WWG: a Wide-Area Infrastructure to support groups

Joan Manuel Marquès
Universitat Oberta de Catalunya
Universitat Politècnica de Catalunya
Barcelona, Spain
jmarquesp@campus.uoc.es

Leandro Navarro
Computer Architecture Department
Universitat Politècnica de Catalunya
Barcelona, Spain
leandro@ac.upc.es

ABSTRACT

Group learning at Internet scale is becoming more frequent in university courses. This complex process requires support by distributed computing learning support infrastructures.

This paper describes the design of WWG (World-Wide Groups): a distributed and decentralized infrastructure with the aim of supporting distributed group learning and team work, centered on the distribution of events, so that every participant can be notified and thus be aware of the actions, changes, progress of the groups he belongs to.

The design issues, requirements and the resulting architecture are presented. WWG is based on a multi-component architecture where metainformation agents are responsible for helping the events to reach the members of the group; the repository agents are responsible for the storage of group information; and user agents are responsible for the representation of users (sources and sinks of events). In this paper we tried to show that, applying events transformation policies, WWG is scalable at group level.

Keywords

CSCL environment, event distribution, distance cooperative learning, virtual groups.

1. INTRODUCTION

When people do group work or cooperative work two aspects are always present: a) they share objects, b) they someway know what are doing other group members. When an activity involves several people working at distance, we realize even more the importance of these two aspects. Every time there are more applications trying to address this area [2], but when we look into the details, we realize that this is not related to a concrete tool. During a long-term cooperative activity we may end up using many different tools to do work. We realize that objects we use are stored on different repositories, are managed from diverse tools, etc. Every piece of the group context is produced on different tool environments, which is not complete and difficult to combine with other pieces.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GROUP'01, Sept. 30-Oct. 3, 2001, Boulder, Colorado, USA.
Copyright 2001 ACM 1-58113-294-8/01/0009…$5.00.

In addition, we might work from different locations (home, office, etc), or use a mobile device to let us work from many more locations. Given the above considerations, it seems to be necessary to provide applications with an infrastructure able to collect and distribute information on actions by people, and able to provide a distributed repository for objects.

A simple sequence of how it works:

a) A group member interacts with a collaborative application

b) The application informs the infrastructure on the actions done and/or the product of these actions (objects)

c) The infrastructure distributes that information to other interested parties (group members)

d) Interested parties receive that information that is processed by the corresponding applications.

This work is centered in the definition and validation of an architecture for this infrastructure.

In particular, our experience comes from theoretical and practical work on cooperative distance learning at the "Open University of Catalonia" (UOC: http://www.uoc.es), a virtual university providing university education to the Catalan and Spanish speaking world, also at UPC (http://www.upc.es), an established university giving in person and half-distance university engineering education.

Complete or partial distance learning, mediated by the use of computer networks, is more commonplace as a solution to the difficulties of having students and sometimes professors in remote locations. Participants have temporal restrictions due to overlapping activities, or just self-pace learning is preferred.

Since the beginning of the Computer Sciences studies in the UOC (1997), learning by working in groups has been a main issue. Several experiences [6] [7] have enlightened the design of our infrastructure. We realized that to most people computer mediated group learning is a brand new way of learning and they need first to get used to it. But this situation raises problems of isolation, a fragile sense of community, and the lack of the feedback that always occurs spontaneously when people share a common physical location [14] [16].

Cooperative Learning [17] and in general, group work is an activity that is increasingly being perceived as beneficial and necessary for a more active and better learning process. Small groups of 3-5 students plus the tutor, doing cooperative distance work have proved to be more effective than individual distance learning [7]. The reason is that Cooperative Learning is a

fundamental source of rich feedback for all participants. In [17] cooperative learning is described as:

"Cooperative Learning is a relationship in a group of students that requires positive interdependence (a sense of sink or swim together), individual accountability (each of us has to contribute and learn), interpersonal skills (communication, trust, leadership, decision making, and conflict resolution), face-to-face promotive interaction, and processing (reflecting on how well the team is functioning and how to function even better)."

Therefore, distance cooperative learning needs some extra awareness information to know what the other members of the group are doing, to know their feelings about the activity, to detect potential conflicts as soon as they occur, etc. That awareness information is not well provided by present applications because the infrastructure they use was designed to support isolated work. Our proposal is focused to provide that awareness information as the key design aspect of our infrastructure.

