Using Structured P2P Overlay Networks to Build Content Sensitive Communities

Paul Stacey[1], Damon Berry[1] and Eugene Coyle[1]

[1]School of Control Systems and Electrical Eng.
Dublin Institute of Technology
Kevin Street
Dublin 8
IRELAND

E-mail: {paul.stacey, damon.berry, eugene.coyle}@dit.ie

Abstract—**This paper details a proposed peer-to-peer system, which allows a user to join communities of other like-minded users in order to exchange files. Utilising the routing capabilities of Pastry, the proposed system will include an indexing service, which will facilitate the creation of virtual rendezvous points for users with similar interests (manifested by shared keywords). Users will be described by the content they store. By using Vector Space Modelling techniques users can be grouped together to form content sensitive communities. The system is intended to be used as the basis of a distributed archive of research papers**

The system is designed to improve the efficiency of file searches over current p2p file sharing applications. Search results will be more comprehensive given a set of search keywords as searching will be based on semantic meaning rather than literal matching.

Keywords: Peer-to-Peer; Structured p2p Overlay Networks; Clustering; File Sharing; Vector Space Modelling; Content Networks.

1. Introduction

Community: a body of people having common rights, privileges, or interests [1]. Applying this notion of community to the Internet, it seems that the Internet as a whole lacks a sense of community. The World Wide Web as originally envisaged in 1986 by Tim Berners-Lee as a way for academics to share knowledge. The web has become a victim of its own success in relation to that goal. While there is a huge quantity of information on the web and it is easy to find text about almost any topic, it is often difficult to distinguish ”high quality” information such as peer-reviewed papers from material from less prestigious sources. The client-server paradigm may be partly to blame for this; power has been taken away from the individual and been placed in the hands of operators of large servers. In recent times, systems such as Napster [2] and Gnutella [3] have gained huge popularity. These systems have initiated a surge of interest and research into the Peer-to-peer (p2p) framework. These systems are restructuring the Internet away from the client server model to one where a client is also a server, giving individuals more freedom and control. This paper presents the core of a distributed document-sharing environment with particular focus on distributed searching and retrieval of research papers.

Problems with Current p2p Systems

Despite their obvious popularity, systems like Napster and Gnutella suffer from many problems. Napster uses centralised indexing servers, an approach which is vulnerable to failure. Gnutella avoids Napster's weakness by using a decentralised indexing technique, however this leaves Gnutella with the problem of locating objects within its network. Gnutella uses a flood-based search technique where each search request blindly hops across the network from one node to another searching for the requested file. The search request will only reach an ever-decreasing section of the network as more users join; as a result this technique doesn't scale very well. In recent times, more scalable object location algorithms have emerged that are based on Distributed Hashtables (DHT). Let us consider these useful constructs and how they can be used to facilitate searches in a distributed environment.

Existing Structured p2p Overlays

Based originally on the research of Plaxton et al in the late 1990s [4], DHT overlay network implementations first appeared during 2001. Projects such as Pastry [5], Tapestry [6], CAN [7] and Chord [8], all produced implementations that adhered to the principles of P2P, decentralisation, robustness and scalability. These systems may be used to form the foundation for functional p2p systems. They provide a routing substrate; a mechanism that efficiently locates objects within a certain number of routing hops. The subject of keyword searching is of particular importance if such DHT systems are to become part of the more mainstream p2p systems such as Gnutella or Kazza [9]. Introducing keyword-searching capabilities into these systems is likely to render them a more powerful tool for file sharing than is currently available. DHT's currently provide only put and get functions. These substrates have however proved useful in building such systems as global storage facilities, including PAST [10], CFS [11] and Oceanstore [12]. PAST which as we shall see has a lot in common with the system proposed here, is built on top of Pastry and uses the power of Pastry to route files entered into the system to a particular point, given a file key. The file may be retrieved once the file key is known.

Before venturing further into the details of p2p technologies, it is helpful to consider a useful application of DHT systems. The application seminal to this research is that of *Content Networks*.

Content Networks

A content network is an overlay IP network that supports *content routing*, Content routing means that messages are routed based on their content rather than their IP-address. In recent years many types of content networks have been developed, including p2p networks. PAST is one such development. Content networks can be classified into many different types. In the following discussion users will be associated with identifiers that have semantic meaning. Users will also be subject to content-sensitive placement. [13] Gives taxonomy to the different types of content networks.

