A P2P Blog System with OpenID Integration

Chen-Pu Lin[1], Yung-Wei Kao[1], Shyan-Ming Yuan[1,2]
[1] Department of Computer Science and Engineering
National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu 300, Taiwan
[2] Department of Computer Science
Asia University, Lioufeng Rd., Wufeng , Taichung County, Taiwan
chenpu.w.lin@gmail.com, ywkao@cs.nctu.edu.tw, smyuan@cs.nctu.edu.tw

Abstract

Blog services become popular in recently years. Web users can post their articles through the services. On the other hand, the users do not have full control of the articles after posted while the services are built with client/server model. Therefore, we propose to use peer-to-peer (P2P) technology to establish blog services. However, the anonymous feature disables the P2P technology to conduct user identification. The OpenID provides secure and unified authentication mechanism to improve the anonymity. In addition, the OpenID has the single sign on procedure to reduce redundant, multiple accounts and passwords. The paper presents a blog service platform which integrates the P2P networks and the OpenID authentication mechanism together. Therefore, users can establish their own blogs easily with full control. The platform can be utilized for other Internet applications as well.

1. Introduction

The Internet is becoming more indispensable for human activities in recent years. The global network infrastructures have been established intensively and increasingly popular in our daily lives. There are many services over the Internet, i.e., blogs, auctions, web photo albums. Especially, the blog services become popular in the recent decade. A blog is a website which a person or people can write articles on it. The articles in the blog are commonly displayed in reverse chronological order. Most web users own their blogs provided by the blog service providers (BSPs). Blog services provided by BSPs are generally built with client/server architecture, and the article data are generally stored on the centralized server. However, users do not have full control of the articles; they may suffer from the lost of data while BSPs change their policies. In addition, less people have professional knowledge or time to establish the blog service by them self bypassing BSPs. Therefore, we intend to build a blog system on peer-to-peer (P2P) architecture to solve these problems.

On the other hand, according to the anonymous characteristic of the Internet, it is difficult to enforce the true users' identities. Nevertheless, in general, most of the web sites require the membership registration to conduct authentications. User Account and Password are the most essential information needed for registration. Occasionally, the web sites may demand users' demographic data, i.e., name, gender, address, birthday, etc. Such data may not need to be modified or updated frequently. Moreover, the information may be required by other web sites as well. Thus, the users need to fill out the data repeatedly for every request from different web sites; users must remember different accounts and passwords for all web sites. Furthermore, the anonymous feature disables the P2P technology to conduct user identification; it isn't easy to handle account problems on the P2P network. However, OpenID, a web authentication standard, becomes strong and popular recently. It is a general solution for multiple usernames across different websites. Therefore, we intend to adopt the OpenID technology to solve the anonymity problem of the P2P network.

The paper presents a new type of architecture to have a blog system by a platform which combines the P2P technology and the OpenID standard.

In the following sections of the paper, we first describe the background of this paper. In Section 3, we introduce the system architecture. The comparisons between the P2P blog system we proposed with relevant related works are discussed in section 4.

978-0-7695-3407-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCIT.2008.75

Name	Latest Version	P2P	Language	Pub/Sub System
JXTA	JXSE 2.5 (2007/11/7)	unstructured	Java	propagation
Open Chord	1.0.5 (2008/4/11)	structured (Chord)	C	Not supported
FreePastry	2.0_03 (2007/11/2)	structured (Pastry)	Java	Scribe

Table 1. Three P2P Pub/Sub implementations

Finally, the paper conclusion is drawn in section 5 and the future works are listed in section 6.

2. Background

2.1. Peer-to-Peer Publish/Subscribe

There are lots of P2P implementations, but less of them have great maintenance and enough scale. Table 1 includes three representative implementations in common usage: JXTA [1], Open Chord [2], and FreePastry [3].

JXTA has a message-passing mechanism called JXTA wire. Its capability is similar to Pub/Sub system, but its operation is related to non-pure P2P network (it needs super peer). The difference between JXTA and general Pub/Sub system is that JXTA uses the one-to-many message-passing mechanism, not the many-to-many message-passing mechanism of the general Pub/Sub system. In addition, performance of JXTA is poor because of its propagation method. Open Chord is based on Chord. Although Chord has related functions for Pub/Sub systems, Open Chord has none of them. By contrast, the Scribe in FreePastry is a simple topic-based Pub/Sub system which meets our need. So we adopt FreePastry to implement the P2P blog system.

