Distributed Development Process AstroGrid Experience for WS-Talk

Pedro Contreras, Fionn Murtagh, and Dimitrios Zervas
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
{pedro, fionn, dimitri}@cs.rhul.ac.uk

Abstract

The intent of this document is to contribute to the discussion about which development platform is better for the WS-Talk project. We present our experience with AstroGrid project, from which we think there are important lessons to be learnt.
AstroGrid has implemented a series of tools which facilitate collaborative work in geographically distributed locations such as news, forum and discussion pages, all of them inspired by the open source and licensing philosophy. Also in the middle-ware layer there is an important effort to build reliable and good code which can run over the Internet where Apache Software Foundation is used to provide a stable platform to deploy applications.
Later in this paper we approach our platform of development for WS-Talk and JXTA peer to peer technology as a possible layer of interaction with Web Services standards.

1. Introduction

The terminology "Grid Computing" has been widely adopted in reference to the use of distributed computer resources across a network, working in a co-ordinated way, and sharing not just information but its processor cycles.
Grid technology is applied in a number of scientific areas, from climate simulation to finance and astronomy just to name a few. In order to achieve these different Grid networks a series of guidelines, standards and protocols are needed, and a range of working group such as those affiliated to the Global Grid Forum [7] have been established.
The astronomical area is no exception to this trend: astronomers and computer scientists are working to ensure standards for a data-grid platform. This context is where AstroGrid is playing an important role.
The paper is structured as follows: firstly we present AstroGrid collaborative environment and its architecture. Secondly se introduce our platform in Royal Holloway dedicated to WS-Talk

project, we propose a platform of development, then we explores how this platform could fit with JXTA peer-to-peer protocols onto WS-Talk through Web Services standards.

2. AstroGrid Development Process and Collaborative Work

AstroGrid [1] is a £10 million project aiming to produce more economic, faster and effective astronomy through the generation of open standards and interfaces. The AstroGrid framework helps to enable different data centres across the United Kingdom to publish services and data into a data-grid infrastructure accessible on the Web. Such a framework also makes it easier for the different astronomical data centres to interact; it offers a wide range of analysis and visualisation tools through a common interface.
AstroGrid is not alone in its attempt to standardise access to astronomy information. It is part of a worldwide effort to transform astronomical data repositories into a world Virtual Observatory (VO), an organisation of which it is also a founding member. Such an initiative is led by the International Virtual Observatory Alliance [9] (IVOA) movement, which consists of around sixteen active international members.
The AstroGrid philosophy is one of an open collaboration of researchers and technologists. To allow these people to work in partnership effectively, a range of web-based tools is required. In such a collaborative software development project there are different obstacles to overcome before the collaborative development can become effective. Firstly, good communication is essential in many aspects. Developers need to be aware of information regarding the project as a whole, information regarding specific sections, and they need to be able to directly communicate with other developers. The latter is of high importance in the AstroGrid case. Developers working within the project are assigned tasks in so-called development cycles which are specified by astronomers through definition of science cases. Developers will work on a certain task or project module for a few months before being moved to another area. In some cases more than one cycle or iteration is necessary to achieve a goal. In any case task assignment is under constant review by the project leaders, with a view towards reallocating human resources when and where they are needed.
The benefit of an approach like this is a highly cross-trained development team. For example in the case of a member's absence another developer who has previously worked on that area can be assigned the task. The fact that, at some point, they would have worked in the particular area will mean they can quickly re-integrate themselves into the development team for the particular project module. However, adequate facility for communication must be in place for developers to be able to exchange ideas and to make and to respond to queries, so that the development process is as efficient as possible. One important issue in relation to this cyclic development process is development by second parties. As well as communication facilities there are other ways to ensure that the collaboration is as effective as possible.
Standards play an important role in facilitating code development by second and third parties. Standard forms of software design and standard coding styles all play a part in reducing development time. Standard software design methods will aid in code integration and coding style standards will help reduce the time required for second party developers to familiarise themselves with software. Standard forms of code documentation can also greatly aid second party developers.

