WWG: a Wide-Area Infrastructure for 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: 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: synchronous awareness for asynchronous work.

The design issues, requirements and the resulting architecture are presented. WWG is based on a multicast mechanism for event distribution with meta-information agents responsible for the dissemination and transformation of events, repository agents responsible for the storage of group information and use agents responsible for the representation of users (sources and sinks of events).

Keywords
CSCL environment, event distribution, Internet-scale distributed systems, Virtual groups.

1. INTRODUCTION

Group learning 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. The period of intensive learning that occurs at the school and the university is a great opportunity to acquire team work skills. These skills will be key to success in the professional activity.

In addition, the learning process is increasingly influenced and mediated by computers, and a fast growing number of universities are beginning to offer virtual campuses to support distance learning, because many students and sometimes professors are in remote locations, they have temporal restrictions due to overlapping activities, or self pace learning is preferred.

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

This is the experience of the authors 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. Furthermore, we have acquired experience from cooperation with IEARN (http://www.iearn.org) a world-wide network of primary and secondary schools working with children from more than 20 countries in cooperative cross cultural projects.

Since the beginning of the Computer Sciences studies in the UOC (1997) the learning through working in group has been a main issue. Several experiences [DAR00] [DAR01] have enlightened the design of our infrastructure. In one hand, 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. On the other hand, learning in group needs some extra awareness information to know what the other members of the group are doing, information not needed in individual learning. 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.

The extension of learning activities from a campus scale to Internet scale 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 groups spread over the Internet.

The role of the tutor on those environments differs a lot. It goes from very active involvement where he organizes the interactions, observes, acts as another member of the group and at the end assesses the overall work, to the case in which the tutor only supervises the group and grades the students. The degree of involvement depends on the kind of activities. In any case, the tutor needs information about what every group member is doing and has done. It requires special computer support to refine and abstract what is happening in the group. While members of a group could be interested on who has read, wrote or modified every document, the tutor will be interested in how groups are working, the kind of contributions done by every student, etc.

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 [ARI00], IMS [IMS00] or the Learning Task Force at IEEE [LTT00]. 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 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. Participants may need to transform (summarize, condense) events, to give the required information to group members with diverse degree or mode of participation. This is the case for tutors, teachers, professors, assistants, supervisors, moderators, evaluators, etc.

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

The scenario as presented before is large, diverse and it has been studied before. 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+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).

3. REQUIREMENTS

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.

4. 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. This is provided by the event distribution mechanism.

WWG is intended for situations where users get virtually synchronous information (equivalent to real-time information but relaxed to scale better and save resources) about the actions that occur on the system. In terms of system design, synchronous event distribution allows us to do the following assumptions:

- Consistency through events: virtual synchrony 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 provide “sense of immediateness”: event distribution provides information about what is happening now in the system i.e. in the groups of interest. Getting the actions done within the group a short time (virtually at the same time) after occurrence allows the members to figure out the evolution of the group.
- Events provide “maximum information”: when a learning or working activity is done in groups 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 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.

5. 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 the action processed or to get the information about the action distributed to other members of the group, in form of an event. 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 (documents, discussions, events, users, groups, folders, etc). 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.

User agents (representatives of people participating on one or several groups, responsible for the exchange of events between a individual and the group) and repository agents (responsible for the efficient storage of group history, state and objects) are separated and interrelated by an intermediate layer in charge of the distribution of events between these agents (repository or user agents).

Traditional applications for information dissemination are based on client-server model. In the “pull” mode, the client polls the server to get event information. The client has to know which server has the right information and server replicas have to be consistent. The client also has to check the server periodically to get new events and to refresh frequently updated information. In the “push” model clients sign up for information to be sent directly to them, rather than fetching it by themselves. That model has limitations at Internet scale with a large number of groups with few participants each.

In our proposal the event distribution layer sits between user agents and repository agents, and is composed by **meta-information agents**, in charge of efficient distribution of information (events) generated by the users and the system. Meta-information agents have *passive functionality* (efficiently routing and distributing event information to interested agents, but also filtering, aggregating and transforming events), and *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).

5.1 A unicast+multicast architecture

The WWG network is composed by a set of coordinated computers running one, two or three of the following functions: user agent, meta-information agent, or repository agent.

User agents represent the end user on WWG, and interact directly with the user (in the same machine or remotely), with a close meta-information agent and with one or several repository agents.

