# Design and Implementation of a P2P Shared Web Browser Using JXTA

Mikito Nakamura, Jianhua Ma, Katsuhiro Chiba,
Makoto Shizuka and Yoichiro Miyoshi
Faculty of Computer and Information Sciences
Hosei University

## Abstract

*The most shared applications use the client/serve model in which, however, a server is usually very complex and heavy since all of group managements are done by the server, and sometimes becomes a communication bottleneck as all of date exchange among group members are mediated via it. To solve the above problems, our shared browser adopted a pure peer-to-peer (P2P) architecture without using any server. A group member or a device called a peer dynamically finds other peers via distributed searching, and directly exchange data with other peers. It supports not only sharing a web document in a peer group but also synchronously viewing the document and manipulating the browser with further support of some group users' awareness information like a user's moving a cursor, entering a new URL and clicking a hyperlink. It is implemented using JXTA protocols and Java programming language. To make the system applicable over the Internet across firewalls and NATs, the HTTP protocol can be used to transfer data via a pipe, a communication mechanism in JXTA.*

***Keywords**: Peer, P2P, shared browser, JXTA, pipe, group management, awareness*

## 1. Introduction

There are two types of people's activities: an individual one in which a person works alone without interaction with others, and a group one in which a group of people work together via information share and exchange. The group activities can be conducted either asynchronously or synchronously among group members. However, the most web browsers including Internet Explore and Netscape Navigator only support an individual to get data/file stored in web servers from the Internet and a group of users to asynchronously shared web documents. They have little or almost no function to support the synchronous group activities, such as group meeting, group tour and so on over the Internet. This paper presents our research on designing and implementing a shared browser that enables multi-users, even behind firewalls or NATs (Network Address Translation) to synchronously share web documents on the Internet.

The shared browser can be seen as one kind of collaborative systems or shared applications, which are customarily classified into two categories, screen sharing and event-driven sharing [1]. Our browser is an event-driven sharing system that makes it better to reduce data communication amount, support effective group floor control, provide certain awareness formation, etc. Similar as our previous client/server based browser [2], the present system enables a group of users share not only a web document but also its viewing and operations.

Although some synchronous shared web browsers like collaborative viewer [3], mwTour [4], webcast [5], digital lecture board [6] and VCR shared browser [2] have been developed in recent years, all of them were implemented based on a client/server architecture. They rely on a complex server to manage groups and users, and pass shared data among browser clients. Due to this the sever load is very heavy and its response to the clients is relatively slow. Such a client/server based shared browser system is fully depended on its server, thus if the server has some trouble, the whole system might be down.

To overcome the problem in those client/sever architecture based shared browsers, we have adopted a pure peer-to-peer (P2P) architecture [7,8] to develop our shared browser system that does not use any server at all. Each browser's client, called a peer, is able to directly connect to other peers and communicate with them. Our system is implemented using JXTA that provides a set of protocols for creating P2P application [9]. Because of platform and transport independent features with JXTA as well as our system implementation with the JAVA program language, the P2P based shared browser can be used in any machine and over any network environment. Furthermore, by using JXTA, it provides flexible, scalable and secure group management.

There are two types of P2P architectures. One is called the hybrid P2P architecture that a server is used for providing an index of resources and peers' information by which a peer can first know other peers and then directly exchange data or share the resources with them. Napster [10], SETI@home [11] and Groove [12] were based on this architecture. The other architecture is called pure P2P in which there is no server and peers search for other peers and resources via a distributed-search mechanism. Gnutella [13] was based on this architecture. Our shared browser has adopted this pure P2P architecture, thus it does not need to prepare any server for mediating data among group peers' browsers.

The rest of this paper is organized as follows. We first describe JXTA protocols in next section and then discuss design and implementation of the P2P shared web browser in detail in Section 3. Section 4 gives system experimental environments and its tested results. Conclusions and future work are drawn in the last section.

## 2. JXTA

Our shared web browser is developed using JXTA, an abbreviation for Juxtapose. The JXTA project began in the summer of 2000 as a Sun Microsystems research project. Its aim is to examine the potential of P2P [14].