The challenge of supporting cooperative learning or in general, group work, in a distributed context is the motivation of this work.

The extension of learning activities beyond a campus scale to participants spread over the Internet presents several problems of scalability and inter-operability between systems based on different architectures. This suggests the need for a cooperative learning infrastructure to support a large number of small groups distributed over the Internet.

The management of the learning process and the representation of meta-information about objects or components used in that process are being supported by initiatives such as ARIADNE [1], IMS [15] or the IEEE Learning Task Force [21]. Our work is complementary to those activities.

This paper describes the design of WWG: a distributed and decentralized infrastructure with the aim of supporting distributed collaborative learning and team work, centered on the distribution of events, so that every participant can be notified and thus be aware of the actions, changes, progress of the groups he belongs to. Participants may need to transform (summarize, condense) a rich collection of events provided by WWG, to give the required information to group members with diverse degree or mode of participation. This is clearly the case in our setting for tutors, teachers, professors, assistants, supervisors, moderators, evaluators, etc. The amount and form of "required information" is application and situation specific, and therefore beyond the scope of WWG.

WWG has been designed for situations where participants interact and work asynchronously, but receive synchronously information about the actions done in the group. This event distribution mechanism provides consistency, sense of immediacy, sense of complete information about what's going on. This infrastructure has to work on Internet scale, be accessible from any site, from mobile users, and support the high degree of interaction and information exchange that occurs on any collaborative setting with many groups, and specially on learning environments.

Major issues are presented in the next section. These issues are translated into system requirements in section 3. The central mechanism of event distribution is discussed in section 4. Event distribution prescribes the interaction with users, and with repositories where documents, history of events, configuration and status of groups are stored. This separation between user interaction - event distribution - storage determines in section 5 the overall architecture of WWG. Section 6 describes how it works in more detail. Previous work that has influenced our design is described in section 7. The paper ends with a description of work in progress and conclusions.

2. ISSUES AND REQUIREMENTS

The scenario as presented before is large, diverse and it has been studied before [9]. The following issues have been considered the most influential to our design:

- Multiplicity: people may belong to several groups at the same time. Implication in every group varies (absence, passive, active participation generating large amount of awareness information), based on diverse communication and dissemination mechanisms (e.g. mailing lists, nntp newsgroups, web forums, workflow). Users need facilitation to handle the complexity of multiplicity and diversity.
- Group membership may be relatively small, even though there may be large groups.
- Awareness: effective group work requires that members must be aware of the progress of the group: what others are doing, at low cost, at a glance. This knowledge includes: actions done on objects, who did these actions, what other people is doing. It is very important that people have up-to-date and rich awareness information.
- Multiple locations: members may connect from different locations: physical location, organization, network provider. This implies variance of working hours, delay, bandwidth, and other traffic characteristics may change significantly.
- Quality of service: The degree of accessibility and reliability of the system improves significantly when information is available in several locations (distributed and replicated). Clients will be offered the most accessible server from the set of currently available.
- Mobility: one person may connect from different environments: work, home, mobile, etc. The view of the groups must be the same from any location. Most users during the day may connect from various locations.
- Degree of connectivity: many group activities do not require to be connected to the rest of members. Given that connections may be expensive, not very reliable sometimes (e.g. analog phone, mobile devices), people may choose to work in *connected* mode: operations are synchronously applied to the group repository, or *disconnected (off-line)* mode: operating locally and connecting to synchronize (conciliate the state of the local and the on-line repository).

In the design of WWG we have addressed each of the previous issues. These issues have been translated into requirements that are briefly described in the following basic requirements for an infrastructure to handle easily and efficiently many learning/work groups at Internet scale:

- Information must be accessible at any time, and be managed transparently. The user does not have to worry about the accessibility and replication of information.
- The user needs the appropriate amount of information produced by the group in form of documents, messages and events (awareness information).
- The system must be scalable: large number of participants, large number of events, participants distributed across large distance, decentralized: no central control/view.
- Group members must have information accurate, updated and consistent about actions being carried out by the rest of the group.
- Objects may be accessible from any location with an appropriate (interactive) response time.
- Access must be transparent and independent to where objects are stored.
- Adaptable to the needs of users: info should be where is more convenient to users. The user also requires availability, reliability and a good access time.
- Adaptive to the needs of the system: load balancing, balancing of storage, minimizing the exchange of information.
- Multiple access points: when a user moves to a new location, the system must adapt dynamically and provide a closer service access point.

