Design of a P2P Infrastructure to Support Plagiarism Detection Mechanisms

Juan Andrés Mussini
Programa de Pós-Graduação em Informática Aplicada
Pontifícia Universidade Católica do Paraná
Curitiba–PR, Brasil, 80215-901
juan.mussini@pucpr.br

Lau Cheuk Lung
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Florianópolis–SC, Brasil, 88040-900
lau.lung@inf.ufsc.br

Fábio Favarim
Departamento de Automação e Sistemas
Universidade Federal de Santa Catarina
Florianópolis–SC, Brasil, 88040-900
fabio@das.ufsc.br

ABSTRACT

Nowadays the Internet has become a reference on information retrieval. But this can be misused, as one can simply access information and take it as his own authorship. This constitutes an act of plagiarism. It has become increasingly common for people to do this, and tools to prevent this are in need. In order to help to deal with this relevant problem, this paper presents the PeerDetect, a new P2P middleware to support a plagiarism detection system. The proposed solution is based on a P2P network, where a plagiarism detection mechanism uses PeerDetect to distribute the effort of doing this detection among peers. The proposed approach allows us to reach a better performance and scalability.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Algorithms, Design

Keywords

P2P, Middleware, DHT, Plagiarism Detection, Internet

1. INTRODUCTION

The advent and the expansion of the Internet brought unquestionable benefits to socialization of information. The Internet has been essential in the life of many people as it is very easy to access the unmeasurable amount of information. This easiness contributes to an efficient distribution of the human knowledge and its acquirement [1]. However, this socialization of information also brought an important problem: plagiarism.

Plagiarism occurrences have been growing in the last years [23], and tools to make copies grew also in the same proportion, from a simple search in a content indexer, such as Google [10], to sites that offer already made work [19, 11]. Many professors, specially the ones that are not from computer-related areas, do not have information about fraud detection techniques and tools. Usually, to detect plagiarism it is used a manual checking, where a professor selects some random phrases from the document and use them to search manually for matching documents using a search engine. However, manual checking is very time-consuming and laborious process becoming many times impractical.

Nowadays, some software tools have appeared [4, 12] in order to automate this process. These tools, generally indicates what is the similarity degree between two documents. If this degree is considered high, a manual checking by a specialist (professor, for example) is needed. This manual check is essential, because there is no detection system that guarantees 100% assurance.

Therefore, we propose in this paper a P2P middleware, called PeerDetect, which is well suited to support a natural language written documents plagiarism detection system. PeerDetect uses the Internet as a data source and the P2P network as a way to ensure load balance among the peers the work of searching plagiarism on Internet. When the peers join the P2P network, they mark themselves as peer available for job. However, when a peer receives a document to be analyzed, it becomes a *main peer* which analyzes the document size and the number of helper peers. Based on this, the *main peer* defines in how many parts the document will be divided, how many peers on the network will be used and which will be the peer/part relation. Each part is sent to the helper peer assigned to that part. These peers do a simple search using a Internet searching mechanism and then publishes the pages found. The *main peer* collects these answers and presents them to the user for final evaluation.

The remainder of this paper is organized as follows. Sections 2 and 3 present concepts and related technology regarding the plagiarism detection system – Section 2 presents peer-to-peer networks and DHTs (Distributed Hash Tables) and section 3 presents current plagiarism detection techniques. Section 4 presents related works. Section 5 presents the

PeerDetect Middleware. Section 6 presents performance results. Finally, Section 7 gives some concluding remarks.

2. P2P ARCHITECTURES

Peer-to-peer (P2P) systems offer resources (services, files, etc) in a distributed manner, where each peer can search and provide resources. Differently from conventional systems, such as client-server model, where services (resources) are usually kept on the server, P2P systems provide collaboration and access between clients and resources in a disperse manner, without the need of a central server in a distributed system. One of P2P properties is to provide to its participants, called peers, a symmetric role, with the same capabilities and responsibilities.

2.1 JXTA

In the beginning, when some P2P protocols and applications began to be used, they did not interact with each other. Aiming to deal with this problem, Sun Microsystems launched the JXTA platform [22]. JXTA was conceived in order to standardize a common set of open protocols allowing any device on the network to communicate and collaborate in a P2P manner. The JXTA protocols are defined as a set of XML messages.
JXTA protocols establish a virtual network overlay over the Internet, allowing peers to directly interact and self-organize independently of their underlying network connectivity and topology. Thus, many Ad-hoc virtual networks can be created and dynamically mapped into one single physical network. JXTA defines a minimum set of requirements for peers to form and join a virtual P2P network. So it is possible that application developers define the network topology that is best suited for their application requirements. JXTA was designed in order to be independent of programming languages, operational systems, services definitions and network protocols.

