DistriWiki: A Distributed Peer-to-Peer Wiki Network

Joseph Morris
Computer Science Department
Ball State University
Muncie, IN 47306, USA
jcmorris@bsu.edu

Chris Lüer
Computer Science Department
Ball State University
Muncie, IN 47306, USA
clueer@bsu.edu

Abstract

In this paper, we present DistriWiki, a peer-to-peer wiki. Motivated by the fact that the client-server architecture of the Web has limitations caused by the centralized nature of Web servers, we designed DistriWiki as a more open, more distributed alternative. In DistriWiki, each user's computer acts as a peer that stores redundant copies of wiki pages; in this way, we can reduce bandwidth and hardware costs, reduce the number of failures due to hardware and configuration errors, and avoid centralized organizational control of the wiki pages.

Categories and Subject Descriptors D.2.11 [Software Engineering]: Software architectures; H.4.3 [Information Systems Applications]: Communications applications; H.5.3 [Information Interfaces and Presentation]: Group and organization interfaces --- collaborative computing, Web-based interaction.

General Terms Design, Reliability.

Keywords Wikis, peer-to-peer networks, collaborative publishing.

1. Introduction

Wikis have, in recent years, gained extreme popularity. Not only Wikipedia, the flagship of wikis, but also many other, smaller wiki communities have sprung up. Whenever a need exists for a Web site that users cannot only read, but also write – in other words, a need for collaborative publishing – a wiki represents one of the best solutions. However, the architecture of wikis is based on the Web – and the Web is essentially a read-only system. Most Web users cannot write to most pages they read. Thus, there is a mismatch between the essentially distributed nature of wikis and the centralized nature of Web sites.

WikiSym'07 October 21–23, 2007, Montréal, Québec, Canada.

Wikis are different. This raises the question whether the Web architecture, although enshrined in numerous standards and tools, is the most suitable network architecture for a wiki network. Peer-to-peer architectures have always been a part of the Internet; in fact, TCP/IP itself realizes a peer-to-peer architecture. But in recent years, peer-to-peer applications have attracted renewed interest for their ability to avoid the limitations of purely Web-based applications. Most famous among these are peer-to-peer file-sharing applications, but peer-to-peer applications also include business applications such as the Internet telephone network Skype [3].

These developments led us to create DistriWiki, a prototypical peer-to-peer wiki. While DistriWiki provides similar functionality as a traditional wiki, it is designed to avoid the limitations of the Web. We believe that in this way, we can reduce the cost and increase the reliability of wikis, and obtain a better match between the logical structure of wikis and their technical architecture.

In Section 2, we give some background information regarding peer-to-peer concepts and our reasons for building DistriWiki. In Section 3, we list the requirements for a peer-to-peer wiki application. We will then follow in Section 4 with a more technical discussion of our approach. Section 5 evaluates the approach, and Sections 6 and 7 discuss future work and related research.

2. Background

To distinguish between the logical and technical parts of wikis, we call the set of pages (and the links between them) in a wiki the *wiki network*. The concept of a wiki network does not imply any specific technology for its realization. We call the piece of software that allows users to accesses a wiki network a *wiki engine*.

Currently, popular wiki engines, such as MediaWiki [2], TWiki [4], and many others, are based on the client-server architecture of the Web. Each engine is implemented as an extension of a Web server, so that it shares the Web's architecture. Thus, each wiki network has a single server to which clients connect. This means that all wiki pages and published resources are on one central machine.

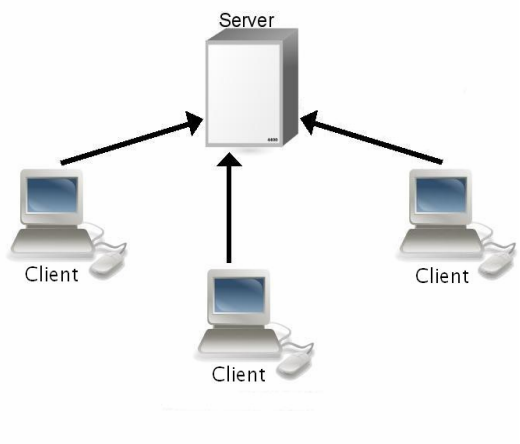

Figure 1. Client-server architecture.

