Building a Semantic P2P Scientific References Sharing System with JXTA*

Yijiao Yu and Hai Jin

Cluster and Grid Computing Lab,
Huazhong University of Science and Technology, Wuhan 430074, China
yjyu@mail.ccnu.edu.cn, hjin@mail.hust.edu.cn

Abstract. A semantic *Peer-to-Peer* (P2P) scientific references sharing system, SemreX, is introduced in this paper. As a P2P application system based on JXTA, the implementation technologies of the P2P communication layer of SemreX are illustrated, including the software architecture and the classes in P2P communication layer. Some lessons about JXTA, from the system development practices and software testing, are also concluded, such as peer automatic discovery, program robustness and bug report.

1 Introduction

Due to the scalable and robust advantages compared with the traditional *client/server* (C/S) communication model, P2P overlay network becomes popular in recent years. P2P application systems providing various services emerge sharply, such as decentralized file sharing, instant message, and games. JXTA is the first attempt to formulate the core P2P protocols and now is one of the most influential P2P protocols [1]. SemreX is a semantic scientific references sharing system we developed, which is based on P2P network and ontology technology, and there is a software module providing the basic P2P communication services for the application system. This paper aims at presenting the implementation of the P2P communication layer of SemreX, and discussing the experiences and lessons from the firsthand developing and testing practices. Especially, suggestions to JXTA are made to improve its usability and robustness.

The paper is organized as follows. Section 2 illustrates the use cases and goals of SemreX, and gives an overview of the software architecture. The design of the P2P communication layer, including the classes and the relationships among them, is presented in Section 3. Section 4 discusses major programming lessons from JXTA package and gives the related solutions. Section 5 concludes this paper with additional comments.

2 SemreX: A Semantic P2P Scientific References Sharing System

Searching scientific references from the Internet is a frequent and important behavior for researchers. It is common that a paper in a remote server is downloaded many

* This work is supported by National Basic 973 Research Program of China under grant No.2003CB317003.

X. Zhou et al. (Eds.): APWeb 2006, LNCS 3841, pp. 937 – 942, 2006.
© Springer-Verlag Berlin Heidelberg 2006

times by different researchers in a lab. If a paper is downloaded from WAN only once and shared in the LAN, it will reduce the unnecessary backbone network traffic and the searching time greatly. Furthermore, researchers in a lab usually have same or similar research interests. If a member reads a good paper and broadcasts this message, the whole research group will get benefit from it. To share and exchange references efficiently, SemreX is proposed and implemented. Bibster is a similar semantics-based bibliographic P2P system [2]. Compared with Bibster, SemreX is more concise because the size of source files of SemreX is much less than that of Bibster. In addition, SemreX supports heterogeneous data and permits the sharing of researchers' individual comments of papers.

First, we explain why P2P technology is chosen in SemreX. There is an assumption that each computer stores some papers, and the owner is eager to share them. With the C/S model, the papers in server are too many to be read and remarked. It is a fact that a great many papers are stored in the server, but only a small number of them have been read. On the contrary, the number of papers in a client is not very large and most of them have been read and remarked by the owner of the machine. With the single file server, it is difficult to add our comments to papers in server for the operation limitations, but we can record comments in our own computers. Then, we do not only share the papers, but also share the comments about papers.

Second, papers are shared not only with the traditional key words matching, but also with the semantic similarity. SemreX supports both automatic semantic information abstracting and manual remarking. To describe the semantic information in SemreX, ontology technology, such as "ACM Topic", is utilized. In addition, a proprietary ontology "SemreX: Reference" is created in our research, which gives some special information to bibliographies description and will be specifically discussed in other papers. The users of SemreX can categorize papers into an accurate sub-classification, and give evaluations and grades to papers. The manual evaluation information of references has higher priority than the automatic semantic information abstracting results in SemreX.

Third, lots of P2P applications have been developed and applied successfully in recent years [3][4][5], whose experiences will give some hints to our development.

The software architecture of SemreX includes the human-machine interface, P2P communication layer, semantic information abstracting and classifying, and semantic topology and routing. The human-machine interface receives user's commands, shows query results, and calls services provided by other modules. The automatic semantic information abstracting and classifying module categorizes each paper into a sub-classification of ontology, according to the information from the references data in the original PDF files. In addition, SemreX provides a graphical interface, which lets user categorize papers into sub-classification manually and add some private comments to each paper. Semantic topology and routing is for the future advanced research. The implementation of P2P communication layer are the focus of this paper.

3 Software Architecture of P2P Communication Layer

Building a stable and efficient P2P communication layer is the basic of SemreX. There are several considerations about P2P communication layer design, such as

which P2P communication protocol should be selected, which P2P software package should be utilized, and how to organize the classes in the P2P communication layer of SemreX. In our solutions, there are totally 12 classes and interfaces in P2P communication layer, and some of them are depicted in Fig. 1. Due to the page limitation, the member operations of classes are not listed in Fig. 1. In fact, the practical implementations of classes are more complicated than the descriptions in this paper.

