Hybrid Consistency Framework for Distributed Annotation Records in a Collaborative Environment

Ahmet Fatih Mustacoglu[1,2], Geoffrey C. Fox[1,2]
[1]*Community Grids Lab, Indiana University, Bloomington, IN, 47404, USA*
[2]*Department of Computer Science, Indiana University*
{amustaco, gcf}@cs.indiana.edu

ABSTRACT

We describe a novel hybrid consistency framework that supports collaboration and maintains consistency among Distributed Annotation Records, which are replicas of the same document, kept at various web-based annotation tools. There are issues in semantics of annotation tools. Each annotation tool stores different and limited metadata, has different rules for tags, and does not provide timing information for the updated records. As a result of these, the same documents can be updated inconsistently with unknown precise timestamps and spread around in existing annotation tools with different versions. Moreover, communication of annotation tools with each other is also very limited through various forms and this also contributes to inconsistencies among Distributed Annotation Records. To deal with these major shortcomings, this paper introduces the notion of "hybrid-consistency framework", which allows collaboration on shared records and maintains the consistency among Distributed Annotation Records held at various annotation tools, through pull and push based consistency mechanism. We discuss the overall design, architecture and the components of the hybrid consistency framework, and provide a working prototype implementation and a roadmap of the future work in this research.

KEYWORDS: Collaboration, Hybrid consistency, Web 2.0, Annotation Tools, Distributed Annotation Records, Events, Tagging.

1. INTRODUCTION

Web 2.0 can be defined as the second generation of Web applications and network-enabled stateless services [1, 2]. It is generally characterized as building on the set of key concepts: (1) REST (Representational State Transfer) Services; (2) rich user interfaces via JavaScript, AJAX, and JASON; (3) online communities and social networks; (4) widgets, gadgets, and badges. There are various types of online community tools aiming at fostering online collaboration and sharing between users and communities. The most popular examples of these tools include Blogs (blogger.com, Google Blog) [3], Wikis (Wikipedia [4], WikiWikiWeb [5], Wikitravel [6]), Social Networking Tools (MySpace [7], LinkedIn [8]), Social Bookmarking Tools (del.icio.us [9], Flickr [10], YouTube [11]), Syndication Feed Aggregators (Netvibes [12], YourLiveWire [13]) and other related tools.

The term "Web 2.0" is now getting more and more popular, and it is representing this wave of new Web-based tools and the use of technologies. This change is also very obvious in the domain of scientific research, with the recent creation of a number of online tools that enable the annotation and sharing of scientific content, such as CiteULike [14], Connotea [15], and Bibsonomy [16]. One of the famous annotation website is del.icio.us and is sometimes referred as Delicious. It is a web-based tool and it enables users to annotate and share URLs. There are other numbers of annotation tools and they support annotation and sharing of a variety of resources, such as videos (YouTube), goals (43things [17]), photos (Flickr), and books (LibraryThing [18]). Especially there exist Web-based online tools focusing on the annotation of scholarly publications such as Connotea, CiteULike, and Bibsonomys. The fundamental service provided by these Web-based annotation tools is the capability that allows users or communities to easily annotate their favorite resources (videos, photos, URLs, or citations) by using the keywords called tags and to share their tagged content with other users.

While the numbers of annotation tools are increasing rapidly, each of them having their own structure, design, interface, format of their holding and very few examples exist of any of these being able to communicate in some form with other annotation tools. These tools and services store annotations and metadata in their system. Users of these tools and services can collaborate, share, update or modify descriptive fields of their entries such as title,

description, or tag, etc. Today various online collaboration tools, peer to peer systems and internet have generated multiple sources of information about the same data. These multiple sources of information are all dynamic, and each of them has value but no one has total value. As a result of this, multiple copies of a same object can be in different places, and users of these systems suffer from having multiple copies of the same record in different versions due to different metadata storage in each annotation tool. In addition, annotation tools do not provide timing information for an updated record, and this can also lead to inconsistency for replicas of the same Distributed Annotation Record (DAR), which is a collection of metadata stored at an annotation tool, once the DAR get updated. In order to cope with these shortcomings, there is a need for architecture or a framework, which supports collaboration, to reconcile these dynamic possibly inconsistent sources of metadata about the same DAR located at different annotation tools in a consistent manner.