2.2. OpenID

OpenID is an open, decentralized, free framework for user-centric digital identity [4]. It provides a free and easy way that users can use a single digital identify across the Internet. It also eliminates the need for multiple usernames across different websites, and simplifies the users' online operation, and enhances the users' online experience.

The terms used in the OpenID protocol are described as followings [5]:

(1) Identifier: The URL or XRI are chosen by the End Users as their OpenID identifier.

(2) End User: The person who wants to assert his or her identity to a site.

(3) Relying Party: The site which wants to verify the end user's identifier. Sometimes it is simply called site.

(4) OpenID Provider: A service provider offering the service of registering OpenID URLs or XRIs and providing OpenID authentication (and possibly other identity services).

The current version of OpenID is OpenID 2.0, released in December 5, 2007. The end user can own an OpenID identifier and login to a website supporting OpenID by the identifier. The website usually has an input box with an OpenID logo for user's identifier. The user's identifier is also a URL that describes the authentication server. The unauthenticated identifier is called "claimed identifier". The relying party can find user's OpenID provider from the URL and require authentication. This moment, the user's browser redirects to the website of OpenID provider. The user logins and permits the authentication from the external site. After that, the authentication server notifies the external site of the successful authentication. The user's browser redirects to the site and uses its services with OpenID identity.

Up to July 2007, there are only approximately 120 million valid OpenID accounts and approximately 4,500 sites have integrated OpenID consumer support [6]. However, some websites with a large number of members also began to support the plan [6] [7]. For example, Yahoo users can use their Yahoo IDs as OpenIDs starting from January 31st, 2008. We can expect the number of OpenIDs becomes larger and larger. The situation of OpenID authentication is more and more universal. Hence, our system adopts the solution to eliminate the problems of identity authentication and management. Therefore, the P2P Network architecture doesn't have to maintain its own account/password information.

3. System Architecture

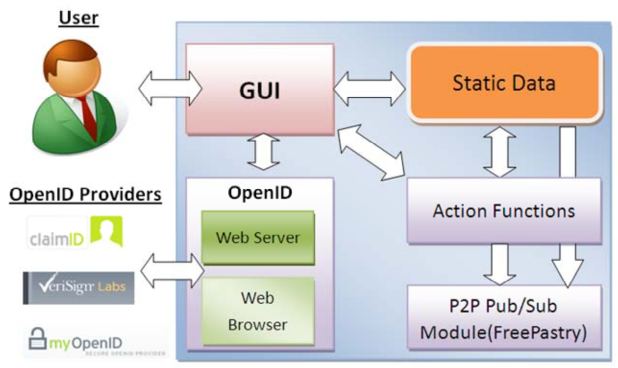


Figure 1. System architecture

3.1. OpenID

The P2P Blog system needs a web server inside to integrate itself with the authentication standard, OpenID. OpenID provides the authentication mechanism for web sites, but no integrated solution for windows applications. We know that there are three roles in OpenID standard: "User", "Site", and "OpenID Provider", and they contact with each other via HTTP protocol. In addition, based on the operation of OpenID protocol, OpenID Provider must receive an absolute address to send the "Site" information back after user logs in. We can not build a central web site to be the "Site" because it will be the bottleneck in the architecture and destroy the original intention of distributed network. However, we still need a web site which responses for accepting messages from the OpenID Provider. Based on the situation, we propose our solution to attach a light web server to user's computer. In other words, users play "User" and "Site" role simultaneously, and "User" can communicate directly with "Site".

The OpenID part in the P2P Blog system includes all mechanisms for OpenID integration. The web server in our system is responsible to build a website temporarily, and the website can work and communicate with OpenID Providers. While users want to login, the windows application launches the web server via a batch file. At this moment, the web application, named WebApp_OpenID, in the web server is usable and allows users to login with OpenID. Users can input their OpenID identifiers on the website, and the website can deal with the identifier. We assume that the port of the web server is 8080, thus, we set redirect address to "http://localhost:8080/WebApp_ OpenID/returnurl.jsp". Next, we find out the OpenID Provider of user's identifier and deliver a request to it. The OpenID Provider exchanges messages with the website. Since our redirect address is on the localhost, it allows the user to redirect to "localhost" after logged in on the OpenID Provider website.

After completed the steps of login through OpenID Provider, users visit the redirect address. On the returnurl.jsp, we deliver the results via socket stream. The stream is a string which is formatted as: "OpenID:useridentifier", e.g., "OpenID:http://user.ope nid.example.org". If there are any failures or errors, the string format will be: "Fail.Error Message", i.e., "Fail.login failure".