2.1. AstroGrid Development Process

Support for second party development in AstroGrid can be classified into three areas. Firstly, there are document repositories, news, notification, and a discussion forum to facilitate commu-

nication between members of the project. Secondly, there are the standards for the design and implementation of software generated by the project. Finally a considerable amount of documentation is required once the software is working. This documentation includes documents for the actual code, integration documents and testing documentation. Then a range of tools is used to tackle the problems posed by AstroGrid's needs for effective collaboration.

2.2. Communication

To facilitate good communication practice any person with expertise or simply an interest in the Virtual Observatory concept, Grid technologies or e-Science is free to post or comment upon news articles, participate in the forum exchanges or post documents making use of the AstroGrid portal. The portal provides an access point to the various communication tools available. The Wiki pages provide support for the posting of relevant documents for the project. Wiki is a leading-edge, web-based collaboration platform targeting the corporate Intranet world. TWiki [26] fosters information flow within an organisation. Some Wiki features are: web browser, auto links, text formatting, search, e-mail notification, structured content, attachment, templates and skins, statistics and plug-in resources. Further to the Wiki pages the AstroGrid portal provides a forum facility which provides a number of grouped discussion areas. The groups deal with site-related topics, project-specific topics and general areas such as virtual organisations and Grids. To allow developers to be kept up to date with various information about articles posted, items of news, events and polls, a news section is available. The idea of a central portal providing access to the various communication platforms is essential to effective collaboration. There is no confusion over how to contact other members as all contacts are made through the use of one central site. Further to these web-based community-oriented communication tools, one-to-one contact is necessary in order to ensure that developers can carry out their tasks efficiently. For example, in the case of second party development the ability to contact a previous developer to query some aspect of the work greatly reduces re-development time. To facilitate the necessary direct communications between project members, project contracted people keep close contact through e-mail, telephone conferences, chat programs and regular meetings.

2.3. Code

Integration Code is tackled using Current Version System [5] to store, modify and keep track of code changes. This has various uses in software development practice helping returning developers to identify and understand changes that have been made since they last worked on a particular project area. JUnit [12] Test is used to prove code integrity and ensure quality of service. JUnit is a regression testing framework for code made with Java. Usually a unit test exercises some particular method in a particular context in order to prove that a specific piece of code does what it is supposed to do. Code bugs are tracked using Bugzilla [3], a "Defect Tracking System" or "Bug-Tracking System". Defect Tracking Systems support individuals or groups of developers to keep track of outstanding bugs in their software effectively. Once an error, misbehaviour or unexpected answer has been detected, this will be described and assigned to a developer in the form of a ticket which contains all the information related with the bug. When a piece of code is developed, in order for further work to be effectively carried out, it is essential that the product is of high quality. To provide standard coding practices a number of tools are used in the AstroGrid project. A range of free Integrated Development Environment (IDE) tools such as Eclipse [6] and JBuilder Foundation [11] are used during the development process. These, in conjunction with code design standards (i.e. UML), are needed to establish the basis on which to allow different people in different places to modify each others' code

easily, the philosophy being that if everybody is using the same tools to build a product the components of the product can be more easily modified or enhanced.

The main programming language used in AstroGrid is Java. Different standards like UML, XML and Web Services are used for the process design to maintain consistency. When implementing code in Java a coding standard is utilised, which allows different people to understand the work of each other, maintain coding style and consistency. This leads to a cleaner and easier coding process. AstroGrid code guideline standards are described in Vermeulen et al. [28], but a good set of standards can be found easily on the Internet.