We propose a hybrid consistency framework to maintain consistency for each replicas of the same DAR held on several annotation tools in a collaborative environment. The ideal approach to reconcile different sources of annotation and metadata for DARs is to have an event-based model [19] to keep track of changes to documents and metadata while providing our proposed consistency framework around it. In our proposed solution, we keep primary copy of each DAR with extra metadata fields in our relational database, where the primary copy of each DAR can be collaborated and updated consistently, and we provide a hybrid consistency framework to maintain consistency between all replicas of DARs stored at various annotation tools and a primary copy of each DAR.

This paper discusses the hybrid consistency framework for the replicas of the same DARs maintained at different annotation tools and expounds its implementation and integration with Semantic Research Grid [20] system. The rest of this paper is organized as follows: Section 2 provides a discussion of the consistency criteria. Section 3 describes the design philosophy. Section 4 gives the details of our proposed hybrid consistency framework architecture. Section 5 explains the architecture components. Section 6 presents a prototype implementation of our proposed framework. Section 7 discusses the future work in this research.

2. CONSISTENCY CRITERIA

The consistency enforcement issue is all about ensuring that all copies of the same data are to be the same. There exist some approaches to maintain consistency are discussed in [21-26] in detail. Tanenbaum [25] categorized consistency under two main category: (1) data-centric; and (2) client-centric. In data-centric approach, all copies of data are updated whether some clients is aware of those updates or not. In client-centric approach, consistency is maintained from a client's perspective. Client-centric consistency model allows copies of data to be inconsistent with each other as long as the consistency is ensured from a single client's point of view.

The implementation of the consistency models can be differentiated as primary-based protocols (primary-copy approach) and replicated-write protocols [25]. In primary copy approach, updates are executed on a single location, and propagated replicas from there, while in the replicated-write approach; updates can be originated from multiple locations. For an example, techniques for maintaining consistency in P2P networks: (1) Push: Owner-initiated Consistency. In this model, messages are propagated through the P2P overlay in push approach; (2) Pull: Peer-initiated Consistency mechanism. Individual peers polls the owner to figure out if a file is stale or not; and (3) Hybrid Consistency mechanism. Our approach enhances the popular consistency techniques, which had been originally designed for the distributed replicated systems, to be applied to DARs to maintain consistency among web-based annotation tools.

3. DESIGN PHILOSOPHY

Annotation tools are one of the major Web 2.0 applications. They basically provide their users with ability to: (1) enter a new record; (2) delete an existing record; (3) modify an existing record; (4) tag their record; (5) share the content of their records with other users. Data is being shared in these annotation tools are varied. However, the most common data shared for major annotation tools include tag, notes, description, title, comment and URL. The consistency concept arises when collaborated and shared data get updated with unknown timestamp. Providing consistency maintenance is a fundamental issue [21], and our research focuses on how to design a consistency framework to maintain consistency for each DAR, which are replicas of the same documents, held on those annotation tools. The design of such an environment should consist of group of annotation tools intended to be consistent with each other, and a main collaborative system, where a primary copy of each document from each annotation tools are collaborated and stored with additional metadata information into a relational database.

There are issues in semantics of annotation tools such as each annotation tool stores different metadata in their system and their rules for tag are also different from each other. Table 1 portrays the stored metadata comparison in Connotea, Citeulike, and Delicious annotation tools. One

Table 1. Stored Metadata Comparison in Annotation Tools

Stored Metadata	Citeulike	Connotea	Delicious
URL	✓	R	R
Title	R	✓	
DOI	✓	✓	
PMID		✓	
ISBN/ASIN		✓	
Reference Type	R	✓	
Authors	✓	✓	
Pub. Name		✓	
Volume No	✓	✓	
Issue No	✓	✓	
Chapter	✓		
Edition	✓		
Start& End Page	✓		
Pages		✓	
Year&Month& Day	✓		
Pub. Date		✓	
Date Other	✓		
Editors	✓		
Journal	✓		
Book Title	✓		
How Published	✓		
Institution	✓		
Organization	✓		
Publisher	✓		
Address	✓		
School	✓		
Series	✓		
Bibtex Key	✓		
Abstract	✓		
Display Title		✓	
Tags	†	R	✓
Tag Suggestions		✓	
Description		✓	R
My Work		✓	
Everyone's Tag	✓		
Privacy Settings	✓	✓	
Release date others		✓	
Priority of Records	✓		
Note	✓		✓
Comment		✓	

