國立交通大學

資訊科學與工程研究所

碩士論文

一個整合 OpenID 與點對點技術之部落格系統

A P2P Blog System with OpenID Integration

研 究 生：林辰璞

指導教授：袁賢銘　教授

中 華 民 國　九 十 七 年 六 月

一個整合 OpenID 與點對點技術之部落格系統

學生：林辰璞　　指導教授：袁賢銘

國立交通大學資訊科學與工程研究所

摘要

網路服務使用廣泛，爲了將資料集中管理，它們通常使用 client/server 架構建置而成。另外，部落格系統近幾年來非常流行，使用者可以在部落格上發表文章，可是若使用 client/server 架設部落格系統卻會造成使用者沒有資料自主權。所以我們使用點對點架構作爲架設服務的新方案。此外，點對點技術具有匿名的特性，而新興的 OpenID 則以解決頻繁且多餘的網路身份認證而崛起，並提供安全、統一的認證機制。本篇論文將點對點技術整合 OpenID 技術爲平台，建置點對點部落格系統。因此使用者可以方便、輕鬆、自主地建置個人部落格。本系統也可以應用在其他的網路服務上，爲另一種提供網路服務的新方式。

P2P Blog System with OpenID Integration

Student: Chen-Pu Lin Advisor: Shyan-Ming Yuan

Institute of Computer Science and Engineering

National Chiao Tung University

Abstract

Internet services are general and built with client/server architecture commonly due to the advantage of easy data management. Besides, the 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 use peer-to-peer (P2P) technology to establish services. Furthermore, the P2P technology possesses anonymous features. The OpenID can provide secure and unified authentication mechanism to improve the anonymity. In addition, the OpenID has the single sign on procedure can reduce redundant, multiple accounts and passwords. The thesis presents a platform that integrates P2P networks and OpenID authentication mechanism together. The blog services are implemented based on the platform. Therefore, users can establish their own blogs easily with full control. The platform can be utilized for other Internet applications as well.

Acknowledgements

首先最要感謝的是袁賢銘老師兩年來的指導，自由開明的風格讓我受益良多；透過比賽、專案、論文的過程，給予很多建議，也讓我們有許多發揮的空間，接觸各方面不同的領域。另外，非常感謝謝筱齡老師在百忙中抽空協助，針對論文初稿的文字做修飾，教導我不少寫論文的技巧。同時，還要謝謝實驗室的秉哲學長、繼弘學長、永威學長、家鋒學長，及所有實驗室的碩班同學鴻仁、士強、明志、宜豐，隨時在研究的過程中給予指教及鼓勵。

最後，感謝父母、兄姐背後的支持和信任，讓我能專心致志的學習，並且一路陪伴我完成我的求學生涯，謹以此文獻給我摯愛的家人。

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Preface

The Internet is becoming indispensable for human activities. The global network infrastructures have been established intensively and increasingly popular in our daily lives. The main feature of Internet is no boundary. Nodes on the Internet can communicate with one another. In addition, applications exchange messages through the Internet as well as distribute large amount of computations among the nodes. Companies and commercial business maintain their own web sites. The original functionalities of a web site provide products information and contact procedures. Therefore, people can expand the business over the Internet. Customers can use the online services without temporal and spacial constraints. Furthermore, it can reduce business expanses while operating over the Internet.

Services implemented over the Internet can be accessed via personal computers, mobile devices, etc by users. However, 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 membership registration, authentications. *User Account* and *Password* are the most essential parameters. Occasionally, the web sites may demand users' demographic data, i.e., name, gender, address, birthday, etc. These data may not need to modify or update often. Moreover, the information may be required by other web sites as well. Thus, the users need to fill out the data repeatedly for every requested web site; users must remember different accounts and passwords for all web sites.

On the other hand, the personalized Internet services play important roles in the

recent decade, i.e., personal homepages, personal web photo albums, and web logs – blogs. Most blogs are established via blog service providers. However, there are drawbacks while blog services are built with traditional client/service mechanisms. The users intend to build their own blogs bypassing the service providers. Unfortunately, most users do not have adequate computer knowledge; they are not capable to construct the blogs by themselves.

1.2 Problem Statement and Motivation

As mentioned in the previous section, people use online services, which are convenient to customers and saving costs for business owners. Also, blogs gradually become personal information broadcast station. The trend of “Internet-Centered” life style has been universalized, but we realized there are still several problems:

1. A large part of web applications are money-related, such as online shopping, online auction, and online banking. It highlights the problem of online identity verification can not be ignored. We need safe identity verification to solve this kind of problems.
2. Whatever the services provided by the web site involves are, all of those require users to create their own account so that service providers can have these personal information and do some analysis. In terms of users, they must have an account to login web sites and use individual services, such as mailbox, e-news, and personal homepage. With exploding amount of the web sites, users have to register more and more accounts. Moreover, they are asked for similar personal information repeatedly on every registration, i.e., name, gender, birthday, hobbies, etc. Users are burdened with filling out repeated information and remembering username and password.
3. Blog system is one of hot services on the Internet nowadays. It is a kind of

communication and becomes popular recent years. There are many web sites which provide Blog service, for example, Blogger.com and WordPress.com. These web sites are called BSP (Blog Service Provider). These web sites are always popular. We know the reason is the contents which are provided from users. Users are these contents' authors. But authors can not own their writings. BSP hold all articles. Whenever the policy of BSP changes, authors may bear losses of no reason [17][19]. Even these authors need to pay BSP some money to back up their writings. Therefore, some bloggers set up blog by themselves. But most of the users have no way to set up a blog and still can't leave BSP industry. They can only silently bear losses. After all, if we build our blog without BSP, the cost of setting up a blog is heavy. The cost includes the machinery maintenance, software updates, data backup, and security protection, in addition to the cost of money, manpower and time considerations. Under so strict conditions, there are only a small number of users who can afford such cost in fact.

Above problems are observations in current Internet environment. Peer-to-Peer (also called P2P) technique grows quickly and becomes mature. Most applications using P2P are related to file sharing, and the two most famous applications are BitComet and e-Mule. The main characteristic of Peer-to-Peer is "distributed". If we implement a blog system using P2P technique as overlay network, then blog users can build their blog on their own computer. In other words, users can establish their own blog system and control their writings without worrying about policies of BSP.

Besides, OpenID, a new authentication tool on the Internet, is a good solution about user authentication in our P2P Blog system. OpenID solves some typical drawbacks like security problem of personal information as well as too many accounts

to remember. Therefore, we propose a concept of “P2P Blog system with OpenID”. The objective is to utilize scalability and compatibility with errors of P2P technique on a totally-distributed blog system.

1.3 Outline of the Thesis

In Chapter 2, we introduce backgrounds and related works. In Chapter 3, we show a diagram of system architecture and describe every partition. Chapter 4 discusses the functions of implemented components, the correlations among the components, and several complicated program flows. Next, in Chapter 5 presents the results of our system and operations. We compare P2P blog system with other blog types in Chapter 6. Finally, we talk about the future works and conclusion in Chapter 7.

2 Background and Related Works

2.1 Background

2.1.1 Peer-to-Peer (P2P) network

The data and computing were centralized in a small number of servers in traditional, and all the clients must be directly dependent on them. This approach is called server-based network, or client/server module. On the contrary, we distributed the data and computing loading over the computers (peers) on the network, this is called Peer-to-Peer (P2P) network [2]. There are many successful P2P applications, such as BitTorrent, Skype. They show the advantages of P2P network and verify the practicability of P2P network architecture.

The main characteristics of the P2P systems are the ability to pool together and harness large amounts of resources, self-organization, load balancing, adaptation and fault-tolerance [6].

“There are two classes of P2P overlay networks: Structured and Unstructured” [7]. “An unstructured P2P network can be easily constructed as a new peer that wants to join the network can copy existing links of another node and then form its own links over time. The main disadvantage with unstructured P2P networks is that the queries may not always be resolved. If a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful”[2]. Structured P2P network usually has a distributed hash table (DHT) to resolve the problem. Such a network use hash function to give endpoint a hash value, and

determine the endpoint to be responsible for which content according to specific protocol. Many studies are published about DHT in the P2P network, such as Pastry[3], Chord[4], CAN[5].

