SYNCHRONA – Data Synchronization Between Different Internet Services

Nilufa Cassim[1], Harini Sirisena[1], Kapila Bogahapitiya[1], Aruna Jayasena[1], Gihan Dias[1], Chandika Jayasundara[2] and Hiraash Thowfeek[2]

[1]Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.
[2]Cinergix Pvt. Ltd, Sri Lanka.

Abstract — Synchrona research project is aimed at synchronizing data between multiple different Internet sites and services. It tries to address some of the issues related to data synchronization between varieties of social Internet services. In this paper we present an open source implementation of an extensible data synchronization platform for this purpose. In this paper Synchrona service is presented with some of the major applications built within for the service. Here we have described synchronization services for the following sites; Blogger, Facebook, Moodle and Twitter. However the service is made extensible via an API so that new synchronization engines can be included for other Internet sites and services. This project is user centric and will provide the average Internet user with the freedom to choose the manner in which they want their data synchronized by stating their preferences in a web browser plug-in. The browser plug-in will act as the user front end and pass user updates to the web service for synchronization. The system also includes real-time data synchronization between the users' friends in the form of notifications. Synchrona, in its implementation will focus mainly on the aspect of fuzzy data synchronization to ensure that user data is synchronized seamlessly between multiple sites without loss of meaning.

Index Terms—Data synchronization, Fuzzy synchronization, Internet Services

I. Introduction

Data synchronization can be considered as one of the essential process carried out today since most of our data have being spread among almost all of the data sources in the Internet. The physics of data management dictates that your data could be either consistent or highly available but never both the same time. Data synchronization plays a vital role in this dilemma [1].

As well as we need data of the sources being synchronized with data sources we have, most of our own information are also shared among various data sources such as Internet services, databases. Internet services are one of the major data sources that can hold a variety of information.

This paper discusses how data synchronization shall be conducted in service types such as social networks and information services. We also discuss about the Internet services we have considered, the aspects of synchronization, *fuzzy synchronization* application, the methodology and the end product specifications to the user.

II. Background

The need for implementing data synchronization between different Internet services has risen due to the arrival of Web 2.0[2]. With Web 2.0 comes an ever expanding arena of social networking websites, news sites and other user centric content sharing websites. As a result, an average Internet user has user accounts on multiple different websites. This phenomena has resulted in the emergence of standards such as OpenID[3] which allow users to login to one website they use and simultaneously be logged into many other websites he/she uses. In the same way that a user would prefer to login to multiple websites with one single login, a user would also sometimes prefer to update content of many of his/her web accounts with one single submission of data. This can be referred to as parallelization or synchronization of user updates across multiple sites [4]. Many of the existing implementations [5] that address this issue limit the type of data that can be synchronized to status updates.

III. CONTEMPORARY RESEARCH

This research aims at providing a solution for synchronization of a broader context of user content updates.

Traditionally data synchronization centers on synchronizing data between different devices, such as the desktop PC, notebook PC, mobile and other hand held devices. Given below are some of the accepted implementations for device data synchronization on which we conducted our preliminary research.

Microsoft ActiveSync is a synchronization architecture used for data synchronization between devices running Microsoft Windows systems. It provides support for synchronizing data between a Windows-based desktop computer and Microsoft Windows CE-based devices. Using time stamps and user preferences, the synchronization process tracks data changes on both computers, then transfers the appropriate data so that each machine has the most-recent versions. Outdated or unwanted data is discarded. [6]

OpenSync is an example of an open source project for device data synchronization. It is an ongoing effort to create a synchronization framework that will be a platform independent, general purpose synchronization engine utilizing modular plug-ins for content formats and different kind of connection types. OpenSync's modularity allows it to be extended easily to new devices and purposes without radically changing the architecture itself, allowing it to support a wide variety of devices used today and in the future. [7]

According to its website, —SyncML is the leading open industry standard for universal synchronization of remote data and personal information across multiple networks, platforms and devices." In reality, SyncML is the only industry standard for synchronization of PIM (Personal Information Manager) data. SyncML defines a basic client-server interface for exchanging personal data. Basically, any devices can act as a server or a client. The requirements for a server are much higher though and to avoid complete data confusion, this approach leads to one central server. [8]

IV. DESIGN AND IMPLEMENTATION

The design for the Synchrona system was started by considering the main functional requirements of the system; data retrieval, data processing and data synchronization. The procedure that was followed in designing the system to meet each of these requirements is described in detail below.

- *Data Retrieval*

One of the major constrains and requirement in data synchronization between Internet services is the real-time update retrieval from any given Internet service. To accomplish this we have been involved in research in variety of existing data synchronization techniques. One of the solutions we came up with is polling of the services. The idea was to retrieve updates by constant analysis of the data posted in each Internet service, there by identifying new updates. But this technique becomes inefficient when it comes to scalability of the system [9]. Polling of large number of services can reduce the performance of the system.