Existing systems do not support the above requirements and issues. The goal of WWG is to provide an infrastructure for information management and propagation, without prescribing how information is represented or how applications operate.

3. EVENT DISTRIBUTION

Given that WWG is aimed at supporting learning and working in groups, the key factor is that group individuals should be informed immediately of whatever occurs within their groups. When people do work every action produces an event (metainformation about the actions). Events belong to a group and they must be offered to all members to let them be aware of what's going on in the group. As events only inform about an action, they are small information objects.

Therefore, events must be distributed, as soon as possible, to every group member. It is very important to separate the notification of an action on an object (event distribution) from the object itself. Notification of an action is provided by the event distribution mechanism.

- In terms of system design, the event distribution mechanism allows us to do the following assumptions: Events provide "maximum information": when learning or working activity is done in groups it is of great importance to have the maximum amount of information about what are doing all participants. For us, "maximum information" means both the number of events received by a member and the amount of information that every event conveys.
- Events provide "sense of immediateness": event distribution provides information about what is happening now in the group. Getting the actions done within the group a short time after occurrence allows the members to figure out the evolution of the group.
- Consistency through events: real-time and consistent distribution of events can lead to a consistent distributed and replicated system. Consistency is possible because the system always knows where the latest version of every object is located. Protocols to preserve the "natural ordering" of events (causal and total order where needed) have to be included.
- Events may be used to select the best location for an object. The origin and destination of events helps to decide the best place in the system to store objects.

Once we have decided that our system is based on event distribution, the next step is to design an architecture to guarantee a distribution of events that facilitates the achievement of these assumptions.

4. ARCHITECTURE OF WWG

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 has to interact with the rest of the system to get an action processed or to get an event distributed to other members of the group. It is also in charge of receiving events about actions done by other members of the group and to provide this information to the user.

Repository agents are dedicated to the storage of the information generated by the group. To facilitate the availability and the accessibility the information on a potentially large scale, information may be replicated in different storage components depending on the needs of every group.

Meta-information agents contain the necessary information that allows to route events between the user agents and repository agents belonging to a group. They also contain information about users and groups. They have a *passive functionality* (efficiently routing and distributing event information to interested agents), and an *active functionality* (suggesting the best meta-information agent for each user agent, helping repository agents to decide the best location and the number of replicas needed for each object).

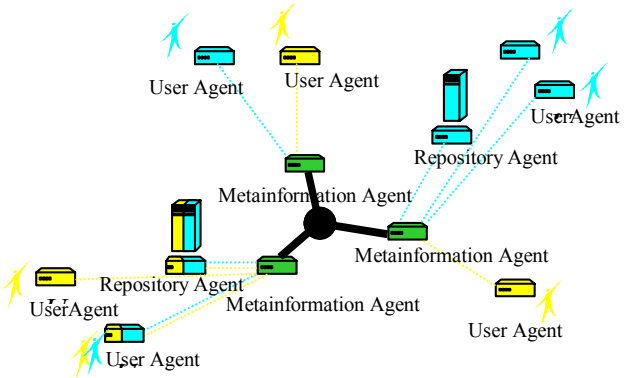

Figure 1. The components of the WWG infrastructure

WWG is not an application. It is an infrastructure that provides functionality to applications. In the next table we classify the different functionality of WWG:

Table 1. WWG functions and agents involved

Function	User Agents (UA)	Repository Agents (RA)	Metainfo. Agents (MA)
Event History	May have a local copy of generated or received events	√	
Object Storage	May have a local copy of accessed objects	√	
Event transformation (aggregation, grouping, summarization, etc)	√ (For synch. dissemination)	√ (For asynch. dissemination)	
Conflict detection		√	
Group naming service			√
User and group* Administration			√
Optimize information access (best location of objects, number of replica, etc.)	Provide information to MA	Provide information to MA	√ (Using information from RA, UA)
Interaction with the user: disseminating events got from the local user	√		
Interaction with the user: present to local user events received from remote agents	√		

* Metainformation agents contain authentication information for users and groups. These two kinds of authentication should be handled separately. Public groups will not need that functionality.

The generated events must arrive to all interested group members (user agents and repository agents). Connected members will get them as they will occur. Members that are not connected will get them all the next time they connect. It is clear that if WWG is used openly on the Internet (millions of people with millions of potential groups), the number of messages generated may get too large. In the next section we will look in more detail to the types of events and how to process events to reduce the number of messages sent.

