture Notes in Computer Science 2536 Springer 2002, ISBN 3-540-00133-6. <http://jngi.jxta.org/>

[9] JXTA: http://www.jxta.org/

Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'04)
1524-4547/04 $ 20.00 IEEE