2.2 Location in Peer-to-Peer Systems

One of P2P networks most important services is the resource discovering. This service allows the users to find in the P2P network the resource (services, files, etc.) they need. In general, the result consists of list of peers where the resource can be found. After the resource is found, the requesting peer can make direct access to the owner of the resource. Basically, there are three approaches to the content location in P2P systems: centralized, flooding and DHT based.
In the first approach, a central server maintains an index of all resources being shared. This index can be accessed by all client nodes. When a node joins the system, it contacts the central server and sends a list of the resources that are available for other nodes. To locate a resource, a client node sends a query to the central server, which performs a lookup in its database and replies with a list of peers that have the desired resource. After querying the central server, it starts communicating directly with the others nodes in order to obtain the desired resource.
In the flooding approach, all nodes are autonomous and have the same role. In this approach, there is not a central server and each node is responsible for maintaining the index of the resources it stores. Because the participating nodes are organized in a non-structured way, each node can communicate directly with some nodes and indirectly to all nodes in the network. Thus, a node cannot be aware of which other nodes are in the network. In this way, in order to perform the querying, a limited-scope query message is flooded to the network. It works as follow: each request is firstly sent to directly connected nodes, which sends this request to nodes directly connected to them, and so on, until the limited scope is reached (usually between 5 to 9 steps). Notice that, since query messages have limited scope, it is possible that nodes cannot locate resources even though the files are available in the network.

2.2.1 Distributed Hash Tables

Distributed Hash Tables (DHTs) are a class of decentralized distributed systems that perform the functions of a hash table. A hash table stores (key, value) pairs and values are look up through the key. In DHT, the key-space is distributed among the multiple nodes. Thus, both storage and lookups are distributed among participating nodes. DHT are designed to handle with continuously joins, leaves and failures of nodes. This allows DHTs to scale to large numbers of nodes. DHTs can then be used to the implementation of a diverse variety of peer-to-peer applications, such as file sharing, distributed file systems, domain name services, cooperative web caches, and instant messaging.
Unlike traditional P2P lookup mechanisms (centralized and flooding), the DHT uses a more structured key based lookup mechanism in order to achieve both the decentralization of Gnutella and efficiency of results of Napster. However, the DHT has one drawback, it only supports exact match search (using the key), rather than keyword search. This functionality can be layered on top of a DHT [2].
The first DHTs based infrastructures (CAN [14], Chord [20], Pastry [17] and Tapestry [24]) were introduced about the same time in 2001. Since then this area of research has been quite active.

3. PLAGIARISM DETECTION TECHNIQUES

The Plagiarism detection can be classified in two different types: Source Code Plagiarism Detection and Plagiarism Detection in Natural Language Documents. In source code detection there are two methods often used: attribute counting and structures measuring [5]. The source code detection is more common in the literature than the natural language approaches. It is expound by the well defined syntax of the programming languages found in source codes. The natural language is more complex, where a single expression has different meanings. In this project, the source code plagiarism detection is out of scope of this paper.

3.1 Plagiarism Detection in Natural Language Documents

Usually, the plagiarism in documents written in natural language is manually detected by professors and tutors. If they are familiarized to the style of their students, they will be able to identify irregularities in documents written by them. This identification can be achieved by comparing the documents with old ones in order to find expressions and vocabularies different from those the students are accustomed to. Other distinguishing characteristics that can be used to help identifying plagiarism in natural language documents are, for instance [6]:

- Vocabulary: comparing the current vocabulary with the previous used vocabulary. The highest is the difference, which means the higher number of new words, the lowest is the probability of copy. Another point is the dynamism of the vocabulary. If the current vocabulary often changes in the same document, it probably means that a copy was done;

- Text incoherence: if the text does not present consistence, it can be an indication of copying;
- Punctuation: the punctuation varies from one text to another. Similar punctuation can be an evidence of a copy.
- Text Similarities: texts using the similar terms, names and definitions, including the order of the similarities. Continuous grammar mistakes;
- Long sequences of well know texts or documents;
- Dependence on specific words and sentences;
- Readability: using measurement methods, the Gunning FOG[16] for instance, the text can be scored. Different documents must have different scores;
- Missing references: when a reference is mentioned in the text but it is not in the bibliography.