2.4. Documentation

Code management and documentation is addressed mainly through the use of standard Java Docs and Maven [17]. Maven is a Java project management and project comprehension tool. It is based on the concept of a project object model (POM) where all the Maven objects are a result of a well defined model. Builds, documentation, source metrics, and source cross-references are all controlled by the POM. Maven aims to make the developer's life easier by providing a well defined project structure, well defined development processes to follow and a coherent body of documentation that keeps developers aware of what is happening within the project. Maven alleviates a lot of what most developers consider a problem. This is essential in projects where there are not many people dedicated to the task of documenting and spreading the critical information about the project, which is necessary in order to dedicate resources to other critical tasks as coding and coding testing.

2.5. Security

One problem not included in the above section is security and the identification of which information is publicly accessible and which is not. As AstroGrid is an open project, almost every document is publicly accessible through the Internet. One exception is the code, where there are restrictions in place as to who is authorised to upload and integrate code. In this case SSH [25] protocol is used to allow secure communication between developers and the server repository. Though remote login is the primary use of SSH, the protocol can be used as a general purpose cryptographic tunnel, capable of copying files, encrypting email connections, and triggering remote execution of programs. Currently AstroGrid uses SSH Version 2 which operates over TCP. In its simplest mode of operation, it connects to a server, negotiates a shared secret key, and then begins encrypting the session. A username and password are passed over the encrypted session and, if authenticated, the server starts a command shell over the encrypted session.

3. Architecture, Software Development, and Deployment in WS-Talk

Much in the vision of AstroGrid development process and software deployment, we present the platform for WS-Talk at Royal Holloway, which is based in open standard, and robust open sources project.

Programming and scripting language: Our primary programming language is Java. Well known characteristic of Java are that is free, platform independent, object oriented, network oriented and additionally is well establish in the programmers community, therefore a wide range of libraries available. Of course there are some characteristic involving security and robustness.
Additionally to the election of Java as programming language a lot of times the use of a scripting language to carry out day to day tasks is necessary, in such cases we are open to use them. Our inclination is to use *Phyton* or *Perl* scripting, mainly because they are available in an wide range of platforms. In the case of Unix/Linux platform *Bash* scripting may be used.

Apache Web Server: With a market share around 70% [18] Apache is by far the most used Web Server on the Internet today. The following list present some characteristic of Apache web server:

- Static and Dynamic web pages: Interaction with the most common dynamic content pages generation such as Perl, Phyton, Java Servlets and PHP.
- Highly configurable: Apache is built in base to a modular architecture, modules which can be added, removed or configured when they are need.
- Security: Several method of authentication are supported by the web server, being one of them SSL encryption.
- Portability: Apache run almost in any available platform today, and certainly is very well supported on Windows, Linux, Unix and OS/2.
- Licensing: All the apache software is distributed under ASF (Apache Software Foundation) licensing, which basically means that can be used by free, this subject to some small restrictions[1].

TomCat: In conjunction with Apache web server, TomCat is our Java Servlets and JavaServer Page container of election. TomCat is the official reference implementation for Servlet and JSPs, which means that we can be sure our code will run as much in TomCat container as in any another which accomplish with the standard definition.

Web Services: Pretty much based on XML standard, Web Services implement the following specifications: SOAP, WSDL, and UDDI. SOAP as messaging envelop, WSDL as description language for accessing services and UDDI as resources directory and locator.

Peer to Peer and JXTA: JXTA protocols implements a Peer to Peer network. Here again these protocols are platform independent, and very well supported in Java language. Later in this document we describe more in deep JXTA.

Is our understanding that the standards, languages and tool presented here make a consistent choose at the moment of developing Web Services for WS-Talk project.