Repository agents cooperate to provide distributed and/or replicated network storage for objects. Group members should have transparent access to objects they may be interested with a reasonable quality. Event information, that is very dynamic and abundant in any collaborative setting, is also useful for repository agents to decide where objects have to be located.

5. THE PROPAGATION OF EVENTS

To achieve the assumptions of consistency through events, "sense of immediateness" and maximum of information a thoroughly study of events is required. Sending all the events generated is not enough to fulfil those assumptions. This option could flood the system and overload end users with too many events: messages should be prioritized.

Events related to conflicts must be sent immediately. Events that modify the global state of the system (i.e. create, delete or modify a document or an user) must be sent as soon as possible. Different policies can be applied to the events that are informative, which will be the majority[1] of the events generated. It is not the aim of that paper to study those policies, nevertheless we have considered four strategies: aggregation (e.g. when 10 actions occur in an object, send a single event indicating that 10 actions have occurred); grouping (e.g. when several events goes to the same destination, send all of them in the same message); delaying (e.g. when a lot of events are generated, wait a little before they are sent); summarization (e.g. some users may not want to get all the details on every action done in a group. A teaching assistant monitoring a group of students may just want to know who and which kind of actions have been done and how many times)

Experience at UOC confirms that informative events are the most frequent. UOC is a virtual university that articulates all interactions through its virtual campus. During three semesters, students of two subjects have used BSCW [2] to articulate their project work in small groups (3-4): Information Structures, where students have to do activities in groups; and Software Development Techniques, where students have to develop a software project collaboratively; and a virtual group constituted by 9 tutors, where they discussed about the contents and evaluation of the subject they were teaching. During that experience, 23.566 events were generated. From those, 80% were informative events and 20% were modify-state events, and only 2.2% were events that may cause conflicts if not propagated immediately. This supports our intuition that most of the events generated in a system were informative events. That conclusion is even reinforced because, in the version of BSCW we used, a read event is only generated the first time a user reads a document. Successive read events are not recorded. In WWG we want to distribute all the events generated on the system to give the "maximum information" to the users. Then, all the read actions would need to be distributed.

Regarding to users, a similar problem occurs. The experiences we have done at UOC proved that users have a finite capacity of processing events [7]. A user, depending on the number of groups he belongs to, on the activity of those groups and in his degree of involvement, needs to receive events with a different level of abstraction. For instance, in a group formed by three people writing a document, all the members may want to get all the events; but a tutor responsible for six groups, with three members in each group generating events, can be easily overloaded. In this case, the tutor needs fewer but more abstract events. These new events are the combination of several related events. Examples of those new events could be: *this group is working very hard*; *a member of this group is not participating on the course*; or *in this group every member has produced at least one working document*.

[1] This was confirmed by the analysis of events generated by an 18 months experiment with several small groups of students and tutors using BSCW.

When objects are used by different people some conflicts may appear. In asynchronous activities, the conflicts will be rare. In an environment such as WWG most of the conflicts can be avoided by choosing carefully some design alternatives. Even that, conflicts are still possible and the system must be able to solve them. WWG provides a special kind of event, the conflict event, used when a conflict is detected. That kind of event has high priority and it is sent to the different parties involved. If the conflict can't be solved automatically, the members of the group will be informed and someone will be responsible for the explicit resolution (as in [19]). Conflicts and conflict resolution has been studied in a separate research report.

The diversity of kinds of events can be classified under different points of view. A possible classification is about the kind of information contained in the event:

- User-actions events: events generated by an application (task-oriented awareness [23]) or by the user agent (social awareness) for each user action. Examples of those kind of events are: read, create, delete, modify, copy, paste, undelete, etc. documents or messages.
- Inferred events: virtual group members need information of how the group evolves. The inferred events are particular interpretations about how the group is progressing. The user agent (or a client application) has information about the group and the actions done by the local user. With all that information, like an external observer, the user agent infers events about the group evolution. Those events are just automatic inferences. They aren't true nor false.

Events can also be classified thinking on the immediateness required by members of the group for the events:

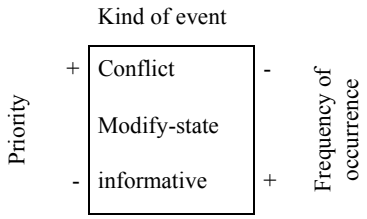

Figure 2. Priority versus frequency for three classes of events

- Conflict events: events that inform that a conflict has occurred or the events trying to solve a conflict.
- Modify-state events: events produced after an action that modifies the global state of the system: a new document, a delete action, a change of location, etc.
- Informative events: events that inform about actions that do not modify the global state of the system. Those events include actions such as read document and inferred events.