There are simpler algorithms that try to detect plagiarism with fewer but important characteristics. These characteristics can be extracted and compared in another moment – both documents do not need to be compared at the same time. An example of this is seen on [8], where the proposed algorithm consists on finding words that only appear once in the whole text are detected. These words are called *hapax legomena*. Documents with a similar *hapax legomena* could be a copy.
Other algorithms explore the document in a deeper level. They analyze further information, such as document structure, keywords extraction (substantives, verbs, adjectives and adverbs) and structure characteristics (document structure related to keywords), and they need both documents present to get compared. Documents that have similar characteristics could be a copy. COPS [3], Glatt[9] and SCAM [18] have these characteristics.
One could reduce the scope of a deeper algorithm aand use it as a simpler algorithm. For example, instead of exploring all document characteristics, one could only extract a single characteristic and compare it.

4. RELATED WORK

There are some plagiarism detection systems, but there is only one dealing with P2P networks, called DetectIt [21]. For the authors of DetectIt, Elif Tosun and Ben Wellington, the ideal solution for a plagiarism system is to keep a record of all written documents so that a new document is submitted, it is compared with all the other documents in the database. Therefore, the more users there are, a larger base there will be. This base, due to its size, would have to be distributed. DetectIt was developed following these characteristics, written in Java, and using Tapestry as a base. It uses the concept of *fingerprints* for documents comparison. If two documents have similar fingerprints, they could be potentially a copy. A fingerprint is a sequence of characters of fixed size n. DetectIt searches both documents, extracts and publishes a specific amount of fingerprints, and compares them. The tests made with DetectIt show that the system is fit for the user needs, but the results only show fingerprint generation times, and not the plagiarism detection. This makes it hard to compare the efficiency between DetectIt and PeerDetect. Unfortunately DetectIt was not further developed, there is only one paper about this system, and it was published in 2003.
Most of the traditional plagiarism systems that deals with documents written in a natural language are paid, such as: Copycatch [7], Turnitin [12] and Eve [4]. These tools are based on traditional client-server model. The user sends a document to the service, which is connected to a huge database that makes the plagiarism detection process. PeerDetect uses P2P technology for reaching scalability, fault tolerance and better performance.

5. PEERDETECT MIDDLEWARE

In this section, we present a natural language written documents plagiarism detection system named **PeerDetect**. PeerDetect is based on the ideas of Web2Peer [15]. Web2Peer is an infrastructure for publication and retrieval of web pages in a P2P network. PeerDetect has two approaches. They differ mainly on the data source used for plagiarism search and also on the algorithm used to perform this search. In this paper, due the lack of space, we only present the decentralized approach.
PeerDetects' decentralized approach uses both WWW (World Wide Web) and P2P networks advantages. Regarding the WWW, it is the data source used to compare the document submitted for verification. The Web has countless documents, and indexing mechanisms such as Google are capable of sweeping a large amount of this information in a few seconds.
PeerDetect's uses Google to search for documents with similar phrases. However, a large document would take too long to be verified if searched phrase by phrase on Google. This could mean a large amount of work for a single computer. This is why we propose to use P2P networks. Document parts to be searched are distributed among the participating peers so the work load can be balanced.
Every network peer communicates through DHT. It is used as a "message board", where every peer publishes its state and tasks. Every peer becomes leader when it has a document to be verified, and the others will be helper peers. However, a helper peer can also be a leader peer at the same time. The decentralized approach consists on two different process, tasks publishing and tasks checking.

5.1 Tasks publishing

This process is executed by a peer whose owner has submitted a document to be searched. This peer first calculates the submitted document size and searches in the DHT to know how many peers are available to work with. With this information, the owner peer (or *main peer*) distributes the tasks for each peer. The tasks consist simply on text parts. The peer then waits for the analysis results. Peer 0 on Figure 1 presents the task publishing procedure and each step is detailed on the sequence.
The owner peer uses a local table to handle the information of which document parts has been analyzed. When it distributed the tasks among the available peers, it updates