A P2P network builds on the overlay network above the Internet. “An overlay network is a computer network which is built on top of another network.” “Overlay networks can be constructed in order to permit routing messages to destinations not specified by an IP address.”[8] And other advantages. So using virtual communication protocol, we can escape from original Internet communication protocol. For example, A structured P2P network maintains a DHT and routing messages to logical address without having IP address. Therefore, the P2P network can work smoothly by its specific protocol.

The disadvantage of P2P networks is that users need query everything to get data. And they find the data from lots of results. Paul [9] has an example: In file-sharing, if Bob hopes to get the latest publication, he must query with the author name, and look for his interest publication to download it. To resolve the search problem in P2P network, researchers tried to integrate Pub/Sub and P2P systems and provided a greater mechanism.

2.1.2 Peer-to-Peer Publish/Subscribe

Publish/Subscribe (Pub/Sub) is a popular message-oriented middleware (MOM). The main characters are publisher and subscriber. The publisher is the information provider. The subscriber has the capability to define its interest event or event type and subscribe them. Whenever the publisher publishes an event, Pub/Sub system will transfer these events to their subscribers [6]. Therefore, Pub/Sub systems have the following three functions [9]:

1. Advertising: publishers publish the event to Pub/Sub systems.
2. Subscribing: subscribers subscribe events or event types.

3. Notifying: after publishers publish new events, the Pub/Sub systems will send them to their subscribers.

Pub/Sub systems have two kinds, topic-based and content-based. In topic-based system, subscribers can subscribe a topic and get any events about the topic. In content-based system, it describes events by using specific attributes and values and has more difficult technology than topic-based system.

An advantage of Pub/Sub systems is loosely-coupled. Publishers and subscribers don't know each other but the Pub/Sub system works smoothly. The other advantage is that Pub/Sub systems are more scalable then traditional client/server module.

After P2P and Pub/Sub are combined, the Pub/Sub system provides services on the upper layer, and the P2P system is responsible for messages routing on the bottom layer. The Pub/Sub system allows the P2P system to own great ability in information filter and cuts down the consumption in bandwidth. In recent years, there are many studies about combining Pub/Sub with P2P. Scribe and Hermes are the representation of the early P2P Pub/Sub systems. They are both based on Pastry. After them, most of the studies are based on Pastry and Chord.

There are lots of P2P implementation, but less have great maintenance and enough scale. The table below includes three representative implementations in common use [10] [11] [12].

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 2-1 Three P2P Pub/Sub implementations

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). And it is one-to-many message-passing mechanism, is different to Pub/Sub systems that are many-to-many message-passing mechanism. In addition, its performance is poor because of propagation method. Open Chord is based on Chord. Although Chord has related documents on Pub/Sub systems, it seems to have no result in Open Chord. By contrast, the Scribe in FreePastry is a simple topic-based Pub/Sub system and meets our need. So we adopt FreePastry to implement the P2P blog system.

2.1.3 OpenID Authentication

"OpenID is an open, decentralized, free framework for user-centric digital identity."[13] 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 users' online experience.

The following are terms in OpenID protocol [14]:

1. **Identifier**: The URL or XRI chosen by the End User 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 that 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 is OpenID 2.0, released in December 5, 2007. The end user has an OpenID identifier and logins 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 directs to the website of OpenID provider. The user logins and permits the authentication from the external site. Then the authentication server notifies the external site of the successful authentication. The user's browser redirects to the site and uses services with OpenID identity.

As of July 2007, there are only approximately 120 million valid OpenID accounts and approximately 4,500 sites have integrated OpenID consumer support.[15] But some sites with a large number of members also began to support the plan.[15][16] For example, Yahoo users can use their Yahoo ids as OpenIDs starting January 31st, 2008. We can expect the number of OpenIDs becomes three times. The situation of OpenID authentication is more and more universal. So our system adopts the solution to eliminate the problems of identity authentication and management.

2.1.4 Blog System

Blog is an abridgment of the term web log. A blog is a website that a person or people write articles and are commonly displayed in reverse chronological order[23]. The blogger writes their words without any limitations. The articles are usually personal experience or life log. The ratio of original articles to all articles is high. The display includes words, pictures, video or hyperlink. Besides, the articles cause exchanges between the netizens. They communicate with others by trackback, comment.

There are two kinds of methods to build a blog. The first is building the blog system by self. It includes writing blog programs or blog suit software, and renting

virtual space in the Internet or using personal machine to store. The second is to rely on blog service providers (BSPs).

The advantage of the first way is the user has the control of all data. But the user must have enough professional knowledge, and he has to expend large effort and time on the maintenance of software and hardware. Therefore, it usually is adopted by the people having computer technology. By contrast, the second way popularizes the blog to people. Anyone can be a blogger without handling the problems about system security, data backup and machine updates. So it is the main way to build a blog. Although building a blog through BSP is convenient, there are potential problems. Every blogger writes articles or data on the blog, but the controller is BSP not the author. Some authors had the experience of damaging writings cause of the BSP's policies changing and began to resist BSP [17][18][19].

More and more bloggers do not want the information held in hands of BSP. Some bloggers begin to build blogs by themselves, but most people still can not escape from BSP and bear the loss silently because of no the enough professional skill.

Furthermore, the number of blog in the Internet is multiples of growth [20]. Each user wants to visit more and more blogs and is unable to stand the load. So the RSS/Atom technology is used to track user's favorite blogs, it eliminates the matters that user has to open the web pages and find new information.

2.2 Related Work

The related applications are not yet general or established maturely and have not been published. So here we only refer a domestic work, it is the product "NUWeb"[21] from Sun Wu in National Chung Cheng University in Taiwan.

NUWeb is a new web system, combined with the WWW and P2P technology. NUWeb

makes users building their websites easily. Users can share photo, video and articles with others. They provide a platform for searching or managing information conveniently.

NUWeb has two areas in mainly, NUWeb PP (Personal Portal) in client and NUWeb CC (CyberCenter) in server. NUWeb PP includes several sub system, a web server in client, browser and information manager (NUBraim), blog system, etc. NUWeb CC includes WNS (Web Name System), backup system, searching system, blog system and a portal site of NUWeb Cyberspace.

Users need to register an account from NUWeb portal site and download software. They publish an article through NUBraim. Some settings need to be set in the web pages. The user's computer is the web server of his personal site, and he can backup data to NUWeb CC. When the local network traffic is particularly heavy or the computer is off-line, visitors can visit the site by backup site in NUWeb CC.

In simple terms, NUWeb allows users being their masters. Users can build personal site in local computer. NUWeb seeks to create a new network service model.