It is intended to build a p2p system on top of Pastry that supports the construction of communities based on the content they store. These communities will make searching for files far easier and more efficient because searches may be directed to particular communities within the network where the files are more likely to be stored. These communities will be formed using state-of-the-art Information Retrieval (IR) techniques in-order to better discover relationships between files and thus return more comprehensive search results.

The above discussion has provided a brief description of the basic technologies that are currently in use in the area of content networks and p2p. The ingredients of the proposed system will now be discussed.

2. Ingredients of the Design

It is hoped to build a p2p framework whereby users will be able to join the network and have contact with other users who are either interested in a topic or have the means to share content on that topic. It is important to describe the chosen routing substrate. Pastry will provide the p2p framework with a scalable and robust routing algorithm (described in section 2.1.). The system will also incorporate a state-of-the art Information Retrieval (IR) technique used for representing documents, known as

Vector Space Modelling (described in section 2.2.). In order to group together similar users within the system, clustering techniques are employed. This will enable the system to form communities of users based on a similarity index derived from Vector Space Modelling. This is the focus of section 2.3.

2.1 Pastry

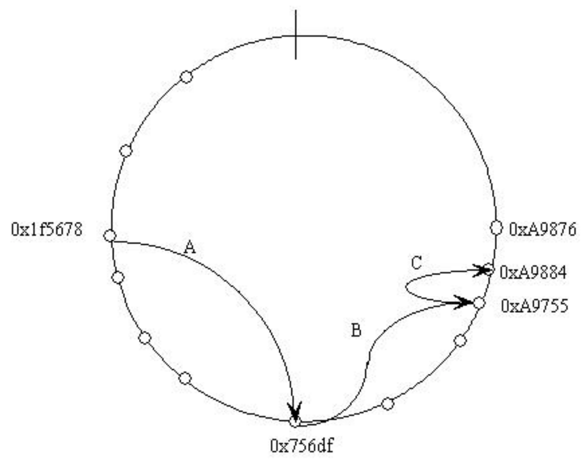

Fig.1

Pastry routes messages to nodes whose nodeId's are progressively closer to the message key.

Pastry nodes are organised around a circular id space. Each node within the Pastry network is assigned a 128-bit unique identifier that is generated typically from the cryptographic hash of, for example, its IP address and users name. E.g. Using the SHA-1 [14] hashing algorithm a string such as "computer science" will produce the hash code "*0bbb843c75b8cb93ceb9d5594e208668484448ee*". Pastry has the ability to route messages between nodes when given a message key. The message is routed to the node whose nodeId is numerically closest to the messages key. Pastry's routing algorithm is efficient, scalable and robust.

The organisation of nodes around Pastry's ring is random. This is due to the way in which nodes are assigned their id's. A hashing algorithm is employed for the purpose of generating the unique codes; a commonly used hashing algorithm is SHA-1. SHA-1 is nonreversible, collision-resistant, and has a good *avalanche effect.* The avalanche effect of hashing algorithms means that given two very similar strings, two very different and non-numerically close hash codes will be produced. Pastry uses this feature as a way of achieving load balancing within the network by randomly placing nodes around the network

2.2 Vector space modelling

It has been shown that a document may be represented as a feature vector. This modelling of a document as a vector is called "Vector space modelling"[15]. In this model each document is considered to be a vector in the term-space. In its simplest form, each document is represented by the *term-frequency* (TF) vector $d_{tf} = (tf_1, tf_2, \ldots\ldots, tf_n)$, where *tf* is the frequency of the ith term in the document. A widely used refinement to this model is to weight each term based on its *inverse document frequency* (IDF) in the document collection. This is commonly done by multiplying the frequency of each term i by $\log(\frac{N}{df_i})$, where N is the total number of documents in the collection, and df_i is the number of documents that contain the ith term (i.e. document frequency). This leads to the *tf-idf* representation of the document in Eqn 1,

$$d_{tfidf} = (tf_1 \log(\frac{N}{df_1}), f2\log(\frac{N}{df_2}), \ldots\ldots\ldots\ldots, f_n \log(\frac{N}{df_n})). \qquad \text{Eqn 1.}$$