- *Data Processing*

Next primary research area, which plays the core role of Synchrona project, is to what methodology should be used when processing data. System retrieves data from variety of sources used for various purposes. As an example, system may receive data from Moodle, which belongs to Moodle journal update. Retrieved data has to undergo some sort of processing before making any update using this journal update and this processing may take different forms depending on the service being updated. This is the point at which we came across the term —fuzzy synchronization". Converting data, into a form that other services can make use of is the major research area of Synchrona project.

- *Data Synchronization*

Updating of the variety of Internet services is another important process in synchronization. Here we have considered existing methods of data synchronization.

Initially we were interested in deriving a method similar to what is being used by feed aggregates. We directed our research towards synchronization technologies based on a markup language. Synchronization markup language is a method used by SyncML for synchronizing data across hand held devices and PC's [10]. However this technology did not fit the requirements of the Synchrona project as it could not facilitate data synchronization between Internet services. The services which support this technology include only mobile applications. If we are to use this technology for retrieving data, then services that we are to work with like Facebook, Twitter, Blogger and Moodle should already have this functionality embedded or it should have an interface for letting a third party to integrate this technology into their product.

Taking the above functional requirements into consideration new synchronization architecture was built for the Synchrona system consisting of four main components. Following are the abstract descriptions of the dependencies and the implementation of those components for the Synchrona system.

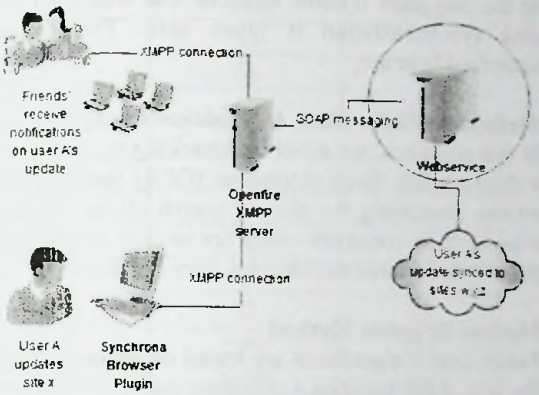

Figure 1. Synchrona Architecture

A. *Synchrona browser plug-in*

Data retrieval or update retrieval will be done at the users' end via a web browser plug-in. The browser plug-in is implemented to capture user updates on websites, set by the user in the browser plug-in preferences. The browser plug-in will get user updates for these selected sites by intercepting the HTTP traffic flowing from the browser to the servers of these sites. The HTTP POST data intercepted in this manner will contain the form post parameters of the user's update. This intercepted data will then be passed on in XML format via XMPP (Extensible Messaging and Presence Protocol) to the Synchrona XMPP server.

B. *Real-time Communication*

The XMPP server sits between the web browser plug-in and the web service. It has two main tasks. Firstly to direct user updates coming from the browser plug-in to the web service and secondly to send notifications about user updates to the user's friends, who are also using the Synchrona browser plug-in.

The main requirement for the XMPP server was for sending real-time notifications to friends. The use of Openfire server (an opensource Java XMPP server) [7] as the XMPP server that lies between the web service and the browser plug-in simplifies this task and removes the requirement of implementing the real-time notification system in the web service. The browser plug-in in this case acts as an XMPP client that sends user updates to the Openfire server and receives friend notifications from the XMPP server. A custom plugin was implemented for the Openfire server to enable broadcasting of received notifications to user's friends. This was implemented by getting all friends of a user from the user's roster and then broadcasting messages to them. Another custom plug-in was implemented to direct incoming user updates to the web service. This was implemented as a client to the web service, that calls the update listener method of the web service, and passes it a SOAP message containing the update XML, in the web service method parameters.

C. *Synchrona Web service*

Synchrona consists of a web service by which the synchronization process will be carried out. The web service is responsible for carrying out all the data processing and the fuzzy data synchronization tasks for the Internet services.

Described below are the two main services of the web service.

- *Status Synchronization*

Status synchronization is one of the aspects we have considered in data synchronization. It consists of synchronizing text based status updates posted on different social networking sites. Synchrona, in its implementation has focused on providing the user with several combinations of status synchronization.

- *Profile Synchronization*

This is another aspect of data synchronization which was considered in this project. Profile synchronization focuses on synchronizing user profile data across multiple different sites. This involves fuzzy data matching between different profile data fields such as contact data, and other personal information.

To implement both status and profile synchronization the fuzzy synchronization architecture described below was used.