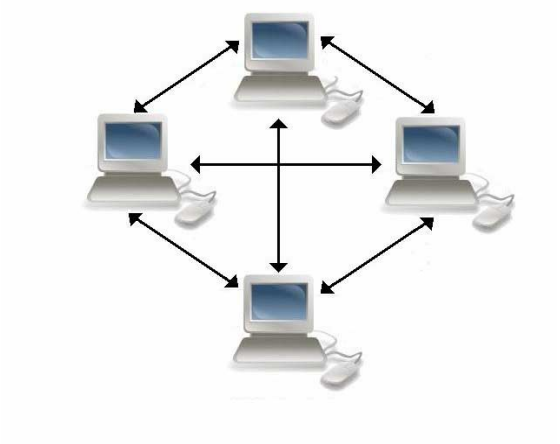

Figure 2. Peer-to-peer architecture.

The client-server architecture has a number of benefits. Most importantly, it allows wiki networks to function as an integral part of the Web, requiring only extensions to server software, but supporting existing clients without change. It also makes wiki engines easy to develop, since they follow well-known principles of Web applications in most regards.

However, the client-server architecture has serious limitations for use in wiki networks:

1. Bandwidth cost and server hardware cost
2. Decreased reliability and access
3. Tight coupling between backend and user interface
4. Centralized administrative control

Hosting a popular wiki on a centralized server is expensive. If the website becomes popular, then there are issues with ensuring that there is sufficient bandwidth to handle requests. The server must also be high-performance since there is the potential for a high load. Wikipedia, the world's largest wiki, spends several hundred thousand dollars a year on bandwidth and hardware. Clearly, a wiki such as this can be operated only by an organization with significant funding, in this case the Wikimedia Foundation.

Like any centralized server, a wiki server represents a bottleneck. This has not only cost implications, but also affects reliability. If the central server fails, for whatever reason, the complete wiki network becomes inaccessible. The larger and more popular a wiki is, the more often failures will occur. Failures may be caused by hardware errors, but will more likely be caused by misconfigurations and administrative mistakes.

In addition, hosting a wiki on a web server implies that the user is restricted to a web interface to edit and publish documents. This tight coupling between the backend and user interface is not flexible enough to accommodate different interfaces based on user preferences.

Finally, client-server architectures suffer from centralized administrative control. This means that there is not only a technical bottleneck, but also an organizational one: the organization that owns the server hardware is the only one that has full control of the wiki network. While other users may be granted access privileges, these can generally be revoked at any time by the owner of the hardware.

In a peer-to-peer architecture [5, 8], each host on a network is a *peer*, meaning that it can act as both a client and a server. Rather than communicating with one server, each peer can freely communicate with any other peer. By removing a central server, we greatly cut costs and solve the problems listed above. Bandwidth is not an issue, since it will be distributed among multiple peers. Access will not be limited because there will be multiple peers with multiple copies. By eliminating the central server and putting the functionality on a peer, the method by which the user chooses to edit the wiki document is entirely up to them. There is no longer a need for a web interface, but instead a variety of wiki clients can be used to edit and publish documents. In addition, users would be able to freely create small wiki networks for their own private use. The system should be more robust in general.

We believe that a peer-to-peer architecture is the right way to push wiki technology ahead. Peer-to-peer networks better match the logical architecture of wiki networks. While in a simple Web site, only one organization has write access (the webmaster or owner of the site), the purpose of a wiki is to provide write access to all users of the site. This is analogous to the architectural difference between client-server and peer-to-peer architectures: just as in a peer-to-peer network, every host acts as both a client and a server, in a wiki every user acts as both a publisher

and a consumer. So what would be more natural than to select the architecture that matches the logical structure?