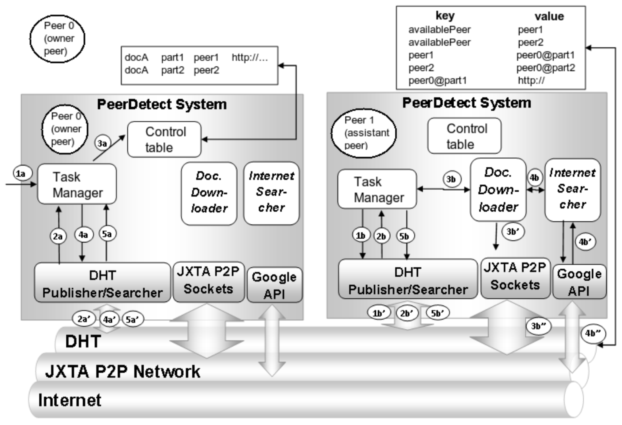

Figure 1: Task publishing (Peer 0) and checking (Peer 1) on the decentralized approach

Table 1: Storage structure used on the DHT for the decentralized approach

	key	value
1	$availablePeer$	$peerId$
2	$availablePeerId$	$ownerPeerId@docPart$
3	$ownerPeerId@docPart$	$result$

this table in order to keep track about what peer is working on which document part. After the peers return the results, the owner peer updates the table with this information.

1. a) User submits a document to be analyzed to the owner peer (peer 0). This document is handled by the "Task manager" module;

2. a) Owner peer searches in the DHT for available peers. This search is made by searching for the key *available-Peer* on the DHT (line 1 of Table 1). The DHT returns a list of peer ids;

3. a) Owner peer distributes document parts for each available peer and updates the control table with this information. This distribution is made by simple dividing the number of pages by the number of available peers;

4. a) Owner peer publishes the task information on the DHT using the structure presented on line 2 of Table 1 and the "DHT Publisher" interface;

5. a) Owner peer periodically searches for answers using the "DHT Searcher" interface. It does that by searching for the concatenation of its own identifier (*ownerId*) and the published document part identifier (*docPart*) (line 3 of Table 1). In case of answers, updates the control table.

PeerDetect uses TTL (time to live) when it publishes an information. The TTL is the amount of time an information will be available in the DHT before it expires and be deleted. The pending tasks information published by the owner peer has a medium TTL, because tasks assigned to peers that are not available anymore should not be left unsolved.

5.2 Tasks checking

The assistant peers (or *helper peers*), whose owners did not post any document, take another role. The first action is to publish in the DHT its state: $availablePeer$ (structure presented on line 1 of Table 1). This information have a low TTL, because it must be accurate, so that the responsible peer when receiving the available peer list does not give a task to peers that are now busy. After that, the assistant peers periodically search in the DHT, checking for tasks that a owner peer could have posted. If the peer finds its own Id on the DHT, it downloads its task using the JXTA server, and executes the task, searching Google for phrases. If this search has some result, the peer publishes this information using the structure presented on line 3 of Table 1. This information has a high TTL.
Peer 1 on Figure 1 presents step-by-step the task publishing procedure and bellow each step is explained.

1. b) When entering the P2P network, the peer (peer 1) does an self-publishing on the DHT using the "DHT Publisher" interface, informing that it is available. It does this by publishing the keyword $availablePeer$ along with its identifier ($availablePeerId$) (line 1 of Table 1);

2. b) The assistant peer periodically searches for its $peerId$ on the DHT. It looks for tasks that a owner peer (peer 0) may have had assigned for it;

3. b) When there is a task, it downloads the parts through the JXTA server;

4. b) The assistant peer then searches through the Web for phrases using Google's API. The user can define the phrase length. The search is made using quotes, meaning that Google will search for that exact phrase;

5. b) Results are published using the structure presented on line 3 of Table 1.

6. EVALUATION

A good plagiarism detection tool must be, above all, helpful. This means that it has to at least point out for the right direction, so that the specialist has the least amount of false positives (detected plagiarism that really is not). It also has to present these results in an acceptable time. In other words, the tool has to be working in a correct (hits) and efficient (time) manner. The tests were made using 3 computers, an AMD Turion 1.6 GHz with 1 GB ram as peer 0 and two AMD Athlon XP 2.6 GHz with 512 MB RAM as peers 1 and 2 in a 100Mb bandwidth network. The software environment used was the Linux OS 2.6.23 32 bits and JVM version 1.6.0.03.
On the decentralized approach tests are made to measure the total time and hit (correct answer) percentage of the results. The phrase length definition by the user is an important step, as the smaller the phrase length, the more false positives are found. Tests showed that a phrase length of 15 words is enough for good results.
Table 2 shows the number of queries executed with a single computer, using PeerDetect with 2 peers and with 3 peers. Three documents were used for tests. The amount of queries consists on every phrase on the document with 15 words. As the number of pages is divided by the number of peers, the