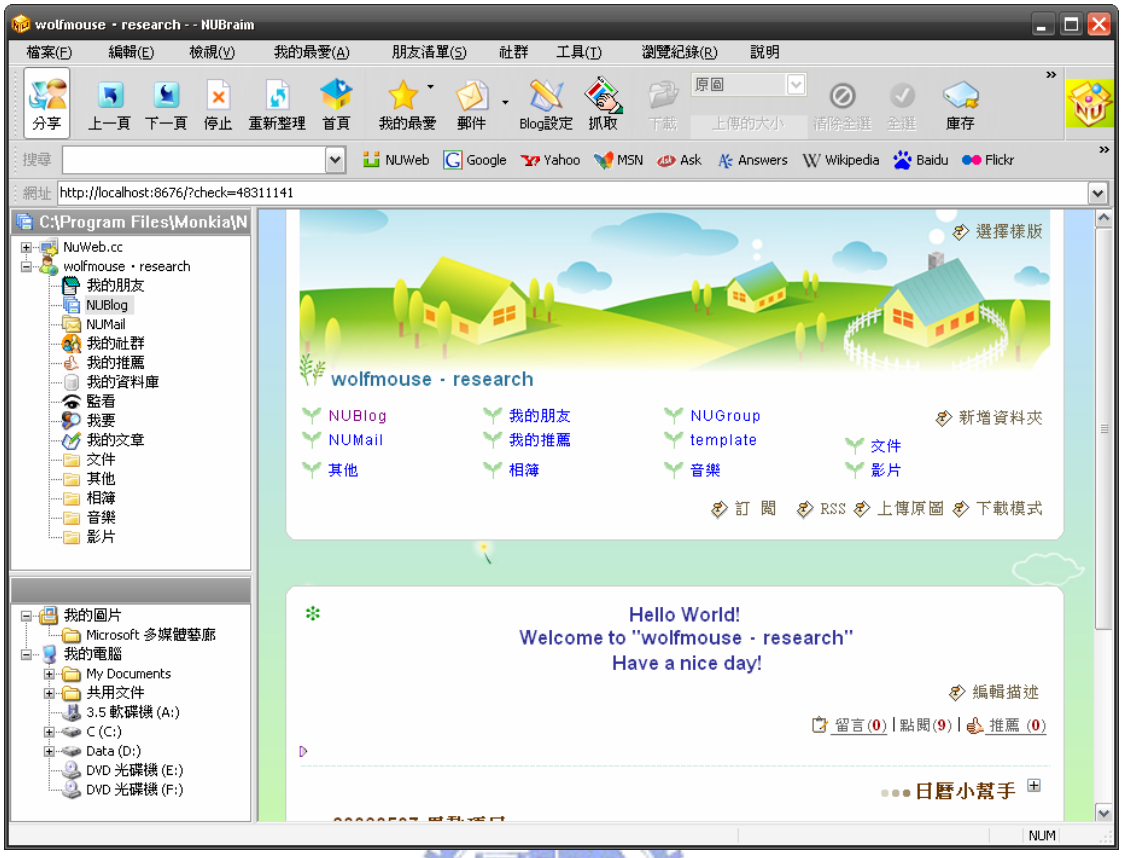

Figure 2-1 NUBraim

Figure 2-2 NUWeb portal site

3 System Architecture

Our system contains several areas and contact with OpenID providers while users login. The OpenID providers for identity authentication are not specific servers from a central organization. In other words, the architecture of OpenID providers is also distributed. In addition, obviously there is no any center which may be a bottleneck in the environment.

P2P Blog System can be divided into several parts to explain: 1. Web Server and Web Browser: The part is related with OpenID integration. 2. P2P and Pub/Sub System: It is responsible for P2P environment operations. 3. GUI and Blog Functions: It uses previous parts to display and implement applications. The details are described in the following sections.

3.1 Web Server and Web Browser

A web server is a program that is responsible for handling HTTP protocol. It accepts HTTP requests from client and sends HTTP responses. There are common web server like Apache, IIS and GFE [22]. Static web pages can be presented by HTML documents, and dynamic web pages need be displayed with related interface, e.g.., JSP, CGI, PHP, etc. The P2P Blog System needs a web server inside to integrate with authentication standard, OpenID. OpenID provides web sites for authentication mechanism, 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 one another via HTTP protocol. 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. On the other hand, from the

operation of OpenID protocol, OpenID Provider must get an absolute address to send “Site” information back after user log in. Thus, we still need a web site that can accept messages from OpenID Provider. Based on the situation, our solution is 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 system starts the web server while users want to log in and shutdowns it after users log out. Moreover, we have a custom web site in the web server. The custom web site implements OpenID protocol and can connect with OpenID Provider and communicate with our windows application.

Web browser is a program that can display the documents in a web server, such as Internet Explorer, Mozilla Firefox and so on. The common way to go on the Internet is through the web browser. Now a web browser is almost in personal computer in general. From the preceding paragraph, we make a web site in users’ computer, and need a web browser to browse web pages. In order to facilitate the users of our system, we embedded the web browser which just runs simple functions into the windows application. Users can visit our web site to log in via the embedded web browser. Therefore, users use our application without switching windows and blocking the intuition of operation.

3.2 P2P and Pub/Sub System

P2P system manages the faculty of P2P network, i.e., builds a new network or joins an existent network, maintains DHT table, communicates with other peers on the P2P network and handles the file storage. In addition, different P2P protocols have different deployments; their maintenances of DHT table and routings are all dissimilar, and also the implementation of the functions. Various P2P protocols have their own advantages and disadvantages, but there are some functions are certainly the same, i.e., queries and inserts data. In theory, using different P2P protocol libraries can make the same functions. Their differences are just suitable for the applications or not. Moreover, P2P system also defines the format of

communication contents for these applications on the P2P network.

Pub/Sub System controls the functions of Publish/Subscribe. Users on the P2P network can obtain information via querying data, but Pub/Sub provides an immediate and convenient way to retrieve data. If we want to use Pub/Sub technologies on the P2P network, they usually need be supported by P2P protocol implementation libraries directly. Besides, Pub/Sub System sends and receives messages, and the contents are different from P2P contents. Therefore, our application makes a Pub/Sub content format, and it is convenient to subscribe and publish messages.

3.3 GUI and Blog Functions

GUI allows users to facilitate the operation and invoke Blog Functions to work. These functions provide different functions according to different applications. Because this paper implements a blog system, these functions provide operations related of blog system, i.e., posting an article. In logical layer, they are established on the upper layer of those modules in section 3.1 and 3.2. Blog functions use OpenID to do identity authentication, and establish functions by using P2P and Pub/Sub System.

GUI is the pivots of the system, and connects to other parts respectively. Something likes the starting and stopping of the Web server, and how to organize the functions to applications on the P2P network, is dominated by this. But the control about web server is still independent, it only relates to login action. In the other side, the partition about P2P has many complex and specific operations because of different applications and libraries. Thus, in order to reduce the correlation between GUI and P2P libraries and simplify the events which GUI deals with, AP Functions are the bridge of the GUI. As long as the libraries' architecture and methods in the implementation are different even the same P2P protocol, we need to rewrite the AP Functions. In Blog System, those are usual actions, i.e., posting an article, getting articles. Therefore, in actually, Blog Functions are the key of any kind of applications in the

P2P with OpenID integration system.

4 Implementation Detail

4.1 Overview

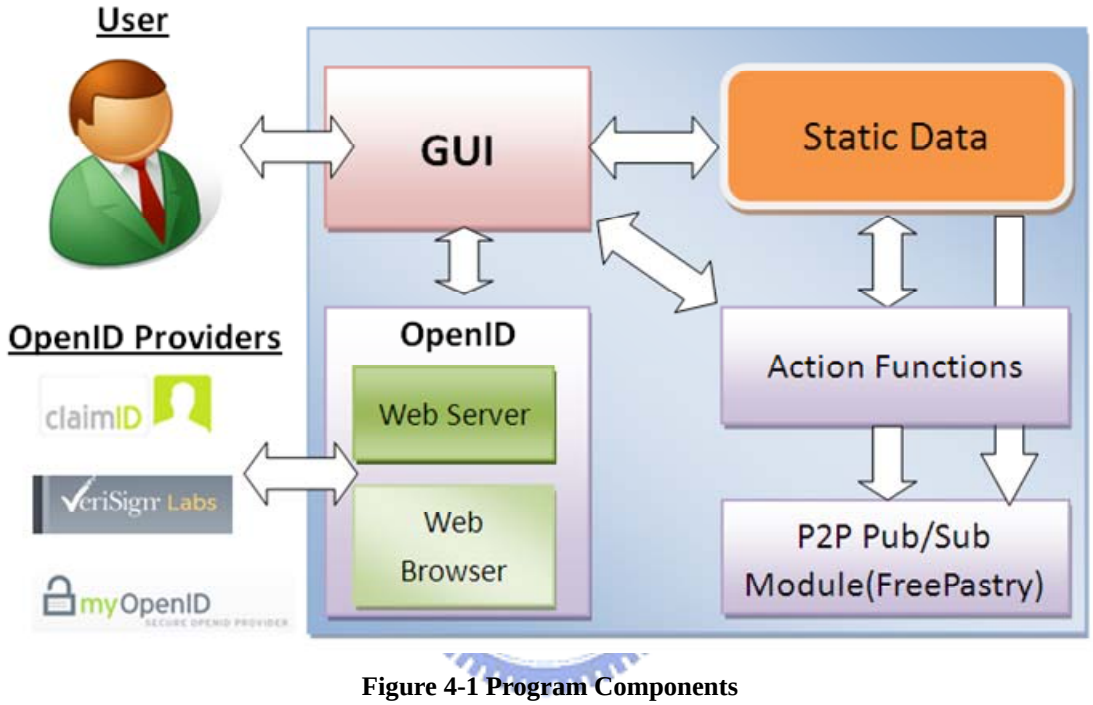

Figure 4-1 Program Components

The diagram above depicts our program components. Users use our system via the graphical user interface (GUI). GUI communicates with other components except P2P Pub/Sub Module (FreePastry). OpenID is responsible for OpenID integration. Static Data controls important static resources in the whole program, and is accessed by GUI and Action Functions; it also access P2P Pub/Sub Module. Action Functions are the particular capabilities about the application. P2P Pub/Sub Module only manages P2P and Pub/Sub operations.