✓ = Supported, R = REQUIRED, † = Adds "no-tag"

major problem with annotation tools is that they do not provide precise timestamps for the updated records. As a result of this, data can be updated inconsistently with unknown precise time. In our proposed framework, we are supporting various metadata fields in our main database where we store the primary copy of each replicas of the same DAR with the timestamp information. Another fundamental issue of using annotation tools without their user interfaces is that annotation tools are lack of services or mechanisms to provide their clients with notification services for deleted, modified or entered new entries into their system. Hence there is no way to realize any changes in those systems unless modifications are done through their interfaces. The only way to identify any change in those tools is having a mechanism to go and check them periodically. We have designed our hybrid consistency framework to be able to: (1) run all the time for consistency enforcement; (2) communicate with integrated annotation tools periodically; and (3) collect the differences between supported metadata of each replicas of the DAR kept in each annotation tool and the primary copy of each DAR stored in a relational database (how to decide that two objects are referring the same digital entity/document is discussed in Duplicate Detection in section 4.1). Eventually, if there are any changes to any replicas of the same DARs in annotation tools, we can retrieve the latest updates by pulling them out from these tools, and apply them to update the primary copy and replicas of each DAR. Furthermore, users can collaborate on a primary copy of each DAR with each other by sharing the same document. And our hybrid consistency framework propagates updates made on a primary copy of a DAR to each annotation tool to reflect the changes in a consistent manner on replicas of it. As a result, we have designed to have a two way mechanism to maintain consistency among integrated annotation tools, where replicas of same document stored, and a primary copy of each DAR. We are going to give the details of our proposed consistency framework in Section 4.

4. HYBRID CONSISTENCY FRAMEWORK ARCHITECTURE

Our hybrid consistency framework has been designed to maintain consistency between DARs kept at annotation tools and a primary copy of each DAR. The hybrid consistency framework is a data centric consistency model, and it is based on the primary copy based consistency protocol approach. In our proposed framework, update propagations are carried out through pull and push based approaches. Push approach enforces strict consistency model on primary copies of DARs. In strict consistency model; whenever updates occurred on a primary copy of a DAR, they are being propagated immediately to each annotation tool to update replication of same document referred as DARs on their site.

However, pull approach is a time-based consistency control approach [27]. We are periodically checking DARs from each annotation tool for any updates. If there is any, then we are pulling them out. Finally, we are applying them onto the primary copy of each DAR, which is stored in a relational database with additional metadata. We have also developed a rollback mechanism to ensure consistency. It basically allows users to rollback to a previous state at any time. We are going to explain rollback mechanism in detail in Section 5.7.2. Figure 1 represents the overall architecture of our proposed Hybrid Consistency Framework. Explanations of the some key concepts are given in the following sub-sections and architecture components are given in section 5.