[1]There is not the scope of this paper discus the license matter since it can be tricky and rather complicated but a good source of information can be found on the Open Source Initiative `http://www.opensource.org`

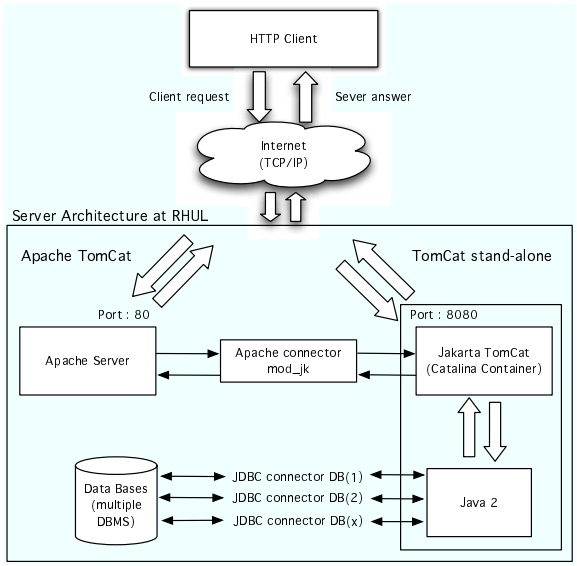

Figure 1. Deployment architecture Apache/Tomcat at RHUL

Figure 1 shows our platform of election for software deployment at the started point of the project. Later we should include to such scheme a Web Service "plug-in" and eventually a JXTA one.

4. Peer to Peer Architecture, JXTA, and Web Services

4.1. Network architecture

Network architecture is defined as the way in which two or more linked computers interact. Some characteristics include:

Topology: Geometry or shape in which the system is organized. It could be ring, star, bus, mesh, and tree. In addition, a topology could be either physical or logical.

Protocol: Defines a common set of signals and rules which are used by computers in the network to communicate with each other. One of the most popular protocols for LANs is Ethernet.

Architecture: Could be classified as client/server or peer to peer.

In our case the architecture is a main concern, topology is defined dynamically when we use JXTA or centralised when we use client/server topology like in Apache wen server case.
Next we describe the client/server and P2P architectures systems, explaining in detail the latter, which is the object of our study.

4.2. Client/Server architecture

In today's world client/server architectures are widely used. It was first mentioned in reference to personal computers in a network in the 80s, designed in response to new user needs (i.e. development of cost effective applications) and to improve mainframe centralized and time shared computing.
In a client/server model, the client is defined as a requester of services and a server is defined as a provider of these services. To carry out this task, server and client must speak the same language, which is defined through a set of rules called communication protocol. This schema is typically used on the Internet to run server-side applications such as Web servers, FTP server, Telnet server, e-mail server, chat server, etc. An advantage of such a system is the central control. It allows a system to be manageable, secure and coherent. On the other hand, as disadvantage we can mention the lack of self-organisation in case the server goes down. Therefore making a scalable, non fault, and tolerant system is restricted to the capacity of each server.

4.3. Peer to Peer architecture

Figure 2 shows a typical peer to peer (P2P) architecture (as we can see, centralized or client/server models could be considered as a particular case of a P2P system). In this network model any node (peer) may be interconnected. A peer may also operate as a consumer or provider. There does not exist a fixed server or client (without discarding the fact that eventually a system could take that form).

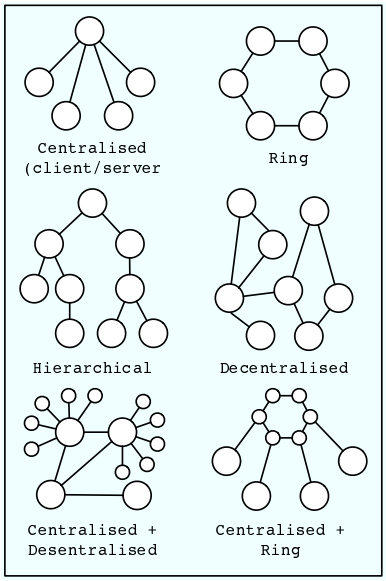

Figure 2. Peer to peer network organisation

As we can see a P2P system is extremely flexible and dynamic. The connections between peers as soon as they appear can disappear. Figure 2 does not show all the possible cases but

gives us a glimpse of what a P2P network could become. From their very nature P2P systems are characterized by being decentralized, highly autonomous and heterogeneous. Such systems present a series of problems and challenges for research. Some of most relevant are:

Scalability: Meaning how large a system could become and adapt to increases on demand. This point usually is mentioned as an advantage of decentralized systems over centralized. However, its bears it owns problems, i.e. the system could become so huge that it is manageable any more, there by losing sense and purpose.