Before we explain every partition of program in implementation, we tell about a commonly used method. The method helps us to link others in this program. It can be said that our program's substruction. This method is Observer Pattern.

4.2 Observer Pattern

Almost the actions need to wait for being dealt with in our system. For example, querying on the P2P network, and logging with OpenID through a web site. In fact, the P2P library we adopted designs use “continuations” to avoid that a program is blocked during doing lookup. “Continuations” allow us to continue processing while receiving results. We can make other requests during waiting periods. These behaviors create new thread, and cause that the main program can not easily obtain results in the multilayer architecture.

Observer pattern is the most commonly used design patterns. Its purpose is to define interdependence among the one-to-many objects. While an object, called subject or observable object, changes its status, the observer pattern will help the object to automatically notify other dependencies, called observer, to do some actions [24]. For instance, in our implementation, GUI is an observer and receives update information from its subjects. Action Functions is the mainly subject of GUI. GUI calls Action Functions to execute actions and waits for messages. In actually, Action Functions are intermediary layers, and observers as well. P2P Pub/Sub Module is the origin of having new threads. Action Functions receive messages from P2P Pub/Sub Module and return them to GUI. Their relations are as shown in Figure 4-2.

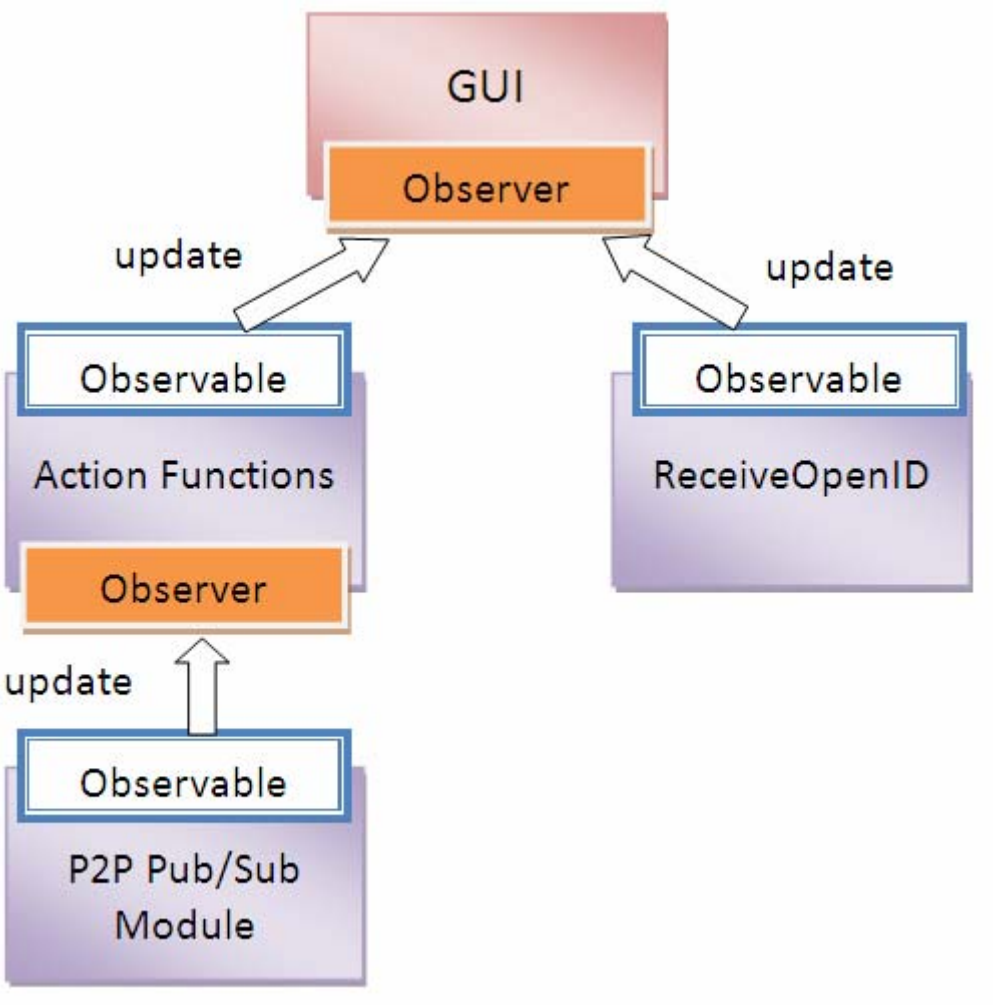

Figure 4-2 Observer pattern in our system

4.3 OpenID

This part includes all mechanisms about 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 applications in the web server are usable. Our web application is a website named WebApp_OpenID, 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. And the OpenID Provider exchanges messages with the website. Because 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 result via socket stream. The stream is a string which format is: "OpenID:*useridentifier*", e.g., "OpenID:http://user.openid.example.org". If there are any failures or errors, the string format is: "Fail.*Error Message*", i.e., "Fail.login failure".

After above actions, the windows application uses another batch file to shutdown the web server. Actually, there are other methods to control the web server, but we adopt the batch file to control it for lower dependencies. In other words, even the version of web server changes and makes mechanisms different, we do not need to modify our program except batch files.

4.4 Static Data

Static Data is a special class which all variables and functions are static. It is similar to the existence of global variables. The common feature of these variables is that the whole program only needs the same one of each variable, such as user identifier and P2P environment parameters. Static Data includes several static functions as well, and most of the functions are invoked in starting our system. Others are common used during program running. In addition, because all variables in this class are static, this class can keep these values only if it is called one time. We can use the class successfully cause of we called the class and initiated it before invoke

GUI.

Static Data gets the configuration file, config.properties. The configuration file decides modifiable parameters. For example, we find an empty port for binding port by using socket testing, and the beginning port is set in this file which the default value is 9701. Then we can call the function in Static Data to join a P2P network. The others in Static Data are retrieving or setting details in P2P operations and applications.

4.5 P2P Pub/Sub Module (FreePastry)

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, so implementation ways almost depend on different P2P libraries.

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. It is worth noting that this partition doesn't rely on Static Data, but static data maintains some constructions are from P2P Pub/Sub Module.

Even if it may be different according to different libraries, there are several necessary elements during the period of building a node and joining a P2P network:

1. While a node is established, it needs a port called "**bindport**". The bindport is the port which deals with messages on the P2P network. The value is determined after Static Data is initiated.
2. After the node is established, it needs at least one **bootstrap** for joining an existent P2P network. A bootstrap is an address of another node on the P2P network. As long as a bootstrap is alive, a node can join the P2P network by it. If all bootstraps are inactive, our system will build a new P2P network by self.

Blog application is belong to data sharing, and each node has space for storing certainly. Thus, P2P Pub/Sub Module determines cache settings, the path, sizes of storage and so on and builds a node based on them. In addition, Pub/Sub system can set stuff by the established node. Moreover, FreePastry is very simple for Pub/Sub control.

Except for functions about initiating P2P environment, P2P Pub/Sub Module has others which can be used by upper layer. These functions are basic operations on the P2P network, i.e., storing data, retrieving data, etc. Components on upper layer use them to build complex functions easily.

Besides, P2P Pub/Sub Module has some data types about P2P operations. It includes the content type for node communicating and for Pub/Sub communicating on the P2P network.

4.6 Action Functions

Action Functions were born to different application's need. The main feature of these classes is that they play not only observer but also observable roles. Because the system uses the observer pattern, the components in the upper layer can call functions in lower layer and the system still works persistently. The system can sense while receiving results, and do next steps. The reason why Action Functions are existent is to make GUI and the detail P2P operations separately. It let these components more flexible and better usable. Another advantage of Action Functions is to help GUI with dividing update contents correctly. If GUI deals with actions directly, it gets pure P2P content type or Boolean value and is difficult to recognize the meaning of these results.