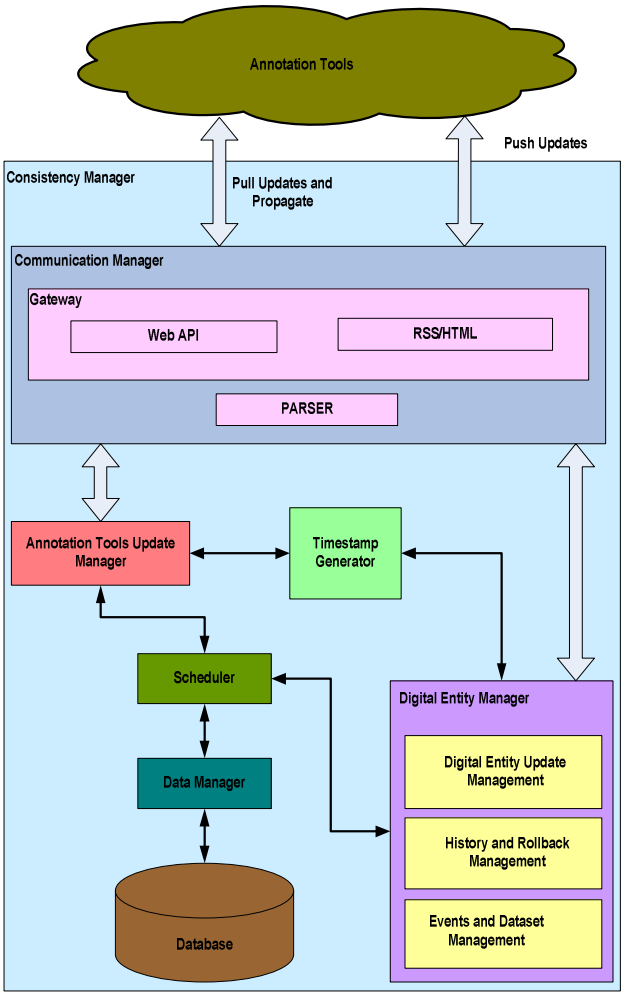

Figure 1. Hybrid Consistency Framework Architecture

4.1 Duplicate Detection

In our proposed research, it is crucial to identify if two records are representing the same document or not. During the consistency maintenance process; each record is retrieved from annotation tools, and processed to form a digital entity, which is a collection of metadata representing a document [19], as explained in Communication Manager in section 5.2. We compare each digital entity with all the primary copies from a repository by using our duplicate detection algorithm to find its matching primary copy if it exists in the database. Our duplicate detection algorithm works based on the available metadata fields of a DAR including URL, title, authors and publication venue. Furthermore, a threshold value needs to be defined for the comparison algorithm such as 0.95. According to our duplicate algorithm result, if a digital entity does not exist in the database, then this is a new update and it is inserted into the system as a major event [19], and propagated to all replicas, which are stored at the integrated annotation tools. If we can find a matching primary copy for this digital entity, then our Annotation Tools Update Manager compares these two digital entities to identify that whether any metadata field of the digital entity is updated or not. If there are any available updates, then they are processed by Annotation Tools Update Manager as explained in section 5.3.

4.2 Handling Concurrent Updates

The main goal of concurrency control is to allow processes to work on a shared data simultaneously in a way that the shared data left in a consistent state after modifications. The consistency is maintained by giving processes access to shared data in a specific order in which the final result is the same as if all processes had run sequentially. Concurrency control algorithms can generally be categorized based on how the read and write operations are synchronized. Synchronization can be provided through either mutual exclusion or ordering processes by using timestamp values [25]. In our proposed architecture, it is very rare to have concurrent updates on a same document. However, our policy for concurrent updates is to adopt Pessimistic Time Stamp Ordering [25] approach. In this approach, each process is assigned a unique timestamp value by using Lamport's algorithm. When concurrent updates occur on a shared data, data manager process the one with the lowest timestamp first. In our proposed research, concurrent updates occurrences are very seldom, and they may occur when:

I. A user uses an annotation tool's own UI to update a replica record and another user tries to update the same copy through its primary copy by using Digital Entity Manager;
II. Two or more users try to update a record through the primary copy by using Digital Entity Manager;
III. While consistency manager is working in the background to collect and process updates on

primary copies and replicas, another user tries to update a record via its primary copy by using Digital Entity Manager.

In the first case, we do not have much control how the annotation tool handles the coming concurrent update requests. In order to handle inconsistencies, annotation tools are supposed to notify our system once an update occurs in their system, but they do not have such functionality. Moreover, they are independent systems and we cannot lock them during the update executions to prevent inconsistencies. For example; let's assume that an update ($Update_{primary}$) coming from the replica's primary copy is executed before a user's update ($Update_{annotation}$) coming from the annotation tool's UI. Then $Update_{primary}$ will be lost. Since, $Update_{annotation}$ overwrites $Update_{primary}$. Let's think the other case that $Update_{annotation}$ is executed before $Update_{primary}$. Then, $Update_{annotation}$ will disappear due to the replacement of it by $Update_{primary}$. So, above cases are exceptions in real life domain, and we assume that updates from annotation tools' UI do not occur since we have already integrated these independent annotation tools into our system including their existing services with added extra capabilities [20]. In the second and third cases; data manager process the request with the lowest timestamp first according to our concurrent updates policy.