5.1 Group scalability

As we pointed out in the previous section, if a lot of events are generated the system can be overloaded. We have done various computer simulations of WWG with different number of events and users, and from these we have observed an acceptable degree of scalability in WWG. The simulations have been done under the following assumptions:

- Three kinds of users with the following behavior:

Table 2. Behavior of several user profiles in terms of events

	Informative events generated	Modify-state events generated	Events per time unit
Observer	99%	1%	0.01
PassiveMember	90%	10%	3
ActiveMember	50%	50%	5

- Observers don't want to receive all the events within the group. They only want to be informed about how the group evolves in general terms. Summary events are sent to them with low frequency just to keep them informed. E.g. In the simulation we have supposed that if the time unit is 5 minutes and observers want to get summary events each 8 hours, summary events have to be sent every 192 units of time.

The following table illustrates the distribution of participants among active, passive and observer members respect to the size of the group:

Table 3. Distribution of participants respect to group size

Number of group members	Active members	Passive members	Observers
4	4	0	0
5	4	0	1
8	4	4	0
9	6	2	1
18	8	8	2
30	5	10	15
60	5	10	45

The result of our the simulation has shown that:

- The number of messages sent depends directly to the number of group members.
- Aggregation reduces the number of messages sent. This reduction is a constant and is affected by the average of informative events (aggregable events) sent by the users per time unit.
- When the number of members of a group grows, the percentage of observers increases rapidly [7]. In those situations, summarize events is an effective way to achieve scalability.
- For not very large groups, the number of events sent depends on the number of active and passive members.

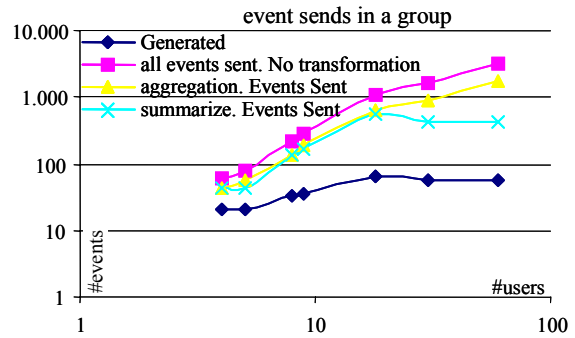

Figure 3. Events sent in one group versus number of members

In our simulations we have only applied aggregation and summarization. Other policies, as grouping and delaying, should be studied in future simulations to see how they affect scalability.

When the number of group members, or the frequency to receive summary events grows very large, simulations show that observers may compromise the group scalability. In these situations additional policies for event transformation may be required.

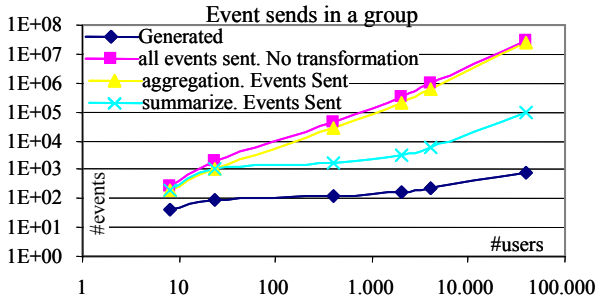

Figure 4. Events sent in larger groups

We can conclude that WWG with the application of event processing policies scales well as the number of members grow (mostly passive and observers). Work is under way to prove the scalability as the number of groups increases, but out intuition is positive because there are limits on the number of groups every person may belong to [10]: his social network.

6. RELATED WORK

The design of the following systems have inspired some of the WWG features. BSCW [2] provides a collaboration environment but the server is not distributed and does not consider local events. IMAP provides the idea of several modes of operation for different connectivity situations. Directory services provide the idea of network accessible user and group configuration information. WebDAV provides the idea of ways to extend http to support publication of documents. Distributed storage is provided for file systems (CODA, Ficus, Freenet) or distributed message/event systems (Usenet News). Siena is an event distribution system for wide-area networks concerned with scale.

6.1 Collaboration Environments: Bscw

The BSCW (Basic Support for Collaborative Work) system [2] is a Web-based application for collaborative information sharing based on the metaphor of shared folders as a repository for group information. BSCW provides awareness information about all the objects within the system (events).