Table 2: Queries for each test

Document size (pages)	Single computer	2 peers	3 peers
13	42	28	22
22	46	23	13
93	390	174	82

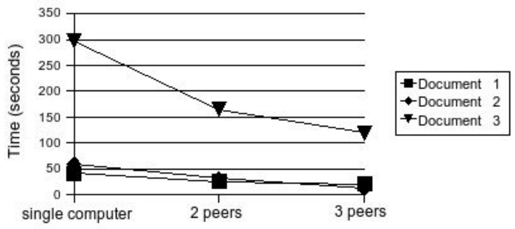

Figure 2: Decentralized approach total execution time

number of queries could increase or decrease depending on how many 15 words phrases are there on the page. The values on Table 2 are an average number of queries for each peer.
Figure 2 presents how much time it took to complete the whole search in each case. The larger the document, the more queries there are, and the longer the time. But having an assistant peer helps bring the total time to nearly half. The time is related to the number of queries each peer has to search, meaning that if the document is small, there is no great advantage of having many peers dividing the tasks. In the scenario where there are two computers instead of one, the execution time is not reduced to half, since there is the task publication time and its answers (overhead).

6.1 Determining query format and size

Let us suppose that the plagiarism author copied the following phrase "The pinball machine is an arcade standard that most enjoy having a game or two on. There are also those who have truly mastered its secrets. Find out about the vagaries and nuances of tilt sensors, replays, matches, combination shots and more". Searching Google for this phrase, the results are presented on Table 3. Among the presented results, in some cases they do not have any relation with the query, they only have the query words spread among the text. These results are not considered to be a plagiarism. The results in which the phrase is identical as the query characterizes a plagiarism. Table 4 presents the hit percent among the presented results. It can be seen that the use of quotes guarantees a 100% hit rate, using a reasonable phrase size, for example 13 words. The test consisted on using 10 different phrases.
The first one consisted on the selection of five words that were the most frequent word on the text and also the five words that were the least frequent on the text. The phrases which these words were part of the selected and then searched on Google. To illustrate this search, on a given document the most frequent words were: period, would, were, United, and states. Some of the phrases in which the first of this words (period) takes part are:

- "(...)brackets by time period. The following discussion provides(...)"
- "(...)history of the time period in which it occurred and(...)"
- "(...)with bracket I covering the period from 1789-1850,(...)"

These phrases were all searched on Google using quotes. Queries which resulted in more than 1.000 results were ignored. The results are presented on Table 5. There is basically no difference using between phrases with least frequent words and most frequent words. All of these results in an average of 1 result per search. Thus, the use of quotes is determining. Documents 1 and 5, since they are essays about common topics (one is a Windows XP help guide and the other is about the film Titanic), had a large number of results.
The tests using no quotes consisted on selecting the 30 words most frequent and the 30 words least frequent on the text. These words were then searched on Google without quotes. The results are presented on Table 6.

Table 3: Result total

Phrase format	Full 42 words	26 words	13 words	9 words	5 words
Without quotes	127	18500	47000	210000	1720000
With quotes	1	1	1	4	6

Table 4: Results relevance

Phrase format	Full 42 words	26 words	13 words	9 words	5 words
Without quotes	0.7	0.005	0.002	0.001	0.0004
With quotes	100	100	100	25	17

Table 5: Queries results using quotes

	Most frequent	Least frequent
Document1	178 searches with 9644 results total - average 54 results per search: 6	3 searches with 518 results total - average 172 results per search
Document2	44 searches with 50 results total - average 1 result per search	17 searches with 21 results total - average 1 result per search
Document3	23 searches with 71 results total - average 3 results per search	1 search with 2 results total - average 2 results per search
Document4	95 searches with 116 results total - average 1 result per search	4 searches with 4 results total - average 1 result per search
Document5	6 searches with 1655 results total - average 275 results per search	0 searches with 0 results total
Document6	119 searches with 128 results total - average 1 result per search	5 searches with 6 results total - average 1 result per search

Table 6: Queries results using no quotes

	Most frequent	Least frequent
Document1	25500	7
Document2	23300	0
Document3	281	0
Document4	146000	0
Document5	3	0
Document6	527	1

The results show that using the most frequent words there is a larger number of results, making it hard for a manual inspection. This can be explained by the simple fact that common words are more likely to be found on a great amount of documents. If there is no restriction filter (like the quotes), the results will be large. Using the least frequent words there are few or none results, perhaps due to Google indexing mechanism, but the relevance of the results if any, are high. Thus, we believe that using no quotes is not a confident way of finding plagiarism.