5. ARHITECTURE COMPONENTS

The detail explanation of the hybrid consistency framework architecture is given in the following sub-sections respectively.

5.1 Annotation Tools

Annotation tools represent the integrated annotation tools into our proposed hybrid consistency framework. Our model, works around these Web 2.0 tools to reconcile DARs from each annotation tool in a consistent way. In the current implemented version, we have integrated Delicious, Citeulike, and Connotea into our prototype system called Semantic Research Grid (SRG) [20].

5.2 Communication Manager

Communication manager transports the data between the computing nodes. It is responsible for retrieving or posting data from/to annotation tools though their gateways. It retrieves records from annotation tools via HTTPClient [28] native libraries by using: (1) Annotation tool's API and get the response in XML format. Data is then parsed by using DOM parser and XPATH [29]; or (2) HTTP GET, and POST methods resulting in getting the response in RSS or HTML format. In RSS type responses, documents are parsed by using DOM parser and XPATH, and in HTML type responses, data is parsed after cleaning faulty HTML by using JTidy [30] native libraries. Having retrieved and parsed documents, Communication Manager passes the organized data to Annotation Tools Update Manager explained in detail in 5.3. Updates are posted to annotation tools via: (1) annotation tools API; or (2) HTTP GET, POST methods through HTTPClient native library unless an annotation tool provides an API. Its modules are explained in the following sections respectively.

5.2.1 Gateway

Gateway is an interface between hybrid consistency framework and an individual annotation tool. Our hybrid consistency model communicates with annotation tools through their gateways. The communications are carried out through HTTP methods by using HTTPClient native libraries [28]. An individual gateway is created for each interacting annotation tool, which has its own communication structures.

5.2.2 Parser

Parser is a native library used for parsing responses coming from annotation tools. There are several parsers to utilize in XML processing. DOM parser is the most widely used one. It reads and validates the XML documents. If the document is valid, then it returns a document object tree. We can randomly access any element since each element is entirely kept in memory. As a result, it provides a very efficient navigation mechanism over the parsed document. On the other hand, its drawback is that it requires large amount of memory in order to hold the whole parsed document. Most of the major annotation tools provide their Web API and responses are like in XML format. So users can communicate with their services easily. In our prototype implementation (described in Section 6), we have used JDOM [31] parser as our parsing library. In some annotation websites, they do not provide a Web API for their services, and then their responses in either in RSS or HTML format. In order to communicate with those annotation tools, we have used XPATH [29] to retrieve the desired element of the document and JTidy native library [30], which is used for cleaning faulty HTML and provide a DOM interface to the documents that is going to be parsed.

5.2.3 Web API

Web API (Application Programming Interface) is a service for accessing data on annotation tools. Most of the major annotation tools provide their Web API and RSS feeds for an easy access to their data. Their Web API and RSS feed return a document in XML format, which can be parsed easily by using a DOM parser in our prototype implementation, to the requester. Hence, data from

annotation tools can be retrieved and modified via their Web API through HTTPClient tool by passing the necessary parameter to HTTPClient object.

5.3 Annotation Tools Update Manager

Annotation tools update manager is responsible for retrieving the updates from annotation tools periodically and applying the updates on the primary copy of each DAR. Its main duties are: (1) obtain the updates from annotation tools via Communication Manager; (2) applying each update on its primary copy stored in the relational database; (3) propagating the updates back to each annotation tools.

Obtaining updates from the documents coming from Communication Manager requires: (1) Finding the primary copy of each record by using duplicate detection algorithm; (2) Comparing each replica record with its primary copy to figure out modifications if there is any. After indentifying the updates, next step is to apply the updates on primary copy and disseminate updates to replicas. Integrated annotation tools do not support publish-subscribe mechanism forcing us to use unicast communication to propagate updates to replicas. However, any application that require and support publish-subscribe mechanism, then broker address and topic can be defined in a property file to provide updates via publish-subscribe mechanism by connecting to the broker and subscribing a topic.

5.4 Timestamp Generator

Timestamp generator provides a service to generate unique timestamp values to the requesting processes. These unique timestamps values are used for ordering processes to execute them once there is an update conflict on the shared data.