3. Requirements

The wiki network envisioned requires the following features:

1. Redundant decentralization
2. Unique identification of documents
3. Efficient retrieval of documents
4. Usability
5. Compatibility with wiki concepts

Decentralization means that there is no central server. This concept is the heart of peer-to-peer networking. Instead, data must be distributed among peers. In order to have some reliability in case a peer goes down, we must have redundant data across peers – no single network or hardware failure should cause the system to become unavailable. This is an essential requirement for a peer-to-peer wiki network.

Because documents can be simultaneously modified in this system, we need a method of *uniquely identifying* documents. If two people produce version 3 of the same document, how does the network distinguish between the two? Since there is no central server, there is no way to prevent concurrent versioning. As a result, all documents need a unique, global identifier and the software should notify the user of such conflicts when possible.

Just like any other wiki system, there must be some means of *searching and reliably retrieving* documents. Network access should be easy and reliable, and the failure of a peer should not affect the network. Unlike in a client-server system, the search function must span different hosts, since no single host will have a complete set of pages.

The tools must be usable. Users are accustomed to Web browsers and existing wiki engines. The user interface should be similar to existing technologies, so that users can easily switch. Since ease of use has been one of the main selling points of wikis in the past, high standards of usability should be maintained.

Related to this, the peer-to-peer wiki system should preserve well known *wiki concepts* and semantics. This means that a similar structure of pages and links and in existing wikis should be supported. The semantics of a link will necessarily be somewhat different in a peer-to-peer wiki, since pages may be distributed among multiple hosts. However, the architecture and user interface should maintain the existing concepts as much as possible.

4. Approach

We selected Sun's JXTA library [7] [10] as the peer-to-peer networking framework. It provides the needed functionality and provides easy methods of searching, caching, discovery, and allows the virtual wiki network to be subdivided into other groups. This ensures that people can create their own private networks that are subsets of the general network, and JXTA's authentication schemes allow for making these groups private. JXTA is also scalable, as it does aggressive caching. It can use multiple protocols (such as HTTP and TCP/IP) to work around various firewall limitations. JXTA provides both a Java and a C API. We implemented this project in Java so it could run under multiple operating systems.

4.1 JXTA Basics

JXTA peers are organized into groups. When JXTA starts up, every peer joins a group called Netpeergroup, which serves to bootstrap the network. At this point a peer may choose to join a subgroup. Joining a group allows a peer to request resources from the group. A peer may belong to any number of groups.

All peers are told about network services and resources through the use of Advertisements. Advertisements are XML documents that describe the resource that a peer or group possesses. Usually, these advertisements have a name, a type, unique ID, and other attributes that are relevant to the resource being advertised. JXTA also facilitates searching for advertisements using literal text and wildcards.

Resource discovery is one of the most powerful features of JXTA. It is an easy way to search for needed or desired

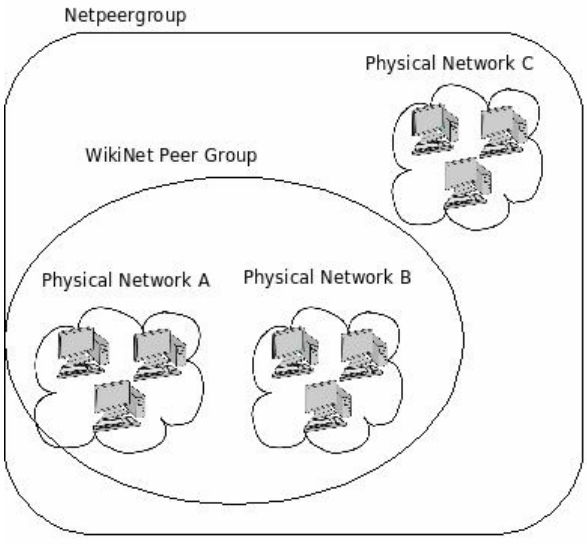

Figure 3. JXTA topology example.

resources on the network. Each peer group has its own DiscoveryService, which a peer can obtain when it joins the group. Searching for resources is typically done using an IP multicast message to discovered peers, but rendezvous peers may use HTTP.