JXTA is a set of protocols for creating P2P applications. It provides a simple and generic P2P platform to host generic network services, while most other P2P applications are built for delivering a single type of network service (Napster for music file sharing and Gnutella for generic file sharing) by using a specific protocol and infrastructure which isolate their users from other P2P applications. Main JXTA features are the followings [15]:

- JXTA is programming-language neutral. Although the first implementation of JXTA is built with Java, implementations in other languages such as C/ C++ are being developed.
- JXTA is platform-independent. Any computer capable of communicating over a network and sending XML messages can use JXTA. Mobile devices, servers, desktop computers, and other devices can all communicate and use each other's services by employing JXTA as a hub platform.
- JXTA is transport-independent. It can be implemented on top of TCP/IP, HTTP, Bluetooth, and many other protocols. This means that a system built on the top of JXTA functions will be in the same fashion when the system is expanded to a new networking environment or to a new class of devices, as long as there is a transport protocol handler for the new protocol.

### JXTA Architecture

JXTA architecture consists of three layers: JXTA applications, JXTA services and JXTA core as shown in Fig. 1 [8].

- ***JXTA Core***

  The JXTA core provides core support for P2P services and applications. In a multi-platform, a secure execution environment, mechanisms of peer groups, peer pipes, and peer monitoring are provided. The peer groups establish a set of peers within a peer group with mechanisms to create policies for group creation and deletion, membership, advertisement, discovery, communication, security, and content sharing. The peer pipes provide communication channels among peers. Messages sent in peer pipes are structured with XML, and support transfer of data, content, and code in a protocol-independent manner — allowing a range of security, integrity, and privacy options. The peer monitoring enables controls of the behavior and activity of peers in a peer group and can be used to implement peer management functions including access control, priority setting, traffic metering, and bandwidth balancing.

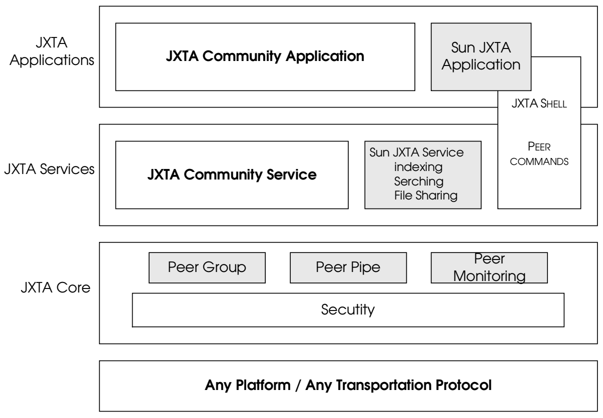


**Figure 1. JXTA architecture**

- ***JXTA Services***

  JXTA services expand upon the capabilities of the core and facilitate application developments. Facilities provided in this layer include mechanisms for searching, sharing, indexing, and caching code and content to enable cross-application bridging and translation of files. Searching capabilities include distributed and parallel searches across peer groups that are facilitated by matching an XML representation of a query to be processed with representations of the responses that can be provided by each peer. These facilities can be used from simple searches like searching a peer's repository to complex searches of dynamically generated content that is unreachable by conventional search engines.

- ***JXTA Applications***

  JXTA applications are built using the peer services as well as the core layer. The JXTA project's philosophy is to support the fundamental levels broadly, and rely on the P2P development community to provide additional peer services and applications.

### JXTA Protocols Used in Our System

Our browser incorporates several JXTA protocols [16]:

- Peer Discovery Protocol (PDP) - PDP allows a peer to discover other peers and peer group advertisements. This protocol is used to find members who use our browser.
- Peer Membership Protocol (PMP) - PMP allows a peer to join or leave a peer group, and to manage membership configurations, rights, and responsibilities.
- Pipe Binding Protocol (PBP) - PBP allows a peer to find the physical location of a pipe, and to bind a pipe endpoint to a physical peer endpoint transport.

## 3. P2P Shared Browser System

### 3.1. System Overview

As shown in Fig. 2 this application consists of seven parts.