Two limitations: 1) it does not take into account events result of local actions, 2) it can be used from any web browser, but it is a centralized system with one single database; it is not replicated: objects may not be at the proximity of the user. It does not work well for distributed groups, the server is a single point of failure, and the user suffers from network degradation or failures as distance increases.

6.2 Client manipulation of remote objects: Imap

IMAP (Internet Mail Access Protocol) allows a client to access and manipulate electronic mail messages on a server functionally equivalent to local mailboxes [5]. IMAP has three modes of operation that are desirable to our system:

- Offline: messages are delivered to a server and a client machine periodically connects to the server and moves (deletes from server) to the client all new messages. Thereafter, message processing occurs at the client machine.
- Online: messages are left on the mail server and manipulated remotely by mail client programs.
- Disconnected: a mail client connects to the mail server, makes a "cache" copy of selected messages, and then disconnects from the server, later to reconnect and resynchronize. The user operates "offline" on the cache.

Online and disconnected operation complement each other and one may alternate between them; they rely on a principal copy at the server and a cache at the client.

6.3 Directory service: ACAP and LDAP X.500

Both the ACAP (Application Configuration Access Protocol) [26] and LDAP-X.500 (Lightweight Directory Access Protocol) [22] services provide mechanisms to store and manipulate generic name-value pairs of information one or several replicated remote servers. They also provide ways to locate and access that information from remote locations.

6.4 HTTP Extensions: WebDAV

The Web Distributed Authoring and Versioning (WebDAV) [8] protocol allows users to collaboratively author their content directly to an HTTP server, allowing the web to be viewed not just as a read-only medium, but as a writeable, collaborative medium [12]. It provides facilities to HTTP for concurrency control, namespace operations, and property management.

WebDAV is an example of how a request/response protocol can be extended with additional methods and the exchange of structured data (xml documents). This extension mechanism may be the basis for the interface between user agents and repository agents.

6.5 Distributed Storage Systems: CODA, FICUS, Freenet

Coda [19] and Ficus [13] are general-purpose replicated file systems intended to facilitate distributed collaboration in a highly reliable and scalable fashion.

Both file systems allow updates so long as at least one replica of a data object is available (single-copy availability).

When conflicts do occur, they are reliably detected. Most conflicts are resolved automatically based on an understanding of the semantics (e.g. directories and replica location information).

While Coda has clients and file servers, in Ficus each machine, including workstations, portable computers and servers, should be empowered with full function so far as replication, file service, and reconciliation are concerned. In this sense, all machines are peers.

Freenet [4] has been recently proposed as a peer-to-peer, completely decentralized, network designed to allow the distribution of information over the Internet in an efficient manner, without fear of censorship.

It will provide an information publication system similar to the World Wide Web. Unlike the Web, information on Freenet is not stored at fixed locations or subject to any kind of centralized control. Freenet is a single world-wide information store that stores, caches, and distributes the information based on demand. This allows Freenet to be more efficient at some functions than the Web.

The CODA, Ficus and Freenet service is close to the service provided by a network of repository agents in WWG.

6.6 Distributed message systems: Usenet News

Usenet [24] is a distributed bulletin board system, built in the eighties as a logical network on top of other networks and connections. By design, messages resemble standard Internet electronic mail messages.

Messages generated at a site are sent to the site's ''neighbors'' who process them and relay them to their neighbors, and so on by a "flooding" algorithm. It propagates messages but it does not provide consistency guarantees.

Messages can be assimilated in many cases to events, forming something similar to a network of meta-information agents.

6.7 Event Distribution Systems: Siena

Siena (Scalable Internet Event Notification Architecture) [3] is a research project aimed at designing and constructing a generic scalable event service. It supports the idea of components that interact with events to inform of a change in their internal state or to request services from other components.

It has been designed to work on a wide-area network, for highly distributed applications that require a fine-grained interaction.

We are currently evaluating the suitability of the Siena model and the current Siena prototype for our purposes. In our approach, support for asynchronous distribution of events is a key issue.

7. WORK IN PROGRESS

Research issues that have yet to be clarified in the short term are:

- Specify the mechanisms for event propagation. Improve the propagation, routing, aggregation, transformation of awareness information optimizing the number of events, and the cost of distribution.
- Specify protocols and mechanisms for the replication of objects at repositories.
- Experiment with the current prototype in a real educational setting to validate the viability of the WWG infrastructure and WWG based applications.