Physically separate networks are joined together by peers known as rendezvous points. Their responsibility is forwarding traffic to another physical network. Pipes are abstractions for the specific use of notification and bulk data transfer. They are somewhat analogous to TCP/UDP sockets, but the underlying protocols they use are unknown to the peer. Unlike a typical TCP connection or UDP transfer, pipes can be created that stretch across multiple physical networks and protocols.

4.2 Network Implementation

JXTA provides many useful features for this project. To make development easier and create a consistent protocol, a series of abstractions were placed around the JXTA features.

For the purposes of this project, we settled on having one small group, called WikiNet, to which all peers could join. The infrastructure for creating subgroups exists in the implementation, but we did not enable this feature in the user interface yet. Once a peer joins this group, it has access to all of its resources. In our code, all of these resources are accessed indirectly through an abstraction we called a WikiGroup.

The main efforts were put into communication and wiki document transfer. JXTA provides a variety of methods to actually implement the transfer and advertising of documents. JXTA advertisements are a good way of publishing documents, but they are inadequate for holding the documents themselves because of JXTA's advertisement caching. If wiki document contents were embedded in an advertisement then every peer on the network would have a copy, and this would be a waste of space. Therefore, pipes are used for the actual document transfer.

When a peer joins WikiNet, the peer initializes a pipe to listen for incoming document requests. This pipe is used for the sole purpose of uploading files to the network. When a JXTA Message object is received on the pipe, the peer will respond to the request by uploading the requested file.

Wiki documents are made known to interested peers via JXTA advertisements. When a new document is created, an advertisement called a WikiAdvertisement is published to the group. This advertisement contains wiki metadata such as publication date, title, revision history, etc. It also includes the pipe ID of the serving peer's listening pipe and the peer's ID. The expiration time set for a published advertisement is the JXTA default, but it would be interesting to see what the optimal setting would be for different sized networks over time.

When a peer requests a document, it does so by first obtaining the desired advertisement from the peer group's DiscoveryService. Once the advertisement is obtained, the peer then extracts the pipe ID field. With the pipe ID, the interested peer then opens a connection to the other peer's pipe and sends a request message. The document is then downloaded. Redundancy and availability are one of the motivations for the peer-to-peer wiki system, but at this point all advertisements have the same pipe ID even though two peers have the document. So, after the download completes, the receiving peer publishes its own advertisement with its own information.

4.3 User Interface

For the user interface, we came across two issues. The first of these was deciding which wiki syntax to support. We decided to support an existing, well-known syntax rather than creating our own. After analyzing a variety of rendering engines, we decided to use the Radeox engine (used by SnipSnap) [6] because of its simple and flexible API.

The second issue we encountered was how to handle links in the wiki documents. On a typical client/server wiki, link handling is natural. The user clicks on a link in the document and the server sends the desired document to the client. It is responsive and quick since all documents are on the same server. Emulating this behavior in a peer-to-peer setup is a little more challenging. There is always the risk of being unable to fetch the requested document, at least within a reasonable amount of time.

Our solution was to handle links by first looking at the local cache, and falling back to searching the network if it does not exist. There are drawbacks to this since the cache may not contain the latest version of a particular document. Though not yet implemented, workarounds will be introduced in the future. These include either the possibility of running a background thread to alert the user of updated information or possibly prefetching links when a document is displayed.

The user interface is currently written in Java Swing, which provides the look and feel of a traditional desktop application. There is support for creating documents using the Radeox wiki syntax, and there is support for viewing the rendered documents.