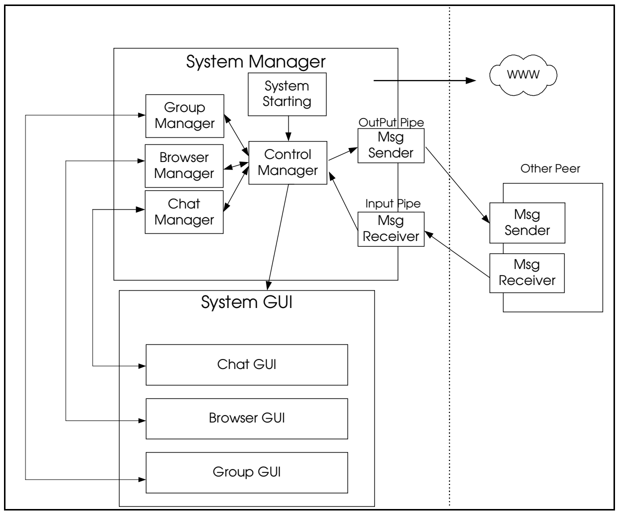


**Figure 2. Overview of system modules**

- *System Starting* - It is for network configuration, security configuration and connection to a JXTA generic net group.
- *Group Manager* - It deals with group search, creation, join and leave.
- *Chat Manager* - It is to control sending and receiving chat messages.
- *Browser Manager* - It deals with basic browser's functions like hyperlink, and browser's operations like sharing a telepointer.
- *Control Manager* - It controls the above three managers, GUI, and pipes.
- *System GUI* - It includes Chat GUI, Browser GUI, and Group GUI.
- *MsgSender* and *MsgReceiver* - They are implemented using JXTA propagate pipes. As shown in Fig. 3, the propagate pipes connect one output pipe to multiple input pipes. Messages flow from the output pipe (the propagation source) into the input pipes in the same peer group. *MsgSender* acts as an output pipe and *MsgReceiver* acts an input pipe.

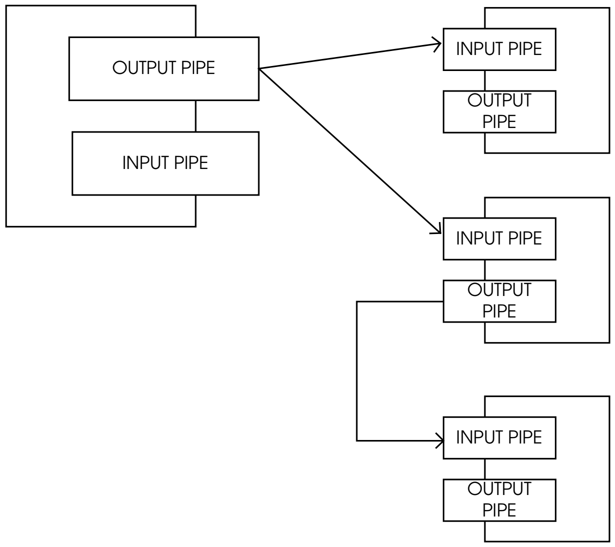


**Figure 3. JXTA propagate pipe**

### 3.2. Group Manager

This shared browser is intended to synchronously browse web document among multiple users in a group, thus each user must first find available peer groups and join a group to share information with other. The *Group Manager* provides functions of group search, creation, join and leave, and also manages peer groups via group GUI as shown in Fig 4.

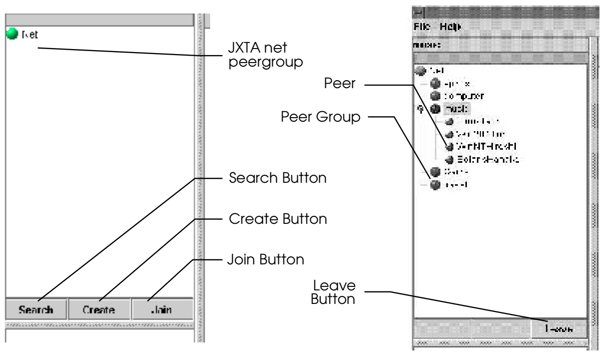


**Figure 4. Group GUI at the start (left) and after searching and joining a group (right)**

**Group Search Operation**

In older to find peer groups, that is, find XML based a peer group advertisements shown in Tab. 1, a group manager send discovery request messages to each immediate neighbors via a discovery service and put their discovery responses with advertisements to local storage. The peer group advertisements includes the information of the unique peer group ID "<GID>", group name "<Name>", a module specification ID "<MSID>" that describes all of the peer group services defined in this peer group and description "<Desc>". The discovery scope can be narrowed down by specifying an attribute and value pair. Therefore, advertisements that match the attribute and value will be returned. After the group search, a list of existing groups appears in the group GUI as shown in Fig. 4.