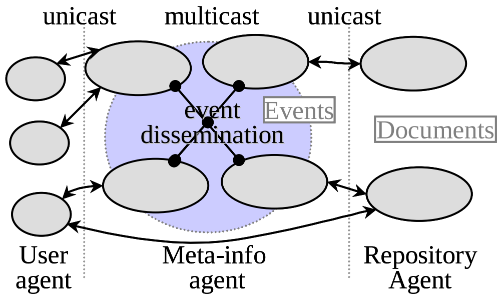
Event distribution by meta-information agents

The event distribution layer is responsible for efficiently offering, collecting and distributing event information among agents:

- Events have to be delivered as soon as possible to all interested parties (interactive delivery, virtually synchronously).
- The volume of generated messages is kept to a minimum (optimize the use of the network by aggregation).
- Events have to arrive to every destination interested (to all, in the right order).

A reliable multicast transport (e.g. LRMP [LRM99] or a higher level infrastructure for event distribution such as Siena [CAR00]) is an appropriate model for the distribution of events among meta-information agents.

The exchange of information between a user agent and a meta-information agent or between a repository agent and a meta-information agent is done using a unicast reliable transport protocol (TCP). In both cases unicast is appropriate for an exchange of particular information between a pair of mutually known agents.



The combined use of unicast and multicast protocols allows an efficient use of the network: a) when an event is generated, it goes by unicast between the user agent and a meta-information agent; then it is efficiently multicasted to the interested meta-information agents; and then it is sent by unicast to a user agent interested on that event.

Efficient distributed consistency protocols (application or network level multicast) for meta-information and content have been proposed in [YU99] and [GOL92]. In addition, [YU99] and [CAR00] demonstrate the scalability of a solution based on application or network level multicast without a central authority.

Events can be expressed as structured messages: for instance as an XML vocabulary [W3C00] with structured data eventually including objects by value or reference.

Network storage by Repository Agents

Repository agents cooperate to provide distributed and/or replicated network storage for objects. Group members should have transparent access to their objects with a reasonable quality.

Event information is very dynamic and abundant in any collaborative setting, and that is clearly useful for user agents to be aware of the progress of groups, but that is also useful for repository agents to decide where objects have to be located.

Event distribution (meta-information agents) should be separated from network storage (repository agents). We have observed in BSCW [BEN95] log files that the number of active participants in groups is on the order of *O(log N)*, being *N* the number of participants. While all *N* participants will have to be aware of the progress of the group (i.e. receive events), only active participants will require a good quality access to documents and other objects at repository agents.

6. HOW IT WORKS

The WWG infrastructure provides a framework that support the event distribution assumptions described in sections 4 and 5. Applications that require asynchronous collaboration in group are built in top of that framework. Those applications have to implement an interface to the user agent and another interface to the repository agent respectively through an API.

When an event is generated and passed to the user agent, WG guarantees the application that the event will arrive to a repository agent and to other members of the group by means of meta-information agents.

WWG is responsible for deciding the best location and the number of replicas of every object or event. Those decisions are taken transparently to the application and are done by the different agents that collaborate within the system. Applications benefit from features of the WWG system without having to deal with their implementation.

WWG supports three modes of operation:

- Connected: the user agent has a permanent connection to a meta-information agent. Every operation is immediately propagated, and every event in WWG is notified synchronously to the user agent.
- Disconnected: the user agent saves locally the required group information, and then disconnects from its meta-information agent. While disconnected work, operations are applied locally. When the user reconnects a

resynchronize process is started. That resynchronize process must take into account any eventual update conflict.

- Offline: This mode is a combination of the above. The user keeps connection with its meta-information agent. It operates over a local copy, and any incoming or outgoing event is delivered asynchronously.

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 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 of the events generated. It is not the aim of that paper to study those policies, nevertheless we present three possibilities: aggregation (when 10 actions occur in an object, send an unique event indicating that 10 actions have occurred), grouping (when several events goes to the same destination, send all of them in the same message) and delaying (when a lot of events are generated, wait a little while to send them).

Regarding to users, a similar problem occurs. Users have a finite capacity of processing events [DAR01]. A user, depending on the number of groups to which he belongs, 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 working*; or *in this group every member is in charge of at least one document*.

When objects are used by different people some conflicts may appear. In an asynchronous environment such as WWG most of the conflicts can be avoided by careful choice of some design alternatives. Even that, conflicts are still possible and the system must be able to solve them. WWG provides an special kind of event, the conflict events, 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 [KIS92]). Conflicts and conflict resolution has been studied in a separate research report.