The method for retrieving documents is currently done by doing a search for a title and/or version of a desired document. We currently rely on JXTA's advertisement search capabilities in order to search for documents. The drawback to this is that there is no way to directly search

the contents of a document; this must be provided in the wiki advertisement. In the future, category and keyword fields will be added to the wiki advertisements in order to help sort through documents.

4.4 Concurrent Changes

Because we are dealing with a decentralized system, synchronization issues may develop. For one, different documents may have the same title or version. There is very little that can be done to avoid this if we want to keep everything decentralized. So, each document is given its own unique ID (randomly generated by JXTA), and every document is given a time stamp. It is then up to the user to handle any conflicts.

5. Evaluation

When evaluating DistriWiki against the requirements set out in Section 3, we determine that these requirements are fulfilled.

The software architecture fulfills our goals of being redundant and decentralized, providing unique identification of documents, being efficient, being usable, and preserving wiki concepts.

The JXTA library provides much of what we need with regards to redundancy and decentralization. JXTA is peer-based, and we make sure that document advertisements are republished when they are downloaded by another machine. This ensures redundancy.

Documents are uniquely identified by using JXTA CodatID's. This ensures that there is a high probability of generating a unique ID for a document. In addition, fields such as author, title, version, etc. can be used to help in the identification of documents.

JXTA's focus on scalability allows us to create and efficient network. The aggressive caching of advertisements allows for quick searches across multiple peers. In addition, bandwidth is distributed, so there is a low probability of flooding a peer with requests.

The separation of the user interface from the networking code allows for good usability. If a user is displeased with a particular interface, a third-party interface could be used. In addition, wiki semantics have been preserved. A user can follow links and search just like a traditional web-based wiki. A user can also edit documents using a familiar wiki mark-up syntax.

Comparing DistriWiki with a traditional client-server wiki network, we can determine the following advantages and disadvantages.

Disadvantages of DistriWiki:

- increased complexity of the software;
- need for handling of concurrent versioning issues;
- need to install additional software on user's computer.

Advantages of DistriWiki:

- technical decentralization – data redundancy causes lower probability of failures and performance issues;
- organizational decentralization – no single organization controls all of the data;
- cost decentralization – bandwidth cost is shared by peers, thus removing the need for a central organization;
- decoupling of user interface from backend – users have the ability to develop custom frontends.

6. Future Work

DistriWiki currently exists as a research prototype. We plan to perform measurements of the prototype, and to extend its functionality.

6.1 Measurements

Efforts are being made to setup a real-world testing environment. We intend to take sample documents from Wikipedia and upload them to WikiNet, and then have test subjects join the network and begin working.

Specific data we want to collect includes scalability, reliability, performance, and overall user satisfaction. At this point, we do not know how well our abstractions around JXTA would scale. There may be alternative approaches using JXTA that may be better. Data may show that a peer-to-peer wiki would work best in private networks rather than the global network we envisioned.

Reliability is something else we want to take into account. With different sized networks, is the availability of the documents we want acceptable? What is a good minimum/maximum number of peers for optimal reliability? Another category we want to measure is performance: we want to find out how long it takes to obtain a document when compared to a traditional client/server peer-to-peer system.; we want to measure response times when following links, downloading, propagating advertisements across the network, etc. On the usability end, we want to focus on overall user satisfaction and feedback. We especially want to know how easy our software is to use compared to traditional wiki systems.

6.2 Planned Extensions

We plan on extending the prototype to allow for subgroups, improved link handling, user interface improvements, wiki advertisement enhancements, and possibly security measures for private networks.

For a large-scale wiki network, the ability to have subgroups could decrease search times and allow for a more efficient network. If a user is only interested in a particular subset of documents, they can join the appropriate subgroup. This can save on both network traffic and disk space on the peer.

We stated earlier that good link handling is necessary for good usability. We intend to introduce and test alternatives to what we currently have implemented. Of particular interest is the concept of link prefetching, where a background thread will run to fetch all the links on a particular page when a user views a document.