**Table 1. Group advertisement**

```
<?xml version="1.0"?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta="http://jxta.org">
     <GID>
        urn:jxta:uuid-7605413D77744F6486F194B8678926E402
     </GID>
     <MSID>
     urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000010306
     </MSID>
     <Name>
         test
     </Name>
     <Desc>
         SharedBrowser
     </Desc>
</jxta:PGA>
```

### Group Creation Action

By clicking the *Create* button in the group GUI in Fig. 4, group creation action occurs. To create a new group, a user needs to give a group name. After name is entered, it will be checked if it already exists by sending discovery request messages. If it doesn't exist, the new group with the name will be created. First, *ModuleImplAdvertisement*, which contains entries for all of the core peer group services, is prepared and then, the new peer group is created with this *ModuleImplAdvertisement* and then appears in the group GUI.

### Join Group Action

The JXTA membership service is used to apply for a peer group, join a peer group and leave from a peer group. The membership service allows a peer to establish an identity within a peer group. In order to join a peer group it needs to select a group by clicking the group name on the group GUI panel shown in Fig. 4. The membership service is extracted from this selected group. When a peer joins a peer group, other peers in the same group send related information like a current browsing URL, window size and scrollbar position to the new peer.

### Leave Group Action

First, a *logout* command, after clicking the leave button in a group GUI, is sent to other peers of a same group through the output pipe. Next, the membership service discards the identity of this peer that is attached to correspondent membership service and the current identity reverts to the "nobody" identity.

## 3.3. Shared Web Browser

As shown in Fig. 5, the shared Web browser consists of 5 main modules:

- *Browser Manager* - it deals with browser's event captures and web data transfer.
- *Browser Panel* - it is a graphic user interface of the shared web browser.
- *Control Manager* - it is used to transfer data between the browser manager and pipes.
- *Msg.Sender* - it creates an output pipe and sends browser's event commands or URLs to other peers through the pipe.
- *Msg.Receiver* - it creates an input pipe and receives browser's event commands or URLs from other peers.

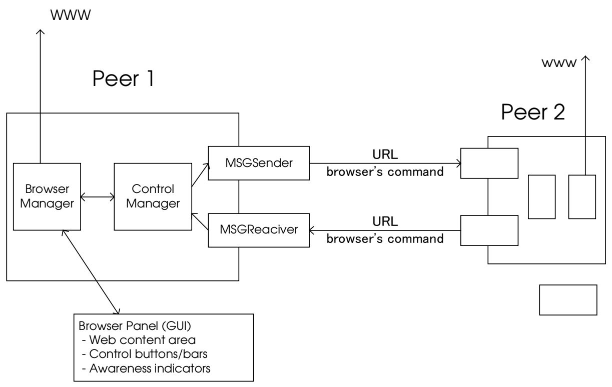
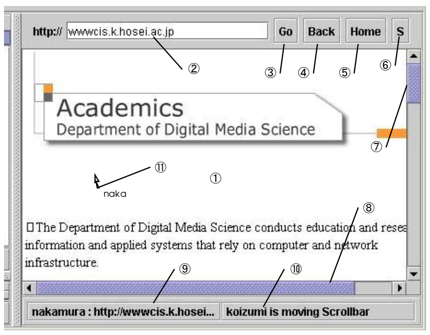


**Figure 5. Web browser function modules**



The Department of Digital Media Science conducts education and resea information and applied systems that rely on computer and network infrastructure.

nakamura : http://wwwcis.k.hosei...
koizumi is moving Scrollbar

| | |
|---|---|
| ① | Web content area |
| ② | URL field |
| ③ | Go button |
| ④ | Back button |
| ⑤ | Home button |
| ⑥ | S button: state control |
| ⑦ | Shared vertical scrollbar |
| ⑧ | Shared horizontal scrollbar |
| ⑨ | URL information panel |
| ⑩ | State Information panel |
| ⑪ | Telepointer |

**Figure 6. Web browser's GUI**

As shown in Fig. 6, the shared web browser provides the following main functions:

- Sharing browsing contents
- Sharing browser's display
- Sharing user's awareness information

**Operations of sharing browsing contents**

At present, this application provides the following shared operations concerning browser's contents:

- Entering a URL from a URL field
- Pressing navigation buttons, such as *Go*, *Back* and *Home*
- Clicking a hyperlink in a web page to go to another page

Above operations are realized by exchanging a URL address with other peers through the JXTA propagate pipe. The group manager gets URL address from a URL field, hyperlink or navigation button, and passes them to *MsgSender*. In the *MsgSender*, the URL address is converted to a pipe message with an element name "*WebURL*" and a message type value "2" as shown in Tab. 2. Conversely, when *MsgRceiver* receives a message of which message type value is "2", the current browsing URL address is gotten from this message. Then it is passed to the browser manager to access a correspondent web server and request the associated web file.

**Table 2. Message command types and formats**

| Message Type value | Element name | Content of element |
|---|---|---|
| 0 | Null* | Logout message |
| 1 | SenderMessage | Chat message |
| 2 | WebURL | Web URL |
| 3 | ScrollHValue | Horizontal scrollbar position |
| 4 | ScrollValue | Vertical scrollbar position |
| 5 | X, Y | Mouse position |
| 6 | Null* | Calling parent mode |
| 7 | WinHeight, WinWidth | Windows Size |
| * Use only sender's name | | |

**Operations of sharing browser's display**

This application provides the following shared operations concerning browser's display:

- Synchronized scrollbar movement
- Synchronized window's size change

The above two functions allow adjusting visible part of a large web page. A dynamically captured scrollbar position and window size are sent to other peers through the output pipe with message type command: 3, 4 or 7 as shown in Tab. 2.

**Operations of sharing user's awareness information**

This application provides the following functions concerning user's awareness information:

- Telepointer, i.e., cursor of a user
- URL information panel
- State information panel

The telepointer function provides two modes, parent mode and child mode shown in Tab. 3, to control whose cursor will be displayed in the shared browser among all group members. The shared telepointer is intended to provide awareness information of parent user's actions by showing the movement of parent mode user's cursor in a browser area. Users can change from a child mode to a parent mode by clicking the *S* button. In the parent mode, the captured cursor position will be sent to others. A user in the child mode can only see parent user's cursor. Currently, this application allows only one user to become the parent mode, that is, when one user pushes S button, all other users become the child mode automatically. A telepointer position is sent to others through the output pipe with message type command: 5 as shown in Tab. 2. The shifting to parent mode is carried out by sending a message of the type 6 command.

**Table 3. Telepointer mode**

| Mode | Telepointer Action |
|---|---|
| Parent mode | Move mouse pointer and send captured pointer position. |
| Child mode | Receive pointer position |

The URL information panel shows who generates an event of a URL address of a browsing content, and the state information panel shows who generates events related to a sharing browser's display and a telepointer.

## 4. System Evaluation

**System Working Environment**

To use the shared browser system, each user's machine needs to install JDK 1.4.0 and JXTA stable version (JXTA Project Stable Builder released in September of 2002). The shared browser has been tested and proved to work using four computers with the OS of Windows 2000, Windows NT, Solaris, and Turbo Linux. They are connected by TCP/IP as shown in the Fig. 7. Because of platform independence and transport independence of JXTA, the system also works in other network environments including the Internet. Of course, further tests are necessary.

**Experiment Results**

The experiments show that the shared web browser works well in above environments. Four users could share web browsing contents without setting up any server. A user could search all groups created by others. By the sharing functions, the shared browser enabled all users to browse a same web content synchronously. The telepointer allowed a child user to know information that a parent user points out. However this browser can't show contents written in JavaScript, Java applet, and other script languages in a HTML file, due to limited functions