7.1 Prototype

We have a first version of a WWG prototype. It implements a simplified WWG environment and a group browser as a WWG application. It has been implemented in Java. It is being useful to refine the specification of the protocols, as an extension of HTTP and WebDAV, for the exchange of events expressed in XML format. Siena [3] is being used as the event propagation infrastructure. Several WWG servers may be started at different locations. Every server contains one metainformation and repository agent. They are connected in a tree topology. Each group browser may connect to different metainformation agents and move from one to another.

This prototype has been used in very small scale validation experiments, but now we plan to use the prototype in a real cooperative work project to get feedback from real use (more information available at: http://www.upcnet.es/~acm1**)**

As can be seen on the next screenshot, the user agent deals with different groups and presents the documents and folders of the selected group.

Figure 5. Document view of a group

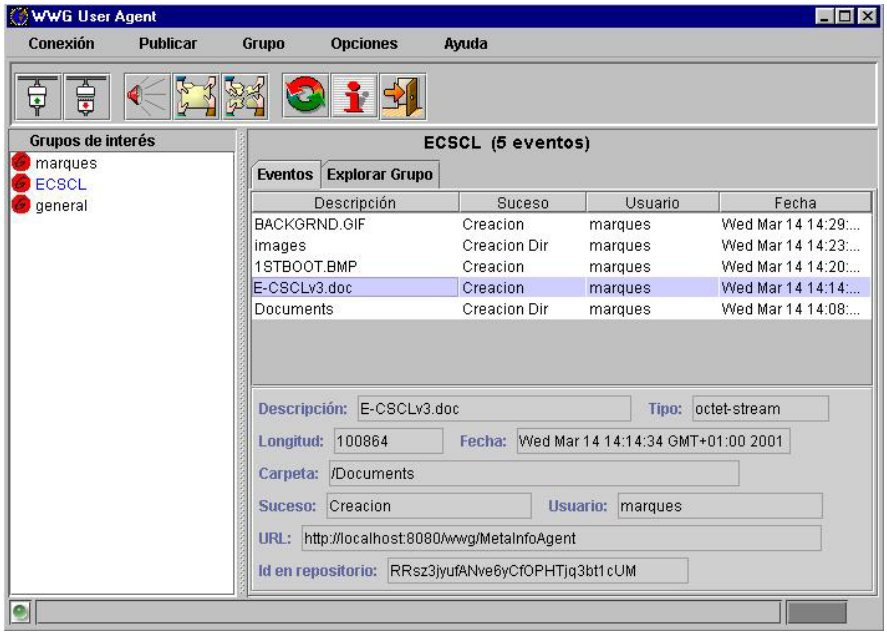

Figure 6. Event history from a group and details from one event

In the above screenshot, a list of events from a group is presented. From those events, a creation event is selected and all the details of the event are presented (kind of event, who generated it, location of the object, etc.)

7.2 Simulator

A simulator is being developed. The purpose of implementing it is to prove the scalability of WWG and as a way to easy refine and essay the WWG architecture and protocols.

8. CONCLUSIONS

The WWG architecture is intended to support collaborative learning (collaboration) among people pertaining to groups in wide-area networks.

We have designed an architecture that incorporates many features from existing systems. The resulting infrastructure provides a large number of services or components where collaborative applications may be easily built and integrated with each other.

The WWG infrastructure may be useful to extend existing centralized systems such as BSCW that give support for small to medium scale groups, but it may also be an important improvement for large scale groups now using primitive tools not adapted to collaborative learning such as mailing lists or Usenet News.

Initial work shows the viability of WWG, but work with the simulator and the prototype is under way to demonstrate and optimize their scalability, evaluate how awareness is supported, investigate ways to collect, transform, and present events to group participants in a way that is appropriate to them, further describe and specify the operations and information exchanged, and perform real world experiments to be able to validate the architecture.

9. ACKNOWLEDGMENTS

We are especially grateful to Antonio Castán for the development of the WWG prototype, as a very successful master thesis project.

We are grateful to the anonymous reviewers for their comments that helped to strengthen the article. This work has been partially supported by the CICYT- Spain.