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, these data are composed of two kinds of types, profile and article. A profile file used to save information of single personal account, such as the number of articles; an article file used to store an article, including title, content and comments.

According to the P2P protocol, every file has an only identifier. The identifier is produced by hashing a key. In other words, we can't find the file if haven't the key. Thus, we define the rules how to decide a key. The key of a profile file is the user's identifier (OpenID), and the key of an article is user's identifier plus the serial number of the article (useridentifier#N). The serial number (N) counts from 1 and is independent of each user. For example, a user's identifier is "http://user.openid.example.org", and the key of his third article is "http://user.openid.example.org#3". Therefore, the program can find out all information cause of a user's OpenID identifier.

4.7 GUI

The GUI is on the top of whole program and interacts with users. GUI implements observer pattern to be an observer. It receives information from its subjects. From the previous section, we know the interaction situations between Action Functions and GUI. In Figure 4-2, we also use observer pattern during the process of login. The main program will create another thread called ReceiveOpenID. The thread constructs an observable class, ReceiveOpenID, and let itself be GUI's subject. The ReceiveOpenID class opens a socket port for receiving socket streams and waits for them. Until gets the string, it notifies observers and closes the socket port and the thread finishes automatically later. Therefore, GUI obtains the result of login from the local web server, and completes the identity authentication

successfully.

In addition, the design of GUI is to be user-friendly, and presents the information to all users clearly.

4.7.1 Embedded Web Browser

This partition is non-essential part of GUI, but is important in actually. In usual situation, users visit websites by a software called web browser; popular web browsers are Internet Explorer, Mozilla Firefox and so on. Moreover, we know that to login with OpenID needs to go on the website of OpenID provider. Users use a web browser to visit the website of OpenID Provider and a site which requests identity authentication by using OpenID. But our application is a windows application at local computer. If our system calls default web browser at users' computer while a user log in, the user will leave the view of our main program and switch to the default web browser. After the user completed login, the web browser can't turn the view to our application back. The serious drawback may confuse users, because it breaks off users' emotion on operating. Users may be distracted and visit other websites on the web browser. On the other hand, users may have no pleasures to use it again because of the complexity and no intuition of the operation.

Therefore, our system contains a light and simple embedded web browser. Users can visit the login website with the embedded web browser in main program, and it will be closed after users completed login. It makes the process easy to use, intuitive and smooth. The embedded web browser is too simple to visit other websites, and can avoid distraction. Overall, although the embedded web browser is an additional component, it plays an important role which focuses user's attention and improves user's positive experiences.

4.7.2 Presentation Canvas

How to display the information is this part's job. The information is nothing but articles. We can slightly typesetting and print them with simple text, but it is not convenient or intuitive. Because users may have many interactions, e.g., subscribe author's articles, 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 follows (<Link> stands for "useridentifier" or "useridentifier#N").

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

Table 4-1 Custom URL in the presentation canvas

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

4.8 Program Flow

From the above sections, we know what each component is responsible for. This section lists several complicated operation processes.

We can see the login process, as shown in Figure 4-3. The main focus is in three parts: **web server**, ***socket receiving port*** and embedded web browser. All of these must be close after user logging. If user's logging is successful, the program will try to retrieve the profile file of the account and preserve a copy. If there is no profile file of the account, it will construct a new profile and keep it in the program. While it needs to update the profile file, e.g., users post an article, the program will actually store the file on the P2P network. In addition, we have to get the profile at first because it has subscription list and we need use it to initiate some settings.

The left in Figure 4-4 is subscribing mechanism. Users can use it after logging, so the program needs to check user's logging status. At first, users choose a target, an author, and give a name for identifying. Then the program will add the target to Pub/Sub mechanism, and it can receive publishing messages instantly. Finally, it updates the subscription list from user's profile. The right flow in Figure 4-4 is the process of posting. Users can post an article by self-account or anonymous. No matter which the signature of the article is, it still needs to update the profile of the author (self-account or anonymous) to get the article's series number. Next, the article is stored on the P2P network. And finally, the system publishes messages to all subscribers, and all on-line subscribers will receive these messages immediately.