In [16] it was shown experimentally, that any measure used should be normalized by the length of the document vectors. In order to account for documents of different lengths, the length of each document vector is normalized so that it is of unit length, i.e. $\| d_{tfidf} \|_2 = 1$. There are two major similarity metrics as regards the vector space modeling of documents [17]. One of them is the angle-based metric that uses for example the cosine function. This is done by calculating the cosine of the angle between the vectors. The cosine function is given in Eqn. 2. Documents may now be compared and a similarity index can be established between two documents using this method.

$$\cos(d_i, d_j) = \frac{d_i \bullet d_j}{\| d_i \|_2 * \| d_j \|_2}, \qquad \text{Eqn 2.}$$

Where "$\bullet$" denotes the "dot product" of two vectors. Since the document vectors are of unit length, the above formula simplifies to

$$\cos(d_i, d_j) = d_i \bullet d_j . \qquad \text{Eqn 3}$$

The ability to represent and compare documents using vector space modelling is a very useful tool, enabling inter-document relationships to be determined more accurately than through the use of keyword matching.

2.3 Clustering

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters) [18]. Document clustering was originally investigated as a means of improving the performance of search engines. Since then document clustering or the cluster-based techniques have been used in domain identification in such areas as radio news and imaging. Cluster analysis allows the identification of groups, or clusters, of similar objects in multi-dimensional space.

Hierarchic clustering has been put forward for its efficiency and effectiveness in Information retrieval. There are numerous documents clustering algorithms. *Agglomerative Hierarchical Clustering* (AHC)[19] algorithms appear to be the most commonly used. However these algorithms have been proved to be slow when applied to large document sets. *Linear time clustering* algorithms have been suggested as the best candidates for large document sets, these clustering algorithms include the K-means algorithm [20] and the single pass method [21].

Given a set of feature vectors, a clustering tree can be constructed. Vectors that are deemed similar, using a similarity index such as the cosine function are placed near each other on the tree and those less similar are positioned farther away. There are many methods available for constructing clustering trees. [18] Provides a good overview and discusses the pros and cons of each method.

3 Putting the Ingredients Together

Having considered Pastry, Vector Space Modelling and Clustering, the next step is to explain how these ingredients can be incorporated to create a p2p system that supports decentralised, scalable and robust content sensitive communities.

3.1 Building a Decentralized indexing service

One of the key elements of the proposed system is the *Pastry routing schema*. Pastry will provide the system with a scalable, robust routing algorithm that is able to route to any node in $[\log_{2^b} N]$ steps on average, where N is the number of nodes and b is a configuration parameter. As stated previously, Pastry routes messages within the Pastry network to nodes whose nodeId's are closest to the key of the message. Within our system we wish to build a decentralized indexing system where nodes "interested" in a certain topic may join communities of like-minded users. In order to do this a sort of rendezvous point where nodes can discover others that "have similar interests" needs to be created. As will be shown, Pastry provides a routing algorithm that can be used to build such a system without any central control.

3.1.1 Creating indexing nodes

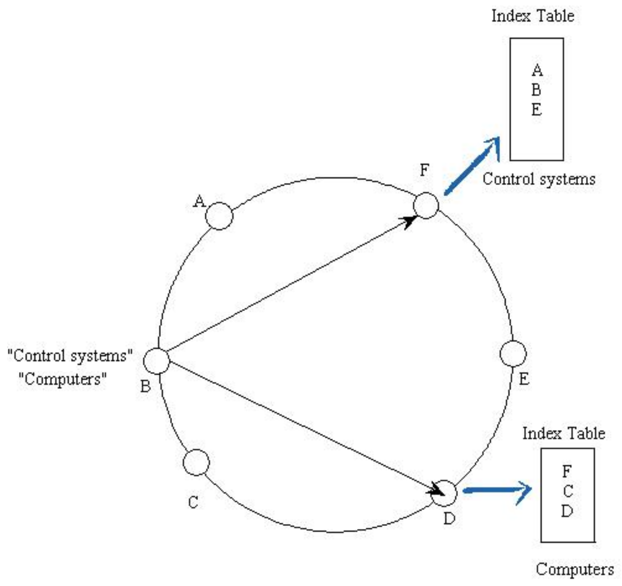

Figure 2
Node B routes a register message with 2 message keys that are the cryptographic hash of the keywords "control systems" and "computers" throughout the Pastry ring. Index tables registering all nodes sharing the same keywords are constructed.