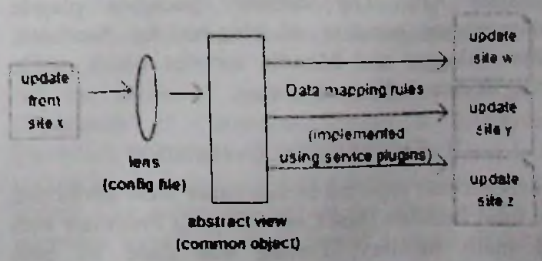

Figure 2. Fuzzy Synchronization Implementation

The concept of fuzzy synchronization centers on the idea of having different data sources of different data

formats being synced with each other. The architecture we have built for facilitating fuzzy synchronization centers around the idea of using a lens for transforming data of different formats into one common format which can be used to update any of the other data sources.

A lens is a bi-directional program. When read from left to right it denotes an ordinary function that maps sources to views. When read from right to left, the same lens denotes an "update translator" that takes a source together with an updated view and produces a new source that reflects the update. [11]

In the Synchrona system the lens has been implemented using a configuration file and a data parser which converts an update from any of the connected Internet Services into a common format. This common format will then be used to update any of the other data sources

D. Synchrona API

Synchrona consists of an API to allow web 2.0 services to connect Synchrona web data synchronization platform. An API is one of the greatest assets of a service which needs careful designing and implementation without leaking any of the service implementation to the outside. [12] Synchrona API implements generic methods to provide the data synchronization functionality to plugged Internet services. The service plug-ins implement the API methods according to the services itself to provide data synchronization.

The API takes use of configuration file information for each service when invoking fuzzy data synchronization for the Internet services connected to the Synchrona web service.

- Internet Service plug-ins

Internet service providers are capable of providing their own configurations and plug-ins to connect to the Synchrona API. The current Synchrona plug-in implementation consists of plug-ins for Facebook, Twitter, Blogger and Moodle services which enable posting of data to these services.

V. Experimental Evaluation

All experiments reported in this paper were performed on a Intel Pentium Dual Core 1.73GHz Processor with 2GB main memory. Testing was done on both Windows 7 and Fedora 10 operating systems using Tomcat 5.5 server and Apache Axis 2 for running the web service. The main tests run were performance tests, to measure the time taken by the web service to synchronize data. Essential target of testing the whole system was about verifying that the product meets the functional, performance, design and implementation requirements identified in the procurement specifications.

Time spent for the method synchronization plays core role in data synchronization. Time complexity analysis of the two most optimal methods that was used in the data synchronization is given here. Two methods described here are,

Method A: Finite State Automation Based Search
In this approach, we avoid backtracking by constructing a deterministic finite automaton (DFA) that recognizes strings containing the desired search string. These are expensive to construct—they are usually created using the powerset construction—but very quick to use.

Method B: Index Method
Faster search algorithms are based on preprocessing of the text. After building a substring index, for example a suffix tree or suffix array, the occurrences of a pattern can be found quickly. As an example, a suffix tree can be built in $\Theta(m)$ time, and all z occurrences of a pattern can be found in $O(m + z)$ time (if the alphabet size is viewed as a constant).

Analysis of synchronization time based on number of users (only one service is being synchronized to).

Table 1. Analysis of synchronization time for Fedora server with increasing no. of users

	Fedora Server	
Users	Method A(ms)	Method B(ms)
1	60	40
2	55	40
3	63	49
4	70	35
5	63	44
6	71	41
7	70	40
8	69	38
9	70	39
Sum	591	366
Avg.	591/9 = 65.67	366/9 = 40.67
Method A = Finite State Automation Based Search		
Method B = Index Method		

Table 2. Analysis of synchronization time for Windows server with increasing no. of users

# of Users	Windows Server	
	Method A (ms)	Method B (ms)
1	61	35
2	58	40
3	60	45
4	63	40
5	64	46
6	75	44
7	73	44
8	73	49
9	72	45
Sum	599	388
Avg.	599/9	388/9
Method A = Finite State Automation Based Search		
Method B = Index Method		