5.5 Scheduler

Scheduler is responsible for controlling concurrency. It determines that which process is allowed to pass to the data manager for read or write operation. Main duty of the scheduler component is to keep track of currently executing processes and their data in order to allow or keep waiting other processes on the shared data.

5.6 Data Manager

Data manager is responsible for executing read or write requests on a data item. Data manager is not concerned about what operations it is performing. It just executes the coming operation on a data item.

5.7 Digital Entity Manager

Digital Entity Manager is an umbrella name for a group of modules that contributes to a DAR management together. Its modules are: (1) Digital Entity Update Management; (2) History and Rollback Management; (3) Events and Dataset management. Details of each module are given in the following sections respectively.

5.7.1 Digital Entity Update Management

It deals with updates that are made directly on a primary copy of each DAR. Each update to a DAR consists of minor event(s) and dataset(s) [19]. Once an update made to a DAR, it becomes a minor event. Having dataset created from minor events, the changes are reflected in the database as events, which allow us to track the changes to a document. Furthermore, the updates are disseminated to annotation tools via the Communication Manager once they occurred.

5.7.2 History and Rollback Management

Using the mechanism described in [19], all the changes that have occurred to a DAR are stored in the user session as minor events [19]. They do not have any effect on the current value of the DAR unless minor events are used for creating a dataset. Once a dataset is created by using minor event(s), the dataset is applied to the DAR metadata during the latest DAR retrieval process.

To allow users to restore the state of the system to any previous state, we have implemented a module that allows users to view the history of each DAR and to undo any changes (rollback). In the history tool of the Digital Entity Manager, each DAR has an initial entry and a list of time-stamped datasets, which represents the changes made to the DAR if there is any. During the rollback execution; first a user selects and applies a time-stamped dataset. Second, the selected state of the dataset compared with the latest metadata of the DAR. Finally, the DAR is roll-backed to the selected state by unrolling the related events from the current version of DAR. Further details can be obtained from [19].

5.7.3 Events and Dataset Management

In our framework, an event is defined as a time-stamped action on a DAR. Our hybrid consistency framework identifies the events: (1) Minor Events that encapsulates the changes to a DAR; (2) Major Events that are represent an entry of a new DAR into the system or deletion of an existing DAR. A dataset consists of collection of minor events. Further details can be found in [19].

5.8 Survey of Technologies

In our implementation of hybrid consistency framework, we have used various technologies. Summary of the technologies [28-33] are represented in Table 2.

Table 2. Technologies

API	Purpose
JDOM	For parsing XML documents
Jakarta Commons HTTP Client	For handling HTTP communication
XPATH	For querying an XML document object
JTidy	For parsing HTML documents
Apache Axis	For creating Java Web Services
JAVA	For implementing the framework

6. PROTOTYPE IMPLEMENTATION

We have applied our proposed Hybrid Consistency Framework to Semantic Research Grid (SRG) system described in detail [20]. Our proposed framework has been implemented using Web Service Technology, and services can be accessed via SOAP calls. The SRG system has been designed based on the Web 2.0 technologies and it consists of tools and services for supporting collaborative Cyberinfrastructure [34] based scientific research. The SRG system integrates a number of existing online research tools (social bookmarking, academic search, scientific databases, journal and conference content management systems) and aims to develop added-value community-building tools that leverage the semantic analysis of DARs. Running instance of the SRG system can be accessed from project demo website [35].

7. FUTURE WORK

In the current implementation, users can only track consistency updates in our system via the history tool. A desired future of the system would be a tool for logging the consistency updates. It will allow users to see automatically applied updates for consistency enforcement and their status as well. We intend to do this improvement by creating a database table for keeping consistency updates and retrieving the data whenever users request to access history of consistency updates.

Another desired future work will be conducting various scalability and performance tests of our proposed hybrid consistency framework.

8. CONCLUSION

In this paper we discussed our proposed Hybrid Consistency Framework to maintain consistency among DARs stored at various Annotation Tools in Web 2.0 domain. We have also mentioned the implementation details of our proposed framework in SRG system. Furthermore, we described the current state of the development of the hybrid consistency framework and outlined some directions for future work.