3.2. Static Data

Static Data is a special class that all variables and functions in this class are static. It is similar to the existence of global variables. The common feature of these variables is that the whole program only needs these variables with the same version, such as user identifier and P2P environment parameters.

The main work of Static Data is initialization, e.g., initializes the P2P network environment, or accesses a configuration file, config.properties. The configuration file maintains several modifiable parameters, i.e., default bindport value.

3.3. P2P Pub/Sub Module (FreePastry)

The P2P Pub/Sub Module manages all operations about P2P network and Pub/Sub system via FreePastry. It needs to be supported directly by P2P libraries, hence, with different P2P libraries, the implementation ways diverge.

Our system is started and joins a P2P network at initiating Static Data. The Static Data achieves the job by the functions of P2P Pub/Sub Module.

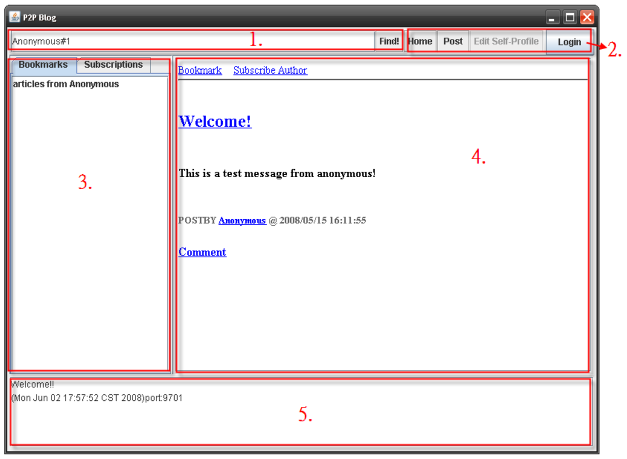
3.4. Action Functions

Action Functions are responsible for specific abilities according to different applications. For example, a blog system should have the ability of posting an article. At this moment, it critically controls how to store an article. We know that our system is totally distributed and the data are stored on the P2P network. Actually, the data are composed of two kinds of types, profile and article. A profile file is used to save information of single personal account, such as the number of articles; an article file is used to store an article, including title, content and comments.

According to the P2P protocol, every file contains one unique identifier. The identifier is produced by hashing a key. In other words, we can't find the file if

Table 2. Custom URL in the presentation canvas

URL	Action (useridentifier / useridentifier#N)
http://Action.Comment.<Link>	View comments / Leave a Comment
http://Action.targetlink.<Link>	List all articles of the author / Link to the article
http://Action.Bookmark.<Link>	Bookmark it
http://Action.Subscribe.<Link>	Subscribe all articles of the author



articles from Anonymous

Welcome!

This is a test message from anonymous!

POSTBY Anonymous @ 2008/05/15 16:11:55

Welcome!!
(Mon Jun 02 17:57:52 CST 2008)port:9701

Figure 2. System screenshot

we don't have the key. Thus, we define the rules of how to generate a key. The key of a profile file is the user's identifier (OpenID), and the key of an article is made of the concatenation of user's identifier and the serial number of the article (useridentifier#N). The serial number (N) is counted from 1 and independent of each user. For example, if a user's identifier is "http://user .openid.example.org", then the key of his third article is "http://user.openid.example.org#3". Therefore, the program can find out all information produced by a user with his OpenID identifier.

3.5. GUI design

The GUI of our system is shown in Figure 2. We describe the five areas of GUI in detail.

(1) Searching box: Users can input an exact key of data to find out the information.

(2) Action buttons: Users can link to his homepage via the "Home" link. If the users do not log in, it will redirect the user to the blog of Anonymous. "Post" function can publish articles. "Edit Self-Profile" function can edit information of self account. "Login" function provides the authentication ability for users to login to our system.

(3) Bookmarks and subscription list: Users can choose the view by tab controller.

(4) Presentation Canvas: This area is responsible for displaying information of blogs. The information is composed of several articles. We can slightly typesetting and print them with simple text, but it is not convenient or intuitive. Users may have many kinds of interactions to the article, e.g., subscribe author' s articles, or leave comments. The interface may become complex and unnecessary while there are lots of buttons for those actions.

Therefore, our program dynamically generates html codes from the information and displays it with web page style on the Presentation Canvas. It is different to the previous embedded web browser. In addition, the program transforms the information into html codes just for the basic color and hyperlink control. The Presentation Canvas has some hyperlinks for usual actions. If users click hyperlinks, the program can parse the hyperlinks and executes actions. The hyperlink URL formats are shown as in Table2. (<Link> stands for "useridentifier" or "useridentifier# N").