Security: The central question here is whether the system can be trusted or not. This involves a series of concerns such as identification, information veracity and authentication.

Tolerance: In order to offer a fault proof service, the system should be able to respond smoothly to unexpected errors and failures. There are many levels of fault tolerance, as a basic one we can mention the ability to keep a system working in the event of power failure (which can be solved duplicating the system with mirrors). A more high level of tolerance refers to the systems capacity to respond acceptably, even if the system receives incorrect input data.

Ubiquity: This refers to how capable the system is to be present in different places at the same time. From the computer point of view it adduces how the system is accessible through different areas and devices.

Routing: Refers to the ability to pass messages from one peer to another. How are peers identified and organized? This is essential for resource discovery and information retrieval, and is closely related to the query system.

Queries: How could the system manage complex queries? An approach to solve this problem is using Metadata through XML and RDF standards in order to build a P2P semantic network where the peers are aware of the neighbourhood.

Flooding: This is a well known problem with P2P networks especially with message broadcasting in the system. The basic question here is how to produce a system capable of managing flooding whilst remaining stable.

Performance: Are the resources used efficiently? Could we improve performance? Which resources are sub utilised? Also we can include here how the resources are monitored and evaluated, i.e. traffic network monitoring, measuring and control.

Integration: P2P systems trend to coexist with other systems, i.e. GRID and Web Services systems. That is because such systems are designed to solve similar problems. Still there is not a clear picture about how these systems will merge together. It is an open question vigorously investigated by diverse research groups.

Evidently the above problems are not all the problems confronted by this area, but they give a general understanding of the issues that a P2P system confronts.

There are different approaches/architectures to implementing a P2P system. Among the most widely established we can mention CAN [22], Chord [4], Pastry [19], Tapestry [31] and JXTA [13]. All these systems approach in a different way the issues mentioned above. But generally all can be considered under the category of Distributed Hash Tables systems. There are also other systems like Jabber [10] which defines a protocol for message exchange. Gnutella [8]

and Kazaa [16] protocols, based on hierarchical networks with super-peer like topology for search and discovery, are other examples.
We have been working with P2P models through JXTA protocols in the past, and we expect to apply some of its functionality in the WS-Talk context. In the next section we briefly explain JXTA technology.

4.4. JXTA Technology

As we have seen in the previous section there are many different peer-to-peer system architectures, but at the conceptual level they can be presented as interactions of three layers components:

Core: Deals with the peer foundations, communication (such as routing), and other low level task.

Services: Provides services considered desirable but not essential to all P2P systems. Here we can mention searching, resources sharing and authentication.

Application: Any another service or application builds on top of service layer such as e-mail, instant messaging, and storage systems.

JXTA is an open source project (originally created by Sun Microsystems) which allows the implementation of P2P systems. It approaches the three layer [2] architecture as is shown in Figure 3.

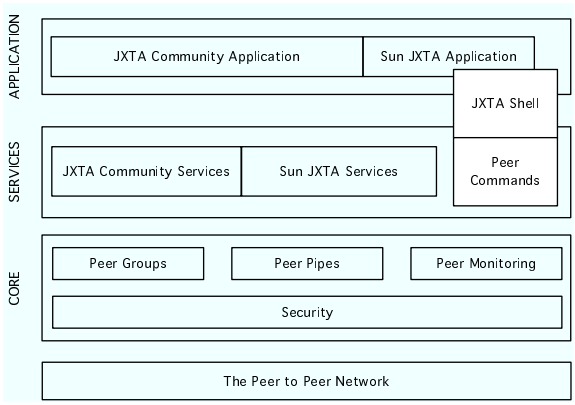

Figure 3. JXTA P2P software architecture

Allocated at the core layer, JXTA defines the following set of six language independent protocols [20] to provide a P2P infrastructure:

- Peer Discovery Protocol (PDP) : Resources search and advertisement
- Peer Resolver Protocol (PRP) : Generic query service
- Peer Information Protocol (PIP) : Net monitoring
- Pipe Binding Protocol (PBP) : Addressable messaging
- Rendezvous Protocol (RVP) : Propagation service
- Endpoint Routing Protocol (ERP) : Routing

JXTA has been designed with the objective of making it ubiquitous. Any device, including mobile phones, PDAs, pagers, electronic sensors through to desktop computers and servers, has the potential of being JXTA enabled. In order to maximize its platform independence and interoperability, standards like XML are at the very core of JXTA communications, implemented through an advertisement structure that names, describes and publishes resources like a peer, peer group, a pipe or a service. The issue of security is dealt with by using Transported Layer Security (TLS) and Digital Certificate and Certificate Authorities.

4.5. JXTA and Web Services

Web Services and JXTA are two technologies oriented to solve similar problems. Since the Internet has became more and more complex new technologies have been introduced to locate, retrieve and exploit information in a secure and high distributed environment.
Web Services describes a distributed computer model that differs from CORBA or Java RMI in the way that is centred in simple Internet-based standards defined by W3C [29] and other standards bodies heavily supported by industry. (On the other hand if we consider a JXTA service, it may include CORBA or RMI services and others such as WSDL and SOAP).
Figure 4 shows a basic Web Services stack based on this open standards where:

- Simple Object Access Protocol, SOAP [24]: is used as wrapper for XML messages providing a XML-RPC mechanism.
- Web Services Description Language, WSDL [30]: is used to describe services available.
- Universal Description Discovery and Integration, UDDI [27]: is used for listing of services and locator of resources for Web Services.

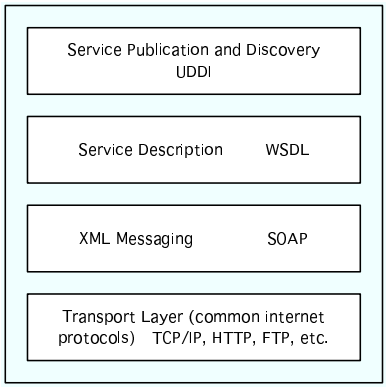

Figure 4. Web Services Stack

Where are JXTA and Web Services overlapping? That is not completely clear, mainly because the solution to specific problems with these two technologies could require slightly different approaches. Thus, in the literature there are different opinions about which technology superposes to the other. We find cases where the following situation is considered: JXTA as a component of Web Services, JXTA as extension of Web Services, JXTA superseding to Web Services, and JXTA and Web Services as convergent technologies.

Appropriate justification can be fund for each of these cases. But from our point of view these two technologies merge together more than one superposing to the other. As similarities we can mention XML's strong orientation to message passing and distributed computing oriented although JXTA technology is not yet as mature as Web Services and also is not as decentralised as JXTA.
There are some projects that have been already working on this integration. For instance we can mention; JXTA Bridge [14] project, which is designed to allow SOAP communication over the JXTA P2P network: or as shown in RegistryVous project [15] where a web service can be created in combination with JXTA; and finally Edutella/JXTA [21], a specially interesting case to analyse since it creates a P2P semantic web application which facilitates exchange of metadata based on the RDF [23] (Resource Description Framework) standard.