10. REFERENCES

[1] Ariadne Project (Alliance of Remote Instructional Autoring and Distribution Networks for Europe), CE. 1996-2000. URL: http://ariadne.unil.ch/

[2] Bentley, R., Horstmann, T., Sikkel, K. and Trevor, J., Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System, in *The World Wide Web Journal: Proceedings of the 4th International WWW Conference*, Issue 1, December 1995, pp 63-74. ©O'Reilly.

[3] Carzaniga A., Ronsenblum D. S., Wolf, A. L., Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service, in *Nineteenth ACM Symposium on Principles of Distributed Computing (PODC2000)*, July 2000.

[4] Clarke, I. A Distributed Decentralised Information Storage and Retrieval System, Unpublish Technical Report *Edinburgh University*, Scotland, June 1999. http://freenet.sourceforge.net/

[5] Crispin, M., rfc 2060: INTERNET MESSAGE ACCESS PROTOCOL – VERSION 4 rev1, IETF - University of Washington, December 1996. urn:ietf:rfc:2060.txt, http://www.imap.org

[6] Daradoumis, T. and Marquès J.M. A Methodological Approach to Networked Collaborative Learning: Design and Pedagogy Issues. In: Proceedings of the 2nd International Conference on Networked Learning. Lancaster University, England, April 17-19, 2000. http://collaborate.shef.ac.uk/nlpapers/daradoumis-p.htm

[7] Daradoumis T., Xhafa F., and Marquès J.M. A Methodological Framework for Project-based Collaborative

Learning. Internal report in http://campus.uoc.es/~grc0_000228_web/Papers/TDP.doc

[8] WebDAV "Web-based Distributed Authoring and Versioning", URL: http://www.webdav.org

[9] English, S, Yazdani M, Computer Supported Cooperative Learning in a Virtual University, in Journal of Computer Assisted Learning, Volume 15 Issue 1; March 1999.

[10] Garton, L., Haythornthwaite, C., Wellman, B., Studying Online Social Networks, in Journal of Computer-Mediated Communication, 3 (1), 1997.

[11] Golding, R.A., Weak-consistency group communication and membership, Ph.D. thesis, published as *technical report UCSC-CRL-92-52. Computer and Information Sciences Board*, University of California, Santa Cruz, December 1992

[12] Goland, Y. Y., Whitehead, Jr, E. J., Fizi, A., S.R. Carter, and D. Jensen. HTTP Extensions for Distributed Authoring -- WEBDAV, *RFC 2518*, Microsoft, U.C. Irvine, Netscape, Novell, 1999.

[13] Guy R. G., Heidemann J. S., Mak W., Page T., Popek G. J., Rothmeier D., Implementation of the Ficus replicated file system, in *USENIX Conference Proceedings*, pages 63-71. USENIX, June 1990.

[14] Hiltz, S. R. and Wellman B., Asynchronous Learning Networks as a Virtual Classroom, in Communications of the ACM 40(9): 44-49

[15] Instructional Management System,url: http://www.imsproject.org

[16] Jarvenpaa, S.L., Leidner, D.E., Communication and Trust in Global Virtual Teams, in Journal of Computer-Mediated Communication, 3 (4), 1998.

[17] Johnson, D, Johnson, R, Smith K, Active Learning: Cooperation in the College Classroom, ISBN 0-939603-14-4, 1998.

[18] Johnson, K. L., Carr, J. F., Day, M. S., Kaashoek F., The Measured Performance of Content Distribution Networks, in *5th International Web Caching and Content Delivery Workshop*, Lisbon (Portugal), May, 2000.

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

[20] GIB: Light-weight Reliable Multicast Protocol, URL: http://webcanal.inria.fr/lrmp/

[21] IEEE Computer Society Learning Technology Task Force (LTTF), URL: http://lttf.ieee.org/

[22] *Open LDAP Consortium*, URL: http://www.Openldap.org.

[23] Prinz, W., NESSIE: An Awareness Environment for Cooperative Settings, in Proceedings of the ECSCW '99 Conference.

[24] Salz R. InterNetNews: Usenet transport for Internet sites, in *Summer '92 USENIX Conference* (San Antonio, TX, June 1992), 93-98.

[25] W3C Architecture: Extensible Markup Language (XML), URL: http://www.w3.org/XML/

[26] Wood, D., Programming Internet Email, in *Mastering Internet Messaging Systems*, Chapter 12, O'Reilly, 1999.

[27] Yu, H., Breslau, L., Shenker, S. A Scalable Web Cache Consistency Architecture, in Proceedings of the SIGCOMM'99 Conference.