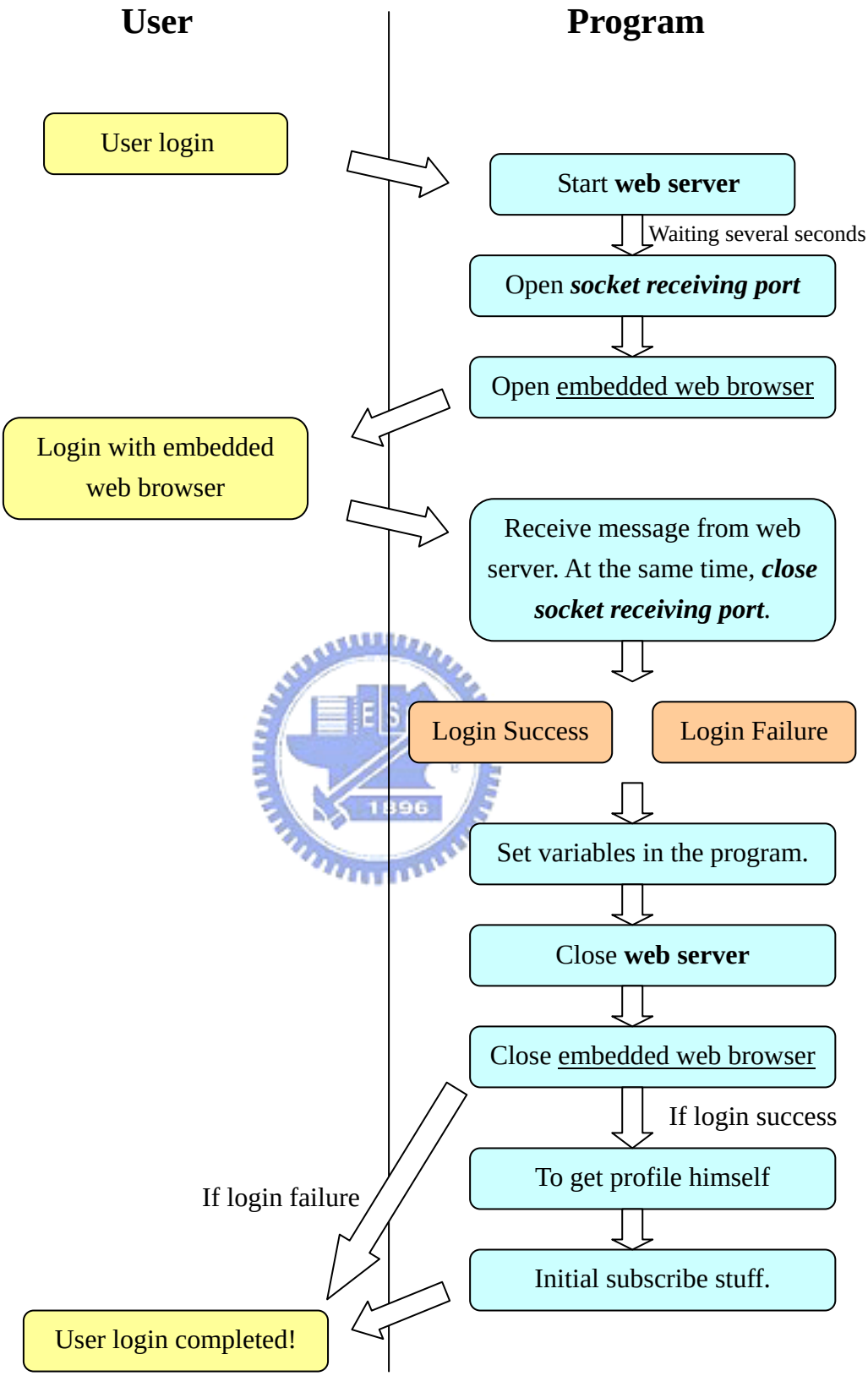

Figure 4-3 Login flow chart

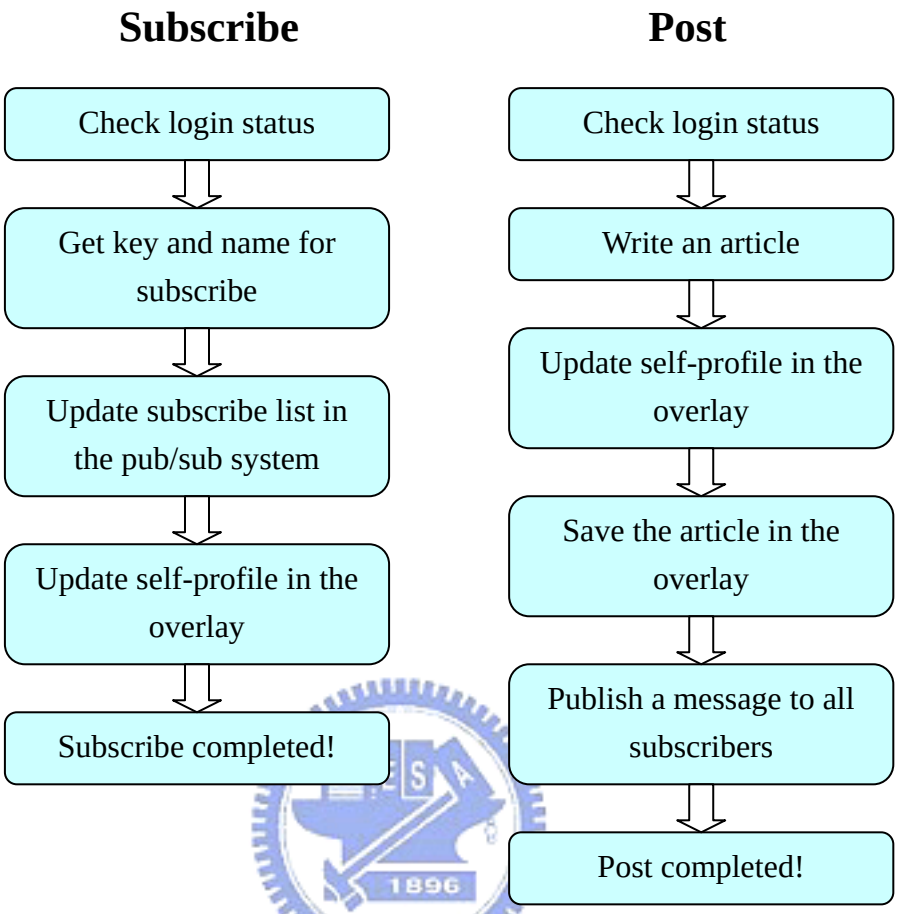

Figure 4-4 Post and subscribe flow chart

4.9 Tools and Libraries

In this program, we use numbers of open source tools and libraries for development. As shown below.

Name	Usage	Version	License
Java [25]	Core	JDK1.6	SUN
FreePastry [12]	Pub/Sub P2P network	2.0_03	BSD-like
Jetty [26]	Light web server	6.1.8	Apache License 2.0
OpenID4Java [27]	OpenID-enable the Java webapp	0.9.4.339	Apache License 2.0
JDIC [28]	Embedded web browser	0.9.4	LGPL
JDOM [29]	Handle XML document	1.1	Apache-style open source license

Table 4-2 Tools and Libraries

5 Application Demonstration

5.1 Overview

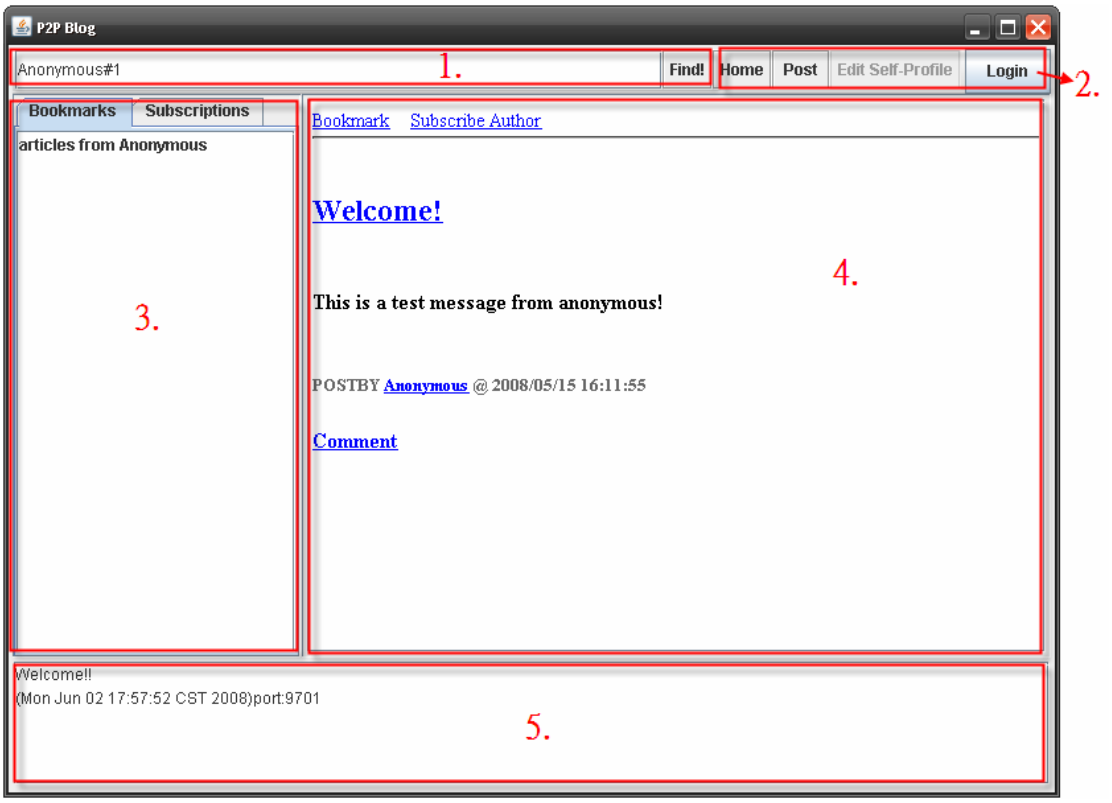

Figure 5-1 Screenshot

This is a picture of our graphical user interface. There are several areas in the figure. Area 1 is an input box for searching; users can input an exact key of a data and find out the information. Moreover, it is the area for display the key. For example, while users link to an article through bookmarks, it shows the key of the article. And we can see that the Figure 5-1 is showing all the articles of "Anonymous".

Area 2 has several action buttons. "*Home*" can link to the homepage of the user. If the users are not logged, it redirects to the blog of Anonymous. "*Post*" can publish articles. "*Edit Self-Profile*" can edit information of self account. In addition, the user

in Figure5-1 isn't logged, so "*Edit Self-Profile*" button is disabled. The detail functions and the "*Login*" button will be introduced in section 5.2.

Area 3 presents the information of bookmarks and subscription list. Users can choose the view by tab controller, and it is in the view of Bookmarks in Figure 5-1. There is a bookmark, "articles from Anonymous", in the bookmark list.

Area 4 is the Presentation Canvas (in section 4.7.2) and is responsible for how to display data. There are two hyperlinks in the top, and they are obvious instances to do actions by hyperlinks. At present, users can click these two hyperlinks, bookmark and subscribe author. Moreover, Area 5 presents messages of program, and these messages are also logged in the system log file.

All of the functions about blog system will be introduced in section 5.3.

5.2 Login with OpenID

While clicking the "*Login*" button, users can see our embedded web browser as Figure5-2. It may show no pages because of less time to start the web server, and users can press "*Login Page*" or "*Reload*" button for reloading web pages. Then users can input their OpenID in input box.

Figure 5-2 Login Page

After users key in the OpenID, the web browser links to the website of OpenID Provider as Figure5-3. For example, the OpenID for testing is *dcslab.pip.verisignlabs.com*. After the OpenID Provider confirms the certification, the web browser is closed.