5. Service Orchestrator, Service Locator, and Active Transformer in WS-Talk

5.1. Service Locator

- JXTA Service design and implementation

5.2. Active Transformer

- Lexical Analyser
- Grammar Definition
- Construction and tune of Parser Libraries
- Semantic Web Definition
- XML Transformation
- Web Services Activation

6. Future Work

Short term tasks Software installation in the server and Tuning. (Apache, TomCat, Java, Ant, SSH, FTP, etc)

Creation of an XML transformer from free text

Research Topics:

- Web Semantic standards
- Automatic taxonomies generator
- Text clustering
- XML generation and transformation into Web Semantic standards

Web Services Deployment

Middle term goal: Research integration between JXTA and Web Services

7. Conclusions

This paper explores the current state of AstroGrid Virtual Observatory collaborative work, and the way in which the project is organised to reach its goals. Our chosen development platform for WS-Talk project is presented as well as peer-to-peer technology with special attention in to JXTA project. In the last section the steps for our future work is presented.

References

[1] AstroGrid. Astrogrid Portal, 2005. http://www.astrogrid.org.

[2] W. Brendon. *JXTA*. New Riders, 1st edition, June 2002.

[3] BugZilla. Bug-Tracking System, 2005. http://www.bugzilla.org/.

[4] Chord. Chord Project, 2004. http://www.pdos.lcs.mit.edu/chord/.

[5] CVS. Concurrent Versions System, 2005. https://www.cvshome.org/.

[6] Eclipse. Eclipse Foundation, 2005. http://www.eclipse.org/.

[7] G. Forum. Global Grid Forum, 2005. http://www.gridforum.org.

[8] GNutella. Gnutella, 2004. http://www.gnutella.com.

[9] IVOA. International Virtual Observatory Alliance, 2005. http://www.ivoa.net/.

[10] Jabber. Jabber Inc., 2004. http://www.jabber.org.

[11] JBuilder. Borland JBuilder, 2005. http://www.borland.com/jbuilder/.

[12] JUnit. Java™ Testing Framework, 2005. http://www.junit.org/.

[13] JXTA. Jxta Project, 2005. http://www.jxta.org.

[14] JXTA-Bridge. Jxta Bridge Project, 2004. http://relativity.yi.org/jxta-bridge/.

[15] JXTA-WSC. Jxta Registryvous Project, 2004. http://www.jxta.org/tutorials/registryVous.html.

[16] Kazaa. Kazaa, sharman networks, 2004. http://www.kazaa.com.

[17] Maven. Apache Maven Project, 2005. http://maven.apache.org/.

[18] NetCraft. Net Craft, 2005. http://news.netcraft.com/archives/2005/04/01/april_2005_web_server_survey.html.

[19] Pastry. Pastry Project, 2004. http://research.microsoft.com/~antr/Pastry/.

[20] Protocols. Jxta Protocols Specification Project, 2004. http://spec.jxta.org/.

[21] C. Qu and W. Nejdl. Interacting the edutella/jxta peer-to-peer network with web services. In E. C. S. Press, editor, *5th International Conference on Practical Aspects of Knowledge Management*, pages 67–73, 2004.

[22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 161–172. ACM Press, Octuber 2001.

[23] RDF. Resource Description Framework, 2005. http://www.w3.org/RDF/.

[24] SOAP. Simple Object Access Protocol, 2005. http://www.w3.org/TR/SOAP.

[25] SSH. Secure Shell, 2005. http://www.ssh.com/support/downloads/secureshellwks/non-commercial.html.

[26] TWiki™. Enterprise CollaborationPlatform, 2005. http://www.twiki.org/.

[27] UDDI. Universal Description Discovery and Integration, 2005. http://www.uddi.org/.

[28] A. Vermeulen. *The Elements of Java™ Style*. Cambridge University Press, Cambridge, 2000.

[29] W3C. World Wide Web Consortium, 2005. http://www.w3c.org.

[30] WSDL. Web Services Description Language, 2005. http://www.w3.org/TR/wsdl.

[31] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerance wide-area location and routing, April 2001. http://www.cs.berkeley.edu/~ravenben/publications/CSD-01-1141.pdf.