After we transformed the information into web page style, the operation of information is close to the user experiences, which makes users easy to get started. More importantly, the webpage-style improves the presentation and scalability of operations.

(5) Message windows: The messages of operations are also logged in the system log file.

4. Comparison

Table 3 illustrates the comparison between the P2P blog system we proposed with relevant related works. In contrast with traditional client/server architecture, our blog system is totally distributed. Although NUWeb is regarded as a distributed system, there is still a central portal site which is the bottleneck of NUWeb, which has the same problem of DSPs. In addition, only web browser is needed to visit the traditional blog systems, but users have to install additional application to use NUWeb and our system.

Table 3. Comparison with NUWeb and traditional blog systems

	P2P Blog with OpenID	NUWeb	Traditional blog systems (via BSP)
Architecture	***Distributed***	Distributed but a central portal site	Client/Server
Additional installation	Custom windows application	Custom windows application (NUBraim)	**none**
Registration	***OpenID***	Provided by central portal site	Provided by BSP
Subscription	Scribe	RSS	RSS, Atom
Domination of blog information	Complete	Complete	Partial

On the other hand, users have to apply for membership of BSP when they intend to build their own blogs. In our system, we only require OpenID to build a blog. Moreover, all three blog systems have subscription mechanism. Finally, the domination of blog information is complete in our system and NUWeb, but not in traditional blog systems.

5. Conclusion

The blog service is more and more popular in recent years. Blog services are generally provided by BSPs. However, the article data is traditionally owned by user but maintained by BSP. Users may suffer from the loss of data without the domination of data. The paper proposes the P2P Blog system to solve the domination problem. We have established the blog service on the totally distributed architecture. Except for the advantage of protecting personal control ability, building the system via P2P architecture is simpler than via client/server model. Since P2P technology already has some essential mechanisms such as routing and backup/duplication of files. However, P2P network requires enough users to keep the network operation steady. In addition, the system has to notify users about the security problems on each host.

In order to recognize the owner of blog information, the user identification mechanism is needed. However, the anonymous feature of P2P network disables the P2P blog system from providing authentication. In the paper, we propose the P2P blog system which integrates the P2P network and OpenID standard to solve the authentication problem. Hence, the user identity is possible in the P2P blog system with entirely distributed P2P network environment.

With the authentication function, the P2P with OpenID platform can be utilized for other Internet applications as well, e.g., auctions, forums.

6. Future Works

We have defined the following problems to be solved in the near future:

(1) The P2P Blog system only searches data by using exact keys to find file identifiers because of the limitation of FreePastry. It is inconvenient to visit blogs by exact keys. We have to integrate our system with advanced keyword search.

(2) The traditional blog systems have a lot of data of statistical analysis, e.g., popular articles, hot blog list, etc. It is difficult to count exact results from the distributed nodes in P2P network. Hence, we intend to provide reference values to users.

(3) Now users can only post articles of full text. We will allow multimedia presentations in an article, i.e., pictures and videos.

(4) The data on the P2P network is stored as files, and the file size will affect the system efficiency, especially when the articles have multimedia contents in the system. For the reason, we intend to divide large files into several fragments for system stability in the later operations. In addition, we may design different

methods for different kind of data type to solve this problem.

(5) Beyond the P2P Blog system, we intend to refine the core architecture to make it a more general purpose service platform. We also plan to implement other applications on the P2P with OpenID platform.

Acknowledgments

This work was mainly supported by National Science Council grant NSC96-2221-E-009-142: pub/sub-based P2P platform and its applications, and partially supported under grant NSC96-2520-S-009-007-MY3.

Reference

[1] JXTA. https://jxta.dev.java.net/

[2] Open Chord. http://www.uni-bamberg.de/en/pi/bereich/research/software_projects/openchord/

[3] FreePastry. http://freepastry.org/

[4] OpenID. http://openid.net/

[5] OpenID. http://en.wikipedia.org/wiki/OpenID

[6] Michael Arrington(Jan 17, 2008). Yahoo Implements OpenID; Massive Win For The Project. http://www.techcrunch.com/2008/01/17/yahoo-implements-openid-massive-win-for-the-project/

[7] Marshall Kirkpatrick (Feb 7, 2008). OpenID: Google, Yahoo, IBM and More Put Some Money Where Their Mouths Are. http://www.readwriteweb.com/archives/openid_big_companies.php