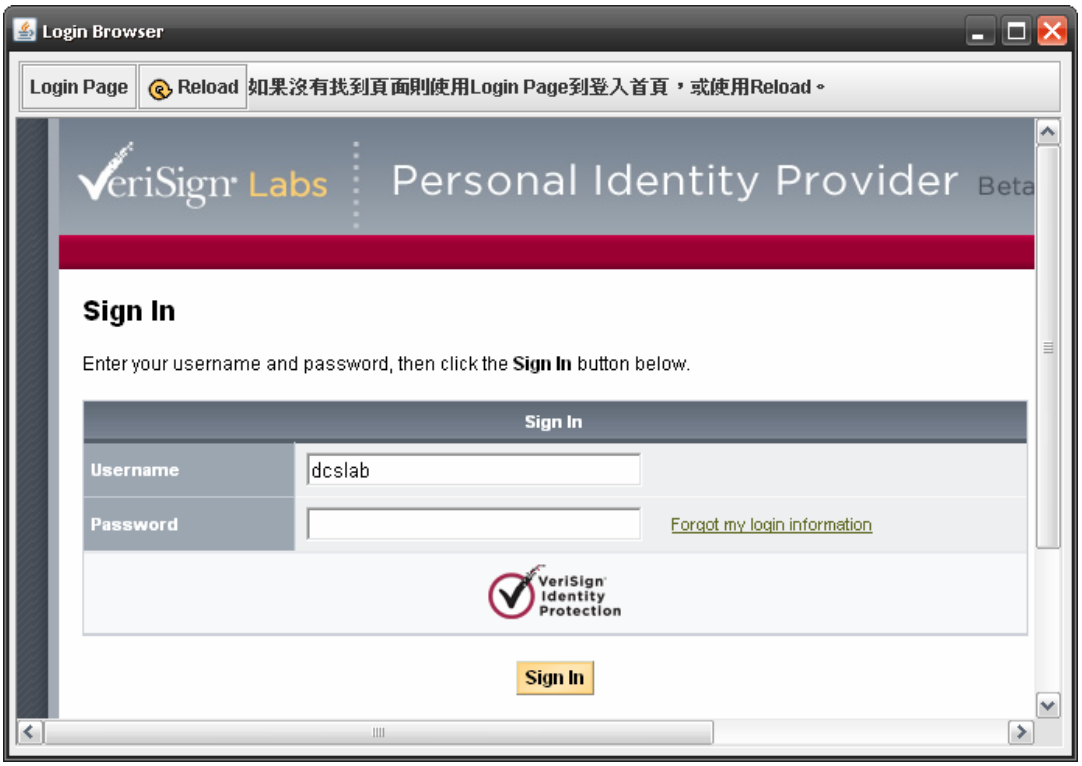

Figure 5-3 An OpenID provider website (Personal Identity Provider)

Then, users come back the main program, and can press "*Home*" to go to their blogs if they logged in successfully. And they can edit self-profile, as shown in Figure5-4. It can add others if adding other personal setting or information in the user's profile in the future. At present users can get the numbers of articles, delete subscriptions, and backup information.

Figure 5-4 Edit Self-Profile Form

5.3 Functions of Blog System

5.3.1 Post

We can see the frame for posting if we press the "*Post*" button. While users are not logged, the post identity just can be "Anonymous" that can be used for everyone to post an article and is anonymous. If users are logged, they can choose "Anonymous" or their identities, as shown in Figure 5-5. And the articles posted by "Anonymous" are classified to the blog of "Anonymous".

Figure 5-5 Post Form

5.3.2 Bookmark and Subscribe

We know that the there is a bookmark hyperlink in the top of the Presentation Canvas, as shown in Figure 5-6. It can bookmark a page to recover the difficulty of input an exact key. The bookmark list in left of the program can show these bookmarks. And users can double click bookmarks in the list to link to the pages, or choose one and press Delete in the keyboard to remove the bookmark.

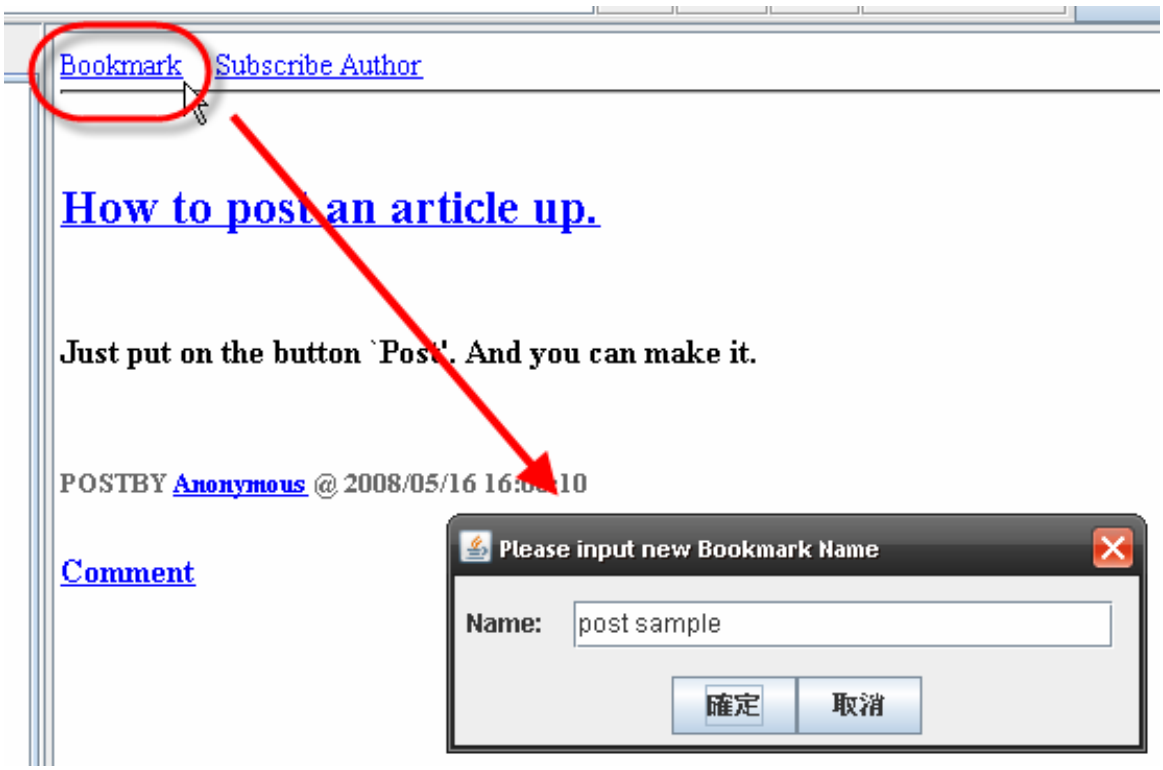

Figure 5-6 Bookmark a page

If users want to subscribe articles of an author, the steps are the same as bookmark. Users only click the Subscribe Author and give a name. But it can be used while users are logged, and the target of subscription is only authors. While there are new articles in the subscription, the subscription list will have changes, or users can press “Refresh Now!” to update the list right away. In addition, after users logging in, main program searches subscriptions if they have updates, and notifies those messages which are published when users are off-line. While users are on-line, the program shows messages in messages box (Area 5 in Figure5-1) if there are any updates in users’ subscriptions. Then users can check the subscription list and double click it to read. In the Figure 5-7 is the subscription list, the two items in the square box have new articles.

Figure 5-7 Subscriptions List

5.3.3 Comment

Users can leave comments in articles in blog system. Users can choose the truly identity while they are logged, but they also can choose other name, as shown in Figure 5-8. The difference can be seen in Figure 5-9. There are two comments in the figure, A and B mark the authors of the two comments. The first author A is announced by truly identity, and it is displayed by a hyperlink that visitors can click it to link to A's blog. The second author B is not truly identity, so it doesn't presented by a hyperlink. Even if users input their OpenID identity in other name box, the system doesn't validate it is a truly identity. Thus, it can avoid fake identity.