Given a string of characters, a hashing algorithm such as the SHA_1 hashing algorithm will produce a 160-bit hash code representing the string. This property will form the basis for the indexing service.

Once a node has calculated its unique nodeId it may join the Pastry network. The joining procedure is provided by Pastry. A node joining the network will have a set of keywords associated with it that best describes the nodes "topics of interest". These keywords will serve as the basis for discovering nodes sharing similar content. Each of these keywords is hashed to get a hash code for each word. These hash codes will be used as message keys so that Pastry can route them around the Pastry ring to a live node whose nodeId is numerically closest to the 128 most significant bits of the 160-bit key. A registry-message is constructed; this message contains the node details such as its IP-address and *node-vector* (described in section 4.2.1). The same registry-message is routed several times throughout the network for each keyword. Each registry-message uses the 160-bit codes generated from the hashing of the keywords as keys. When the messages have arrived, each destination node is required to register the new node (*see Figure 2)*. The effect of this is that every other node stating "controls systems" as a keyword as in Figure 2 will end up being registered at node F. If an index does not exist at node F, one will be created. The node whose nodeId corresponds to the 128 most significant bits of the hashed keyword will now serve as the rendezvous point or indexing node for all other nodes using the same keyword. Nodes will register at the same point for two reasons:

- A hashing algorithm given the same input string will always produce the same output key. Therefore a registry-message will always get the same key for the same keyword.
- Pastry's routing algorithm routes messages to the live node whose nodeId is numerically closest to the message key. Therefore two nodes will always end up registering at the same point once they share a keyword.

The above only holds true of course when the indexing node remains live on the network. To deal with node failures and hence loss of indices, it is proposed to replicate the index table among the indexing nodes *k* nearest neighbours (*k* is a configuration parameter that determines the number of neighbouring nodes where the index will be replicated). There are a number of other systems (e.g. PAST) that use the properties of DHT systems for similar purposes. When a file is inserted into PAST, Pastry routes the file to the *k* nodes whose node identifiers are numerically closest to the 128 most significant bits of the file identifier (fileId). These nodes then store the file. Other global storage systems built on top of DHT's include OceanStore [12] that is built on top of Tapestry, and CFS [11], which uses Chord.

3.2 Building Content Sensitive Communities

In this section we describe how nodes are compared based on stored content, the organising of the p2p network into communities and how this can be viewed as a two-layer network.

3.2.1 Comparing Nodes Based on Content Stored

Consider a node storing a set of documents that share the same subject content. It can be said that a node's set of documents can be classified under one general heading. This heading will have a relation to the subject content of each of the documents stored. Another way to look at this general heading would be to describe it as the "average subject" of the documents. Consider again the situation where each document has an associated vector representation, derived from the *td-idf* representational model. It is now possible to generate an *average vector* of these document-vectors. This average vector is representative of the average subject content of a particular node's document set. It therefore tells something about the subject content the node "is interested in". A way of deriving such a vector comes from Centroid based classification [22]. Given a set of S documents and their vector representations, a centroid vector C can be defined, which is the average vector of the set of documents. This is given by Eqn 4.

$$C = \frac{1}{|S|} \sum_{d \in S} d . \qquad \text{Eqn 4}$$

These average vectors are called *node-vectors*. Nodes may now be succinctly represented based on the content they store. It is also possible to compare a pair of node-vectors to assess a similarity index between the corresponding pair of nodes by using the cosine measure as described in section 2.2. These tools provide a way of comparing and hence grouping nodes that are similar together within the network. This is described in the following section.