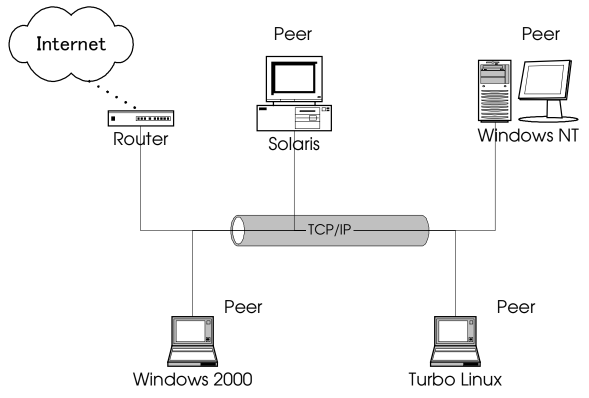


**Figure 7. Evaluation environment**

provided by the Java JEditPanel component. Using existing web browser's engines like Internet explore or Netscape Navigator might solve this problem. The browser can only share web documents in a web server and can't share a HTML file located in a peer's machine if it has no web service. The current prototype has not provided enough floor control mechanisms and thus conflict browser operations often occur when user's number in a group become large. One example of the conflict operation is like that a user moves a vertical scroll bar up, and simultaneously another one moves the bar down. Another main reason having the conflict operations is relatively limited user's awareness information provided by the prototype.

## 5. Conclusions and Future Work

This paper describes our shared web browser with emphasis on implementation using a pure P2P architecture and the JXTA technology. Different from current other client/server model based sharing browsers, it enables a users to share web information and communicate with others without setting up any server. JXTA technology allows us to search and make groups without asking the address of other peers to a central server. The shared browser's viewing functions and shared manipulations realize WYSWIS. As a result, the P2P shared web browser allows us to navigate web pages among group users and do collaborative work such as discussion, presentation and so on in a group by sharing information on the web and the Internet.

Because it is a relatively new P2P based shared browser, lots of research and developments should be done in the future. We are continuing to improve the system in the following aspects.

A) The browser's performance needs to be improved because the current prototype is not able to show pages including JavaScript, Java applet, and some others due to limitations of the JEditPanel package.
B) It is also necessary to add some new function to directly get web page data from other peers. A possible solution is to use a customized proxy web server to handle communications of peers.
C) Some satisfied floor control mechanisms should be incorporated into the system.
D) Rich user's awareness information must be provided.
E) Support of shared free drawing and other files like text, pdf, image, music, etc.

## Reference

[1] J. B. Begole, *Usability Problems and Causes in Conventional Application-Sharing Systems*, http://simon.cs.vt.edu/begolej/Papers/CTCritique, 1999.
[2] J. Ma, R. Huang and A. Kondo, *A Shared Browser for Synchronous Web Navigations by Multi-users*, in Proc. Int. Conf. On Information Society, Nov. 2000
[3] S. Konomi, Y. Yokota, K. Sakata and Y. Kambayashi, *Cooperative View Mechanisms in Distributed Multiuser Hypermedia Environments*, in Proc. 2nd IFCIS Int. Conf. On Cooperative Information Systems Kiawah Island, South Carolina, 1997.
[4] T. Jiang and M. Ammar, mwTour: *A Dynamically Controlled Scalable Web touring System Using Reliable Multicast Delivery*, http://www.cc.gatech.edu/computing/Telecomm/mwTour/
[5] NCSA, *Collaborative Document Sharing via the MBONE*, http://www.ncsa.uiuc.edu/SDG/Software/Xmosaic/CCI/webcast.html
[6] W. Geyer and W. Effelsberg, *The Digital Lecture Board – A Teaching and Learning Tool for Remote Instruction in Higher Education*, In Proc. ED-MEDIA'98, World Conference on Educational Multimedia and Hypermedia, Germany, June 1998.
[7] Andy Oram ed., *Peer-to-Peer Harnessing the Power of Disruptive Technologies*, O'REILLY. 2001.
[8] M. Parameswaran, A. Susarla, A. B. Whinston, *P2P Networking: An Information-Sharing Alternative*, IEE COMPUTER, July 2001.
[9] Li Gong, *Project JXTA A Technology Overview*, Sun Microsystems, 2001.
[10] Napster, http://www.napster.com
[11] Kelly Truelove, *Gnutella: Alive, Well, and Changing Fast*, Clip2 DSS, 2001,http://www.openp2p.com/pub/a/p2p/2001/01/25/truelove0101.html
[13] Groove Network, http://www.groove.net/
[12] SETI@HOME, http://setiathome.ssl.berkeley.edu
[14] Project JXTA, http://www.jxta.org/
[15] Project JXTA, *Java programmer's guide*, http://www.jxta.org/jxtaprogguide_final.pdf, Sun Microsystems, 2001.
[16] JXTA v1.0 Protocols Specification, http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html, Sun Microsystems, 2001.
[17] Sing Li, *Early Adopter JXTA,* Wrox press, 2001.