REFERENCES

[1] P. Graham, "Web 2.0," 2005. Available from http://www.paulgraham.com/web20.html

[2] T. O'Reilly, "What is Web 2.0," 2005. Available from http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[3] Blogger web site. https://www.blogger.com/

[4] Wikipedia web site. http://en.wikipedia.org/wiki/Wiki

[5] WikiWikiWeb web site. http://c2.com/cgi/wiki

[6] Wikitravel web site. http://wikitravel.org

[7] MySpace web site. http://www.myspace.com/

[8] LinkedIn web site. http://www.linkedin.com/

[9] Delicious web site. http://de.icio.us

[10] Flickr web site. http://www.flickr.com/

[11] YouTube web site. http://www.youtube.com/

[12] Netvibes web site. http://www.netvibes.com/

[13] YourLiveWire web site. http://www.yourlivewire.net/

[14] CiteULike web site. http://www.citeulike.org

[15] Connotea web site. http://www.connotea.org

[16] Bibsonomy web site. http://www.bibsonomy.org

[17] 43things web site. http://www.43things.com/

[18] LibraryThing web site. http://www.librarything.com/

[19] A. F. Mustacoglu, A. E. Topcu, A. Cami, and G. Fox, "A Novel Event-Based Consistency Model for Supporting Collaborative Cyberinfrastructure Based Scientific Research," To appear in *Collaborative Technologies and Systems CTS 2007 in Technical Cooperation with The IEEE Computer Society*. Orlando, FL, USA, 2007.

[20] G. Fox, A. F. Mustacoglu, A. E. Topcu, and A. Cami, " SRG: A Digital Document-Enhanced Service Oriented Research Grid," in *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference*. Las Vegas, NV, USA, 2007, pp. 61-66.

[21] S. Chengzheng and C. David, "Consistency maintenance in real-time collaborative graphics editing systems," *ACM Trans. Comput.-Hum. Interact.*, vol. 9, pp. 1-41, 2002.

[22] L. Jiang, L. Xiaotao, S. Prashant, and R. Krithi, "Consistency Maintenance In Peer-to-Peer File Sharing Networks," in *Proceedings of the The Third IEEE Workshop on Internet Applications*: IEEE Computer Society, 2003.

[23] R. Jonathan, F. Sarah, and V. Sankar, "Consistency management for distributed collaboration," *ACM Comput. Surv.*, vol. 31, pp. 13, 1999.

[24] V. Jurgen, V. JiRgen, G. Werner, C. Li-Te, and M. Michael, "Consistency Control for Synchronous and Asynchronous Collaboration Based on Shared Objects and Activities," *Comput. Supported Coop. Work*, vol. 13, pp. 573-602, 2004.

[25] A. S. Tanenbaum and M. V. Steen, *Distributed Ssytems Principles and Paradigms*, 2002.

[26] G. Werner, V. Jurgen, C. Li-Te, and M. Michael, "Supporting activity-centric collaboration through peer-to-peer shared objects," in *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*. Sanibel Island, Florida, USA: ACM Press, 2003.

[27] L. Rui, L. Du, and S. Chengzheng, "A Time Interval Based Consistency Control Algorithm for Interactive Groupware Applications," in *Proceedings of the Parallel and Distributed Systems, Tenth International Conference*: IEEE Computer Society, 2004.

[28] JAKARTA COMMONS HTTP CLIENT. Available from http://jakarta.apache.org/httpcomponents/httpclient-3.x/

[29] XML Path Language (XPATH). Available from http://www.w3.org/TR/xpath

[30] JTIDY. Available from http://jtidy.sourceforge.net

[31] JDOM. Available from http://www.jdom.org

[32] WebServices-Axis web site. http://ws.apache.org/axis/

[33] JAVA Technology web site. http://www.java.sun.com

[34] D. E. Atkins, K. K. Droegemeier, S. I. Feldman, H. Garcia-Molina, M. L. Klein, D. G. Messerschmitt, P. Messina, J. P. Ostriker, and M. H. Wright, "Revolutionizing Science and Engineering Through Cyberinfrastructure. Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure," 2003.

[35] Semantic Research Grid (SRG) Project web site. http://gf16.ucs.indiana.edu:54569/SRGrid/