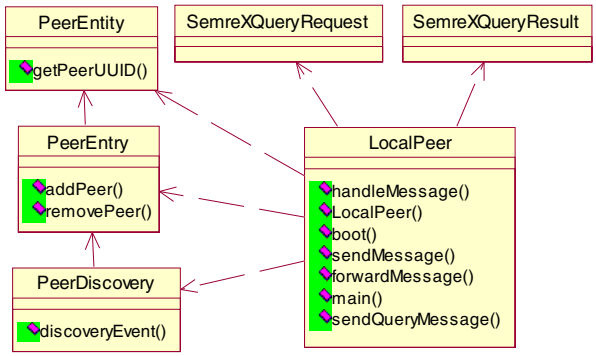

Fig. 1. Some classes and interfaces of the P2P communication layer

LocalPeer is the core of the P2P communication layer and is the most complicated class in this module. It extends from *Thread* of Java, which listens *InputPipe*, accepts query requests and query results. When receiving a message, the daemon thread extracts the message type, fires events to notify other modules of SemreX. For example, when a *SemreXQueryResult* message comes, *LocalPeer* produces a *SemreXQueryResultEvent* to fire the human-machine interface, and the query results will be shown. Because the communications among different modules are based on event-driven programming, the operations related to the communication layer are asynchronous. Furthermore, this class provides some public methods to other modules. For example, *sendQueryMessage* operation is executed when user wants to search.

PeerDiscovery is also a daemon thread, which monitors the changes of the peer group. When a new peer joins, it notifies the human-machine interface by adding description information about the new peer in the peer list. If a peer leaves, the peer information is removed. The operation *getRemoteAdvertisement* is executed periodically to get the change of peer group. Since *getRemoteAdvertisement* is an asynchronous operation, this class inherits the *DisvoveryListener* interface in JXTA. Each on-line peer in the peer group is recorded, and *PeerEntity* is proposed. Some public operations are defined, such as *getPeerName*, *getPeerUUID*. All the peer entities in the SemreX group are encapsulated in *PeerEntry*, which inherits the *Vector* class in Java.

All the exchanging data between peers are encapsulated into messages. At present, the important messages types are *SemreXQueryRequest* and *SemreXQueryResult*. Message in JXTA is represented in XML format, and JXTA package provides powerful information exchange abilities. These advantages of JXTA are utilized directly in SemreX. Some special and necessary *Tags* are added in message, such as *TypeTag*, *SourcePeerTag* and *DestinationPeerTag*. With these tags, the receiver can extract

some description messages to process the receiving data in an appropriate way. Due to the unreliable and unstable network environment, the delays of network operations are various. For the unstable network environment, all operations related to the P2P communication layer are asynchronous. For example, when a peer queries references from other peers, it only sends a query requests, and needs not wait for the responses.

4 Lessons from JXTA Programming

During the programming and testing, we get some lessons about JXTA programming, and a lot of time is spent to handle the troubles from JXTA. From the perspective of programming, JXTA is an influential network programming environment, but not friendly and powerful as expected. Furthermore, a bug in the core component of JXTA is found, and some key lessons from SemreX are discussed below.

4.1 Peer Discovery

SemreX is mainly tested in two different network environments. One is that all peers run in different computers in an Ethernet, and the other is in different Ethernets in a campus network. These peers in a LAN can discover each other automatically with IP multicast and broadcast in JXTA, and it is not necessary to specify a rendezvous peer [1][6]. We spend a lot of time in doing peer automatic discovery without rendezvous peer, however, the peers even in a computer are not able to discover each other at all. Then, a rendezvous peer is defined, and the address is published to other peers. After that, the error is removed. Unfortunately, the principle, peers in an unstructured P2P network are equal and decentralized, is violated. Requiring a rendezvous peer brings a great challenge to the application of SemreX, because nobody wants its peer to be the rendezvous peer. Furthermore, all peers before starting should know at least an IP address of a rendezvous peer, which makes the application not simple and easy as expected.

Some other P2P application systems based on JXTA are executed in our testing network environments, and they also have the same trouble about peer discovery. It is reasonable that the JXTA may not be able to provide the powerful peer discovery ability as it claimed. Through the discussion with other researchers, we know that the LAN supporting UDP multicast can discover peers without rendezvous peer. However, most of LANs in our campus do not satisfy this requirement.