3.2.2 Organising Network into Communities

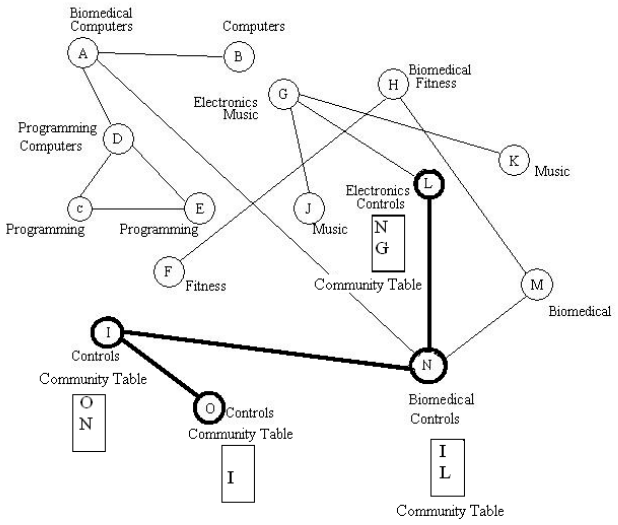

Figure 3
Shows the effect of the community layer formed by community tables

Assuming the node has a specific topic of interest, all documents stored will be related in some way to this main topic. A node-vector can therefore be used to compare nodes that store similar content. If each node stores a community table with a list of nodes that are similar (based on the similarity index from the vector space model described above), document collections of a similar content will then be implicitly linked or grouped together (Figure 3). This can be seen as organising the network into domains whose boundaries are not strictly defined but have a “fading” effect as we jump from one community table to the next. By moving along the path *O→I→N→L,* (highlighted in Figure 3), the document collection moves from “*Controls*” to “*Biomedical”* and then to “*Electronics*”. Node *A* can be seen as the node within the network that stores documents relating both to “*Biomedical*” and “C*omputers”* and thus is the point within the network where the two domains overlap. Any linked group of nodes can be seen as a domain. Each domain can be categorised based on the nodes that have created connections between each other. As they all store similar content, these “communities” of nodes are *content sensitive* in nature because only nodes that store similar content to the community will become part of it.

3.2.3 Two Layer Network

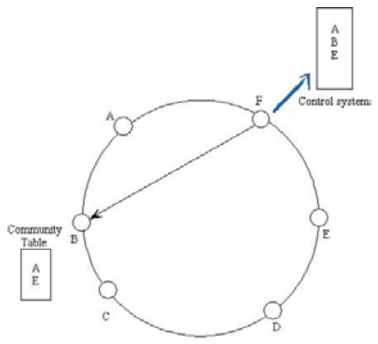

Figure 4.A
The new node B populates its community table with the other nodes whose node-vectors are most similar to its own

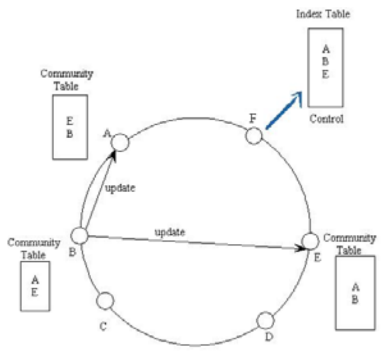

Figure 4.B
Node B then informs nodes A and E that they must add B to their community tables

The network is maintained and organised by employing two layers, the Pastry layer and the Community Layer. Pastry maintains routing tables and *leaf sets* (Leafsets are tables containing neighbouring nodes within a certain number of hops that each node "knows" about [5]). This layer is a means for nodes to find indexing nodes of certain subject areas. The nodes are organised randomly due to the nature of the hashing algorithm employed to create their unique nodeId. The second layer, the *community layer*, is more organised. The ability of the second layer to organise itself is a direct result of the indexing service built on top of Pastry. When a node is added to an indexing node's index table, the indexing node is provided with the new node's node-vector and IP-address. Comparing its node-vector to that of the already registered nodes, the indexing node places the details of the registering node in the appropriate place within a clustering tree. The new node then uses similar nodes within the cluster tree to populate its community table (see Figure 4). All nodes added to the community table are then contacted and asked to add the new node to their community table. This is done for each keyword. After this procedure has been carried out, the new node has links to other nodes that have similar node-vectors and hence have a good probability of being interested and sharing similar content. This means that nodes that "have similar interests" know about each other and can share content directly. This type of organisation makes searching within p2p networks much more efficient as all the content is grouped together into communities and so searches can be directed to a specific area of the network instead of being flooded blindly throughout the network.