It would be nice to offer the user a choice of interface. One of our intents is to implement a proxy to provide a web-based frontend to the network. In this way, the user can access WikiNet without having to give up the familiar web-based wiki interface.

Wiki advertisements currently contain limited information. It is desirable to include more descriptive information such as wiki contents and keywords to assist in searching. Finally, support for JXTA's authentication protocols would allow for private, secured groups; private, limited-access networks should exist.

7. Related Work

The only other peer-to-peer wiki network we are aware of is Code Co-op [1], a commercial product from Reliable Software. Code Co-op is a distributed peer-to-peer revision control system aimed at software developers. It supports synchronization and collaboration via email, a LAN, or a VPN. It also supports a bug database and the option to create a wiki for a given software project. It has its own easy-to-use wiki editor and publishing capabilities. Synchronization is as easy as selecting an option from a menu.

Compared to Code Co-op, our approach is generic and not targeted towards software developers. The actual revision control is done as a means to implement a wiki rather than being a primary goal.

Oster et al. [9] discuss the importance and potential of peer-to-peer collaborative editing, including peer-to-peer wikis. They do not, however, develop such a system.

The idea of separating the user interface (frontend) of a wiki from its data storage layer (backend) was presented by Völkel [11], who uses Web services to accomplish this goal.

A wiki is a hypertext system, and peer-to-peer hypertext systems have been proposed. For an overview, see the discussion and references in [12]. However, these approaches generally discuss hypertext systems that are more heavy-weight than wikis.

8. Conclusions

This paper is an overview of DistriWiki, a peer-to-peer wiki network. We gave requirements for a peer-to-peer wiki network, described our approach, and evaluated it.

While DistriWiki is still in prototype stage, we believe that it gives us valuable insights into future wiki technologies. Its peer-to-peer approach gives us the opportunity to develop wikis that are truly open and distributed. In current wikis, there is always a central server and a central organization that owns the server. In a peer-to-peer network, we obtain the exciting opportunity to have a fully open network, where the technical and organization structure matches the logical structure of wikis.

In a wiki, everyone is an author. In DistriWiki, everyone can contribute to the technical infrastructure by providing a peer. Thus, the logical growth of a wiki scales together with its physical growth. An additional benefit is the reduction of cost; while large wiki installations such as Wikipedia occur expensive hosting costs, a peer-to-peer network allows for the sharing of cost among its users.

References

[1] *Code Co-op*. Accessed last: 2007. http://www.relisoft.com/co_op/.

[2] *MediaWiki.* Accessed last: 2007. http://www.mediawiki.org/wiki/MediaWiki.

[3] *Skype.* Accessed last: 2007. http://skype.org/.

[4] *TWiki.* Accessed last: 2007. http://twiki.org/.

[5] Androutsellis-Theotokis, S. and Spinellis, D. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys,* 36, 4, (2004) 335-371.

[6] Jugel, M.L. and Schmidt, S.J. The Radeox Wiki Render Engine. In *WikiSym '06*, ACM, New York, 2006, 33-36.

[7] Oaks, S., Traversat, B. and Gong, L. *JXTA in a Nutshell.* O'Reilly, Sebastopol, 2002.

[8] Oram, A. (ed.) *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, Beijing, 2001.

[9] Oster, G., Urso, P., Molli, P. and Imine, A. Data Consistency for P2P Collaborative Editing. In *Computer-Supported Cooperated Work 2006,* ACM, New York, 2006, 259-267.

[10] Traversat, B., Abdelaziz, M., Duigou, M., Hugly, J.-C., Pouyoul, E. and Yeager, B. *Project JXTA Virtual Network,* Sun Microsystems, Inc., 2002.

[11] Völkel, M. SemWiki: A RESTful Distributed Wiki Architecture. In *WikiSym '06,* ACM, New York, 2006, 141-142.

[12] Wiil, U.K. Peer-to-Peer Hypertext. In *Hypertext 2002,* ACM, New York, 2002, 69-71.