4.2 Demo Programs Are Not Strong in Real P2P Application Environments

Programmers usually imitate the demo programs, but the testing results show that some demos in [6] are not robust. For example, the codes of getting an instance of *ModularSpecAdvertisement* are listed below, which is executed frequently in JXTA programming.

```
1: en = discovery.getLocalAdvertisements(
     DiscoveryService.ADV,"Name","JXTASPEC:JXTA-EX1");
   ……
2: ModuleSpecAdvertisement mdsadv =
            (ModuleSpecAdvertisement) en.nextElement();
```

In the software testing, exceptions are frequently thrown when the second code line is executed. There are several advertisement types supported by JXTA, such as *jxta:APA*, *jxta:RA* and *jxta:MIA*. The *ModularSpecAdvertisement* type is presented with *jxta:MSA*. According to the parameter *DiscoveryService.ADV* in the first code line, it is possible the local advertisements getting are not exactly the *ModuleSpecAdvertisement*. We judge that the instance of advertisement in *en.nextElement()* is not *ModularSpecAdvertisment* when exception is thrown. The codes getting the type of advertisement and judgment are inserted before the second code line, and the revised codes are as follows.

```
1: en = discovery.getLocalAdvertisements(
     DiscoveryService.ADV,"Name","JXTASPEC:JXTA-EX1");
   ……
2: Advertisement adv = (Advertisement)en.nextElement();
3: if(adv.getAdvType()=="jxta:MSA")
4: {
5:   ModuleSpecAdvertisement mdsadv =
            (ModuleSpecAdvertisement) adv;
6: }
```

With revised codes, the exceptions are never thrown and the overall P2P communication layer of SemreX becomes stable. From this case, it is clear that the JXTA demo programs are not strong enough for the real P2P application environments, and the simplicity brings troubles to the JXTA programmers. There are other similar cases in [6] and these demo programs should be improved for future use.

4.3 A Bug Report About *setPipeAdvertisement*

At the beginning of SemreX development, *JXTA-lib-2.3.4.jar*, released in March 2005, is utilized. While in software testing, our programs always stopped because of the *stack overflow* exception. At first, we guess there are some mistakes in our codes, and check them carefully. We are surprised to find that even the demo programs from [6] also produce the same exception. Finally, the bug is located at *setPipeAdvertisement*, a member operation of the *ModuleSpecAdvertisement*. The instance of *PipeAdvertisement* is produced by JXTA, but it leads to *setPipeAdvertisement* exception. *setPipeAdvertisement* is a frequent operation of P2P system, however there is a bug. To avoid this bug, the latest JXTA 2.3.4 is replaced with JXTA 2.3.3. Then recompiling the programs and running it, we can see this operation is correct. It is impossible to avoid bugs in software development; however, a bug is not allowed in frequent operations. In this case, the latest version of JXTA may not be tested thoroughly.

Compared JXTA-lib-2.3.3 with JXTA-lib-2.3.4, it is clear that *jxta.jar* and *jxtaext.jar* are revised in March 2005. The classes related to *setPipeAdvertisement* are packaged in *jxta.jar* and maybe the latest revision leads to the reported bug. The source codes of JXTA are not very difficult, and reading the source codes is helpful to understand JXTA and to build our application systems.

5 Conclusion

The design and implementation of the P2P communication layer of SemreX are presented in detail. The experiment results show that it is a simple but effective solution. The experiences and lessons listed in this paper are expected to give some suggestions to other developers with JXTA. With these firsthand experiences, JXTA should be stronger and easier to be utilized.

In the future, the semantic topology should be studied deeply and implemented with the semantic routing. In the prototype system design, only parts of P2P core services of JXTA are used, and the sophisticated services provided by JXTA should be utilized to accelerate our system. To realize a powerful semantic P2P system, some semantic information of each peer should be applied in communication layer to improve the query performance when the number of peers is very large.

References

1. L. Gong, "JXTA: A Network Programming Environment", *IEEE Internet Computing*, 3 (2001): 88-95.
2. P. Haase, J. Broekstra, M. Ehrig, et al., "Bibster - A Semantics-Based Bibliographic Peer-to-Peer System", *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 1 (2004): 99-103.
3. C. Parker, S. A. Collins, and D. C. Clear, "Building Near Real-Time P2P Applications with JXTA", *Proceedings of 2004 IEEE International Symposium on Cluster Computing and the Grid,* (2004): 338-345.
4. E. Halepovic and R. Deters, "Building a P2P Forum System with JXTA", *Proceedings of the Second International Conference on Peer-to-Peer Computing,* (2002): 41-48.
5. A. Zunino, C. Ciminiera, and L. Ciminiera, "A distributed JXTA-based architecture for searching and retrieving solar data", *Future Generation Computer Systems*, 3 (2005): 349-359.
6. Sun Microsystems, "JXTA v2.3.x: Java™ Programmer's Guide", 21 Jan. 2005, http://www.jxta.org.