3.3 Searching for Documents

Pure flood-based searches have become an essential operation in unstructured p2p networks such as Gnutella [3] and LimeWire [23]. These systems rely on flooding of search messages throughout the network in order to locate files stored by nodes. Within these systems pure flood search requests are given a Time to live stamp (TTL), this TTL sets the number of hops a search message is allowed to execute before “dying”. Pure flood-based search methods have proved inefficient and non-scalable and result in bottle necking within the Internet. However, Pure flooding has been shown to scale well within the Gnutella network up to 10,000 nodes [20].

With a more structured overlay network, flood-based searches can be used while maintaining scalability and cutting down on unnecessary query messages. This is possible by doing a focused flood search where the search is focused on a specific area of the network. It is possible to target a particular part of the system and perform an exhaustive search on those areas that are more likely to contain the type of files being requested. This works within the proposed system by first of all entering a list of keywords. A *search-vector* is then produced. In order to direct the search to an area of the network storing files relating to the keywords, the search-vector is compared locally to nodes within the community table. The search request is then forwarded to those nodes whose node-vectors are the most similar to the search vector. The contacted node performs a flood-search on its community table using the original keywords as the search parameters. The proxy searcher then compiles a list of “hits” and returns them directly to the requesting node. The requesting node may choose to download directly any of the files found or perform another search on a different part of the network.

4. Related work

P2p is becoming an ever-increasing popular research topic. Another similar project that attempts to organise structured p2p networks based on content, is PeerSearch [24]. PeerSearch’s basic idea to achieve greater search capabilities by limiting the amount of nodes that have to be searched. PeerSearch is built on top of CAN. Within

CAN, the total space in the overlay network is divided into topology areas. “Expressways” [25], is an auxiliary mechanism that is used by PeerSearch to deliver high routing performance on top of CAN. The idea of expressways is similar to real world expressways in that it augments CAN’s routing capacity with routing tables of increasing span. As has already been stated, the system proposed here also bears similarities to PAST. PAST uses the power of DHT systems to store files on nodes whose nodeId is closest to that of a fileId. In our proposed system, the same idea is employed to form indexing nodes for particular keywords.

5. Conclusion

By building an indexing service on top of Pastry, it is possible to create a virtual space where users / researchers with similar interests can meet and discover each other in a distributed environment. This enables the construction of communites of users. Organisation of the network in this way has many advantages; it creates a more searchable sharing system. It also creates a more realistic representation of links between files by discovering the semantic relationships that exist through the use of Vector Space Modelling of documents. Structured p2p overlay systems provide exciting and interesting possibilities when combined with Information Retrieval (IR) techniques. These two emerging technologies complement each other in providing a way to share files and data in a distributed enviroment.

There are still many issues to be addressed in the proposed system. Security within the system will need to be looked at in the future. Pastry does not currently provide any security features. The use of keywords to find users may not provide the accuracy needed in discovering other users with similar interests. Words have different meanings and one topic may be classed under several different keywords so an additional meta-layer may need to be added to form true “interest group” communities. Another point that needs more attention is the use of node-vectors in classifying a user’s document set. This use of the node-vector could prove to be naive and a more accurate implementation of this idea may need to be investigated.

The authors have been involved in the area of health informatics and in particular on the difficult problem of sharing fragments of Electronic Healthcare records [26][27] across an intranet or virtual private network. Notwithstanding obvious security difficulties, the work presented here could form the basis for discovering and sharing record fragments between healthcare providers. For example, by integrating the indexing service presented here with an ID management system such as PIDS [28], nodes storing “islands of information” could be grouped into a secure healthcare provider community. This would enable healthcare providers who were caring for a single patient to form a temporary community in order link the scattered fragments of an EHCR for the patient in their care. This will be the focus of future work.

6. References

[1] www.dictionary.com

[2] Napster. www.napster.com

[3] Marius Portmann, Pipat Sookavatana, Sebastien Ardon, Aruna Seneviratne. *The Cost of Peer Discovery and Searching in the Gnutella Peer-to-peer File Sharing Protocol. Networks*, 2001 Proceedings, Ninth IEEE international conference on Networks, 2001 Proceedings Pages: 263-268

[4] C. Greg Plaxton, Rajmohan Rajaraman, Andrea W. Richa. *Accessing Nearby copies of Replicated Objects in a Distributed Environment*. Theory of computing systems1999. Pages 32:241-280

[5] A. Rowstron and P. Druschel. *Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems*. In IFIP/ACM Middleware, Nov. 2001.