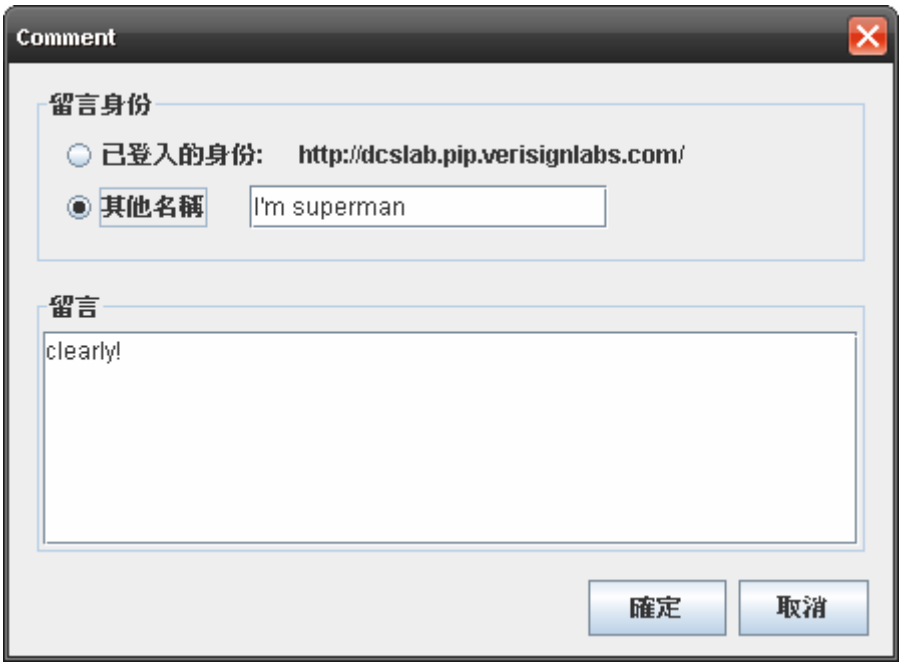

Figure 5-8 Leave a comment

Figure 5-9 Different comment identity

6 Comparison

Our system compares with relevant works as following table:

	P2P Blog with OpenID	**NUWeb**	**Traditional blog systems(via BSP)**
Architecture	***Distributed***	Distributed but a central portal site	Client/Server
Additional Software	Custom windows application	Custom windows application(NUBraim)	**none**
Registration	***OpenID***	To register on the central portal site	To register on BSP
Subscription	Scribe	RSS	RSS, Atom

Table 6-1 Comparison with NUWeb and traditional blog systems

Different 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, just like another BSP. In addition, only web browser is needed to visit present blog systems, but users have to install additional application to use NUWeb and our system. On the other hand, users have to apply for membership 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.

7 Future Works and Conclusion

7.1 Future Works

We shall settle following problems in near future:

1. The P2P Blog system only searches data by using exact key to find file identifiers because of the limitation of FreePastry. It is inconvenient to visit blogs. However, the system will be integrated with another research about P2P protocol in our laboratory, and it will improve the system about keyword search.

2. The traditional blog systems have many data of statistical analysis, e.g., popular articles, hot blog list, etc. It is difficult to count exact results from these distributed data on P2P network, so 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., picture, video.

4. The data on the P2P network are stored as files, and the file size can affect the system efficiency. Especially the system will have multimedia elements in articles. For the reason, we intend to divide large files into several fragments for system stability in later working. In addition, it will have kinds of methods according to different content types.

5. After the P2P Blog system, we intend to implement other applications on the P2P with OpenID platform. Internet auction is the next target we discussed. It has great challenge about Internet security and data synchronization.

7.2 Conclusion

The thesis proposes the P2P Blog system to solve the problem that service providers monopolize users' data, and users may suffer loss in non-self-consciousness and no-control. We have established the blog service on totally distributed architecture. Except for the advantage of protecting personal rights, building the system via P2P architecture is simpler than via client/server model because P2P technology already has some essential mechanisms like file-backup. However, P2P network needs enough users to keep the network operation steady. Besides, the system must notify users about the security problems on each host.

The platform which is used to build the blog system combines P2P network with OpenID authentication standard. It still remains advantages of entirely distributed network environment. Moreover, the P2P with OpenID platform can be utilized for other Internet applications as well, e.g., auctions, forums.

References

[1] Wikipedia(n. d.). Client-server. Retrieved May 25, 2008, from http://en.wikipedia.org/wiki/Client-server

[2] Wikipedia(n. d.). Peer-toper. Retrieved May 25, 2008, from http://en.wikipedia.org/wiki/Peer-to-peer

[3] A. Rowstron and P. Druschel, Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

[4] Ion Stoica , Robert Morris , David Karger , M. Frans Kaashoek , Hari Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, p.149-160, San Diego, California, United States, August 2001.

[5] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In Proceedings of ACM SIGCOMM 2001.

[6] Datta, A.K., Gradinariu, M., Raynal, M., Simon, G. (2003). Anonymous publish/subscribe in p2p networks. Proceedings of the 17th International Symposium on Parallel and Distributed Processing (pp.74.1). Washington, DC, USA: IEEE Computer Society.

[7] K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim. (2005). A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. Communications

Surveys & Tutorials, IEEE.

[8] Wikipedia(n. d.). Overlay network. Retrieved May 25, 2008, from http://en.wikipedia.org/wiki/Overlay_network

[9] Paul-Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, Wolfgang Nejdl. (2005) Designing Semantic Publish/Subscribe Networks Using Super-Peers. Semantic Web and Peer-to-Peer. Springer Verlag.

[10] JXTA. Retrieved May 27, 2008, from https://jxta.dev.java.net/

[11] Open Chord. Retrieved May 27, 2008, from http://www.uni-bamberg.de/en/pi/bereich/research/software_projects/openchord/

[12] FreePastry. Retrieved May 27, 2008, from http://freepastry.org/

[13] OpenID. Retrieved May 25, 2008, from http://openid.net/

[14] Wikipedia(n. d.). OpenID. Retrieved May 25, 2008, from http://en.wikipedia.org/wiki/OpenID

[15] Michael Arrington(Jan 17, 2008). Yahoo Implements OpenID; Massive Win For The Project. Retrieved May 25, 2008, from http://www.techcrunch.com/2008/01/17/yahoo-implements-openid-massive-win-for-the-project/

[16] Marshall Kirkpatrick(Feb 7, 2008). OpenID: Google, Yahoo, IBM and More Put Some Money Where Their Mouths Are. Retrieved May 25, 2008, from http://www.readwriteweb.com/archives/openid_big_companies.php

[17] 高有智(民 96 年 5 月 24 日)。Yahoo!無預警刪除同志部落格。中時電子報 http://blog.chinatimes.com/blognews/archive/2007/05/24/169035.html。

[18] nnickk(民 96 年 4 月 11 日)。你的智慧財產被侵犯了嗎--Yahoo!奇摩服務條款。民國 97 年 5 月 25 日，取自：http://blog.nnickk.com/read-524.html

[19] 陳英傑(民 97 年 4 月 9 日)。刪部落格批藍文 Yahoo!奇摩被指白色恐怖。民國 97 年 6 月 11 日，取自：自由時報

http://www.libertytimes.com.tw/2008/new/apr/9/today-p4.htm

[20] CNET 新聞專區：John Borland (民 95 年 2 月 7 日)。Blog 數量倍數成長。取自：http://www.zdnet.com.tw/news/software/0,2000085678,20104169,00.htm

[21] NUWeb。民國 97 年 6 月 11 日，取自：http://www.nuweb.cc/tw/

[22] Netcraft(Apr 1, 2008). Netcraft Web Server Survey. Retrieved May 26, 2008, from http://survey.netcraft.com/Reports/200804/index.html

[23] Wikipedia(n. d.). Blog. Retrieved June 11, 2008, from http://en.wikipedia.org/wiki/Blog

[24] Wikipedia(n. d.). Observer pattern. Retrieved June 11, 2008, from http://en.wikipedia.org/wiki/Observer_pattern

[25] Java. Retrieved June 11, 2008, from http://java.sun.com/

[26] Jetty. Retrieved June 11, 2008, from http://www.mortbay.org/jetty-6/

[27] OpenID4Java. Retrieved June 11, 2008, from http://code.sxip.com/openid4java/

[28] jdic. Retrieved June 11, 2008, from https://jdic.dev.java.net/

[29] jdom. Retrieved June 11, 2008, from http://www.jdom.org/