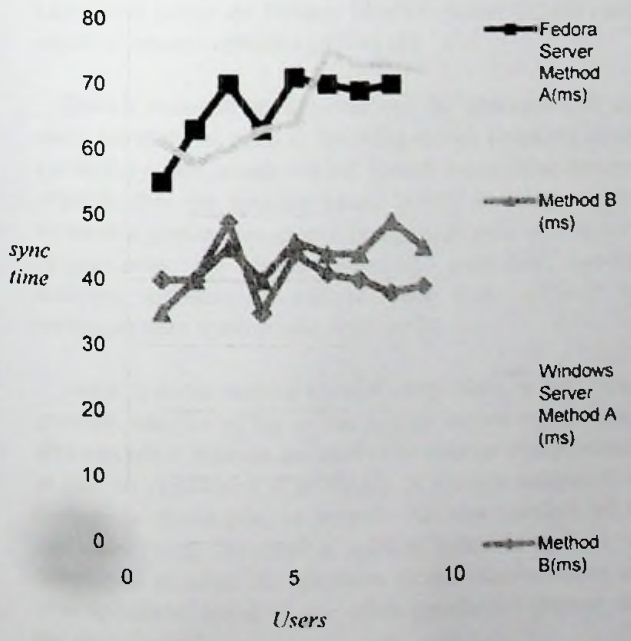

Figure 3. Graph showing synchronization time in ms (y axis) Vs users (x axis).

VI. Possible Optimizations / Future Work

Synchrona proposes to come up with the synchronization of image and video data in the social services that users are connected in as a future enhancement. Also supporting multiple web browsers in the Synchrona browser plugin is another aspect that needs to be worked on in future.

We also believe that the existing implementation (with the API) could be optimized to provide a more robust flexible architecture for synchronizing data between Internet services in the future. In addition to that, to improve the webservice scalability in anticipation of increasing user demand webservice replication can be used.

VII. Conclusion

This paper described so far our goal of presenting a platform for data synchronization between different Internet services. The main intension was to come up with a scalable solution for the real world problem of maintaining the synchronization of user's data in an easy way. The proposed solution consisted of two major functionalities. One is the web service/API for data synchronization and plug-in of emerging web 2.0 Internet services. Other is the notification of the real-time updates for the users' friends while synchronizing the data in each profile.

Acknowledgement

We would like to thank Prof Gihan Dias, Department of Computer Science & Engineering, University of Moratuwa for the guidance and the great support given to us during the project time period. He has being the pillar of the success of Synchrona. Our external supervisors, Mr. Chandika Jayasundara, CEO and Mr. Hiraash Thowfeek CTO of Synergix Pvt. Ltd, Sri Lanka for coming up with the idea of providing a solution for a real world problem in data synchronization.

All the academic and non-academic staff at the Department of Computer Science & Engineering, University of Moratuwa for their guidance and help in various kind of situation during the development of this project. Our colleagues and all the others helped us in numerous ways being the pillars of strength all the time.

REFERENCES

[1] Jon Udell, "Why data synchronization still matters", Data Management, InfoWorld, November 30, 2005, [Online]. Available: http://www.infoworld.com/d/data-management/why-data-synchronization-still-matters-429. [Accessed: Apr. 10, 2010]

[2] P. Sharma, "Core Characteristics of Web 2.0 Services", Nov. 28, 2008. [Online]. Available: http://www.techpluto.com/web-20-services/ . [Accessed: Apr. 12, 2010]

[3] OpenID foundation, "Benefits of OpenID", Get an OpenID, OpenID, 2006-2010.[Online]. Available: http://openid.net/get-an-openid . [Accessed: Apr. 22, 2010]

[4] Chris Messina, "Current state of OpenID", "The Open, Social Web," May. 2009. [Online]. Available: http://agrotime-blog.com/2009/05/08/next09-video-interview-with-chris-messina-on-the-current-state-of-openid/. [Accessed: Apr. 22, 2010]

[5] Seesmic Team, "Ping.fm / Update all of your social networks at once!", [Online]. Available: http://ping.fm/blog/ . [Accessed: Apr. 26, 2010]

[6]ActiveSync,msdn.[Online].Available: http://msdn.microsoft.com/enus/library/ms879772.aspx . [Accessed: May 13, 2010]

[7] "A Synchronization Framework", OpenSync, June, 2010. [Online]. Available: http://www.opensync.org/ [Accessed: July 3, 2010]

[8] Maximilian Berger. Integrated PIM data management with SyncML. Maximilian Berger, 2001. [Online]Available: http://max.berger.name/research/syncml/syncml.pdf

[9] "Performance consequence of polling", The old new thing, 24 Jan 2006.[Online].Available: http://blogs.msdn.com/b/oldnewthing/archive/2006/01/24/516808.aspx

[10] SyncML, Open Mobile Alliance, 2007. [Online]. Available: http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html . [Accessed: May 13, 2010]

[11] J. Nathan Foster, Grigoris Karvounarakis. Provenance and data synchronization. IEEE Computer Society Technical Committee on DataEngineering. [Online]Available: http://sites.computer.org/debull/A07dec/foster.pdf . [Accesses: June 12, 2010]

[12] Joshua Bloch, How to design a good API and why it matters, Google. [Online] Available: http://lcsd05.cs.tamu.edu/slides/keynote.pdf . [Accessed: June 14, 2010]