[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. *Tapestry: An infrastructure for fault-resilient wide-area location and routing*. Technical Report UCB//CSD-01-1141, U. C. Berkeley, April 2001.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. *A scalable content- addressable network*. In Proc. ACM SIGCOMM'01, San Diego, CA, Aug. 2001.

[8] 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. ACM SIGCOMM'01, San Diego, CA, Aug. 2001.

[9] Kazza. www.kazza.com

[10] A. Rowstron and P. Druschel. *PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility.* HotOS VIII, Schloss Elmau, Germany, May 2001.

[11] Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., AND Stoica, I. *Wide-area cooperative storage with CFS*. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01) (To appear; Banff, Canada, Oct. 2001).

[12] J. Kubiatowicz, et al. *OceanStore: An Architecture for Global-Scale Persistent Storage*. ASPLOS, Cambridge, MA, December 2000.

[13] H. T. Kung, C. H. Wu. *Content Networks: Taxonomy and New Approaches.* To appear in The Internet as a Large-Scale Complex System, Kihong Park and Walter

Willinger (Editors), published by Oxford University Press as part of Santa Fe Institute series, 2002(Check reference)

[14] D. Eastlake, 3rd and P. Jone. RFC 3174: *US secure hashing Algorithm 1*, Sept. 2001. Status: INFORMATIONAL.

[15] Salton, G., Wong, A., and Yang, C. S., *A Vector Space Model for Automatic Indexing.* Communications of the ACM, 18(11):613-620, November 1975.

[16] Willet P., *Similarity coefficients and weighting functions for automatic document classification an empirical comparison.* International Classification, 3: 138-142, 1983.

[17] Jones, W., and Furnas, G., *Pictures of Relevance: A Geometric Analysis of Similarity Measures.* Journal of the American Society for Information Science, 38(6): 420-442, November 1987

[18] Jain, A.K., Murty M.N., and Flynn P.J. (1999): *Data Clustering: A Review.* ACM Computing Surveys, Vol 31, No. 3, 264-323

[19] P. Willet. *Recent trends in hierarchical document clustering: a critical review.* Information Processing and Management, 24:577-97, 1988.

[20] J. J. Rocchio, *Document retrieval systems - optimization and evaluation.* Ph.D. Thesis, Harvard University, 1966.

[21] D. R. Hill. *A vector clustering technique.* In Samuelson (ed.), Mechanised Information Storage, Retrieval and Dissemination, North-Holland, Amsterdam, 1968.

[22] Han, E-H. and Karypis, G. *Centroid-Based Document Classification: Analysis and Experimental results* In Proc. Of the 4th European Conference on Principles and Practice of Knoledge Discovery in Databases (PKDD) September 2000.

[23] LimeWire. www.limewire.com.

[24] C. Tang, Z. Xu, and M. Mahalingam. *Peersearch: Efficient information retrieval in peer-to-peer network*". In Proceedings of HotNets-I, ACM SIGCOMM, 2002.

[25] Z. Xu, M. Mahalingam, and M. Karlsson. *Turning heterogeneity to an advantage in overlay routing*. Technical Report HPL-2002-126, HP Laboratories Palo Alto, 2002.

[26] Jane Grimson, William Grimson, Damon Berry, Gaye Stephens, Eoghan Felton Dipak Kaltra, Pieter Toussaint, Onno W. Weier: *A CORBA-based integration of distributed electronic healthcare records using the Synapses approach*. IEEE Transactions on Information Technology in Biomedicine 2(3): 124-138 (1998).

[27] Jane Grimson, Eoghan Felton, Gaye Stephens, William Grimson and Damon Berry. *Interoperability issues in sharing electronic healthcare records - the Synapses approach.* Proceedings of Third IEEE International Conference on Engineering of Complex Computer Systems, IEEE (1997) 180-185.

[28] CORBAmed document: **formal/01-04-04** (Person Identification Service Specification, v1.1)