7. CONCLUSIONS

The main objective of this work is to offer an efficient plagiarism detection middleware. The decentralized approach uses the WWW as data source, but also uses the P2P network to load balance the work of Internet searching among the peers. This method uses a set of existing technologies and mechanisms, like DHT and JXTA. The decentralized approach has the advantage of giving a quick answer for evident plagiarism. The evaluation section presented the performance of the proposed middleware, showing the applicability of the system.

The authors are planning to use PeerDetect as an official plagiarism detection tool in the University, where the anti-plagiarism culture is not well known and many professors have no idea how to do that. The authors hope that PeerDetect will help the professors is such a task. The PUCPR University uses a collaborative teaching system called Eureka [13] where professors and students have virtual classes. This system provides many functionalities, like the online homework delivering, allowing students to upload their work through a web browser and professor to take them also from a web browser, for evaluation process. This would be a good data source to PeerDetect. Thus, if PeerDetect is used with a similar system in other universities, each university would become a peer in the P2P network providing its documents.

8. REFERENCES

[1] P. G. Armour. The five orders of ignorance. *Communications of the ACM*, 43(10):17–20, 2000.

[2] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking up data in p2p systems. *Communications of ACM*, 46(2):43–48, 2003.

[3] S. Brin, J. Davis, and H. García-Molina. Copy detection mechanisms for digital documents. *SIGMOD Rec.*, 24(2):398–409, 1995.

[4] CaNexus. EVE Plagiarism Detection System. Last Checked: 2008-06-16, available from: `http://www.canexus.com`.

[5] X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker. Shared information and program plagiarism detection. *IEEE Transactions on Information Theory*, 50(7):1545–1551, 2004.

[6] P. Clough. Plagiarism in natural and programming languages: An overview of current tools and technologies. Technical Report CS-00-05, Dept. Computer Science, University of Sheffield, UK, 2000.

[7] C. S. Development. Copycatch Gold. Last Checked: 2008-06-16, available from: `http://www.copycatchgold.com`.

[8] S. Finlay. Copycatch. Master's thesis, University of Birmingham, 1999.

[9] Glatt. Glatt Plagiarism Services. Last Checked: 2008-06-16, available from: `http://www.plagiarism.com`.

[10] Google. Google Search Engine. Last Checked: 2008-06-16, available from: `http://www.google.com`.

[11] HighBeam Research. Homework Center. Last Checked: 2008-06-16, available from: `http://www.infoplease.com/homework`.

[12] iParadigms. Turnitin. Last Checked: 2008-06-16, available from: `http://turnitin.com/`.

[13] Puc-Pr. Eureka – Ambiente Virtual de Aprendizagem. Last Checked: 2008-06-16, available from: `http://eureka.pucpr.br`.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc.of the Symposium on Communication, Architecture, and Protocols for Computer Communications*, pages 161–172, San Diego, CA, USA, August 2001.

[15] H. B. Ribeiro, L. C. Lung, A. O. Santin, and N. L. Brisola. Web2peer: A peer-to-peer infrastructure for publishing/locating/replicating web pages on internet. In *ISADS '07: Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems*, pages 421–428, Washington, DC, USA, 2007.

[16] Robert Gunning. Plain Language At Work Newsletter. Last Checked: 2008-06-16, available from: `http://www.impact-information.com/impactinfo/newsletter/plwork08.htm`.

[17] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, pages 329–350, Heidelberg, Germany, November 2001.

[18] N. Shivakumar and H. García-Molina. SCAM: A copy detection mechanism for digital documents. In *Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries*, 1995.

[19] StartSpot Mediaworks. Homework Spot. Last Checked: 2008-06-16, available from: `http://www.homeworkspot.com`.

[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc.of the Symposium on Communication, Architecture, and Protocols for Computer Communications*, pages 149–160, San Diego, CA, Aug 2001. ACM Press.

[21] E. Tosun and B. Wellington. Detectit: a peer-to-peer plagiarism detection system. *Corant Institute, NYU*, 2003. (Unpublished manuscript).

[22] B. Traversat, M. Abdelaziz, M. Duigou, J. Hugly, E. Pouyoul, and B. Yeager. Project JXTA virtual network, 2002.

[23] K. Zernike. With student cheating on the rise, more colleges are turning to honor codes. *New York Times*, Novembro:10–11, 2002.

[24] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, Apr. 2001.