7. RELATED WORK

The design of the following systems have inspired some of the WWG features. BSCW [BEN95] 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.

Collaboration Environments: Bscw

The BSCW (Basic Support for Collaborative Work) system [BEN95] 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 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.

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 [CRI96]. 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.

Directory service: ACAP and LDAP X.500

Both the ACAP (Application Configuration Access Protocol) [WOO99] and LDAP-X.500 (Lightweight Directory Access Protocol) [OLD00] 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.

HTTP Extensions: WebDAV

The Web Distributed Authoring and Versioning (WebDAV) [DAV00] 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 [GOL92]. It provides facilities to HTTP for concurrency control, namespace operations, and property management.

WebDAV may be the basis for the interface of user agents and repository agents.

Distributed Storage Systems: CODA, FICUS, Freenet

Coda [KIS92] and Ficus [GUY90] 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 [CLA99] 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.

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

Distributed message systems: Usenet News, Freenet

Usenet [SAL92] 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.

Event Distribution Systems: Siena

Siena (Scalable Internet Event Notification Architecture) [CAR00] 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.

8. WORK IN PROGRESS

A prototype implementation is in development. It will be useful to refine the specification of the protocols, as an extension of HTTP and WebDAV. In addition, we plan to do use the implementation for field studies with real groups.

Research issues that have to be clarified are:

- Decide the mechanisms for event propagation in each mode of operation (connected, disconnected, offline).
- Select the best protocols for the replication of objects at repositories.
- Research on mechanisms to decide the number and location of replica for the objects of every group.
- Improve the propagation, routing, aggregation, transformation of awareness information optimizing the number of events, and the cost of distribution.
- Evaluate the impact of different event policies (the kind of events collected and aggregation policies used) on the scalability of the architecture.

9. CONCLUSIONS

The WWG architecture is intended to support 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 will provide 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 is under way to demonstrate and optimize their scalability, evaluate how awareness is supported, describe the operations, and build the system to be able to get feedback from real use.

10. ACKNOWLEDGMENTS

This work has been partially supported by the CICYT-Spain.

11. REFERENCES

[ARI00] Ariadne Project (Alliance of Remote Instructional Autoring and Distribution Networks for Europe), CE. 1996-2000. URL: http://ariadne.unil.ch/

[BEN95] 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.

[CAR00] 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.

[CLA99] Clarke, I. A Distributed Decentralised Information Storage and Retrieval System, Unpublish Technical Report *Edinburgh University*, Scotland, June 1999. http://freenet.sourceforge.net/

[CRI96] 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

[DAR00] 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

[DAR01] Daradoumis T., Xhafa F., and Marquès J.M. A Methodological Framework for Project-based Collaborative Learning. Internal report available in http://campus.uoc.es/~grc0_000228_web/Papers/TDP.doc

[DAV00] WebDAV "Web-based Distributed Authoring and Versioning", URL: http://www.webdav.org

[GOL92] 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

[GOL99] 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.

[GUY90] 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.

[IMS00] Instructional Management System, URL: http://www.imsproject.org

[JOH00] 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.

[KIS92] Kistler, J.J., Satyanarayanan, M., Disconnected Operation in the Coda File System, in *ACM Transactions on Computer Systems*, Feb. 1992, Vol. 10, No. 1, pp. 3-25 http://www.coda.cs.cmu.edu/

[LRM99] GIB: Light-weight Reliable Multicast Protocol, URL: http://webcanal.inria.fr/lrmp/

[LTT00] IEEE Computer Society Learning Technology Task Force (LTTF), URL: http://lttf.ieee.org/

[OLD00] *Open LDAP Consortium*, URL: http://www.Openldap.org.

[SAL92] Salz R. InterNetNews: Usenet transport for Internet sites, in *Summer ’92 USENIX Conference* (San Antonio, TX, June 1992), 93-98.

[W3C00] W3C Architecture: Extensible Markup Language (XML), URL: http://www.w3.org/XML/

[WOO99] Wood, D., Programming Internet Email, in *Mastering Internet Messaging Systems*, Chapter 12, O’Reilly, 1999.

[YU99] Yu, H., Breslau, L., Shenker, S. A Scalable Web Cache Consistency Architecture, in Proceedings of the SIGCOMM’99 Conference.