Strathprints Institutional Repository

Fongen, A. and Eliassen, F. and Ferguson, I. and Stobart, S. and Tait, J. (2001) *Distributed resource discovery using a context sensitive infrastructure.* In: Proceedings of the Third International Conference on Information Integration and Web-based Applications and Services (IIWAS). IOS Press. ISBN 3854031505

DISTRIBUTED RESOURCE DISCOVERY USING A CONTENT-SENSITIVE INFRASTRUCTURE

Anders Fongen[1]
Frank Eliassen, Ian Ferguson
Simon Stobart, John Tait

Distributed Resource Discovery in a World Wide Web environment using full-text indices will never scale. The distinct properties of WWW information (volume, rate of change, topical diversity) limits the scaleability of traditional approaches to distributed Resource Discovery. An approach combining metadata clustering and query routing can, on the other hand, be proven to scale much better. This paper presents the Content-Sensitive Infrastructure, which is a design building on these results. We also present an analytical framework for comparing scaleability of different distribution strategies.

Keywords: Distributed IR, Resource Discovery, Metadata Clustering, Query Routing

1. Introduction

Most papers describing approaches to Internet Information Retrieval (IR) agrees that traditional IR design using full-text indices does not work well in this area. The problems experienced can be divided into four categories [1]:

1.1. Scaleability

Automatic indexing is commonly done using indexing agents (often called *web spiders*) that read the content of a web page and add indexing information about this page into an index database. In order to keep the index database up-to-date, the indexing agent must employ a polling scheme to find pages that have changed content or disappeared. This process places a communication cost on the indexing agent believed to be proportional to the size of the text volume to be indexed and the rate of change of the content.

The “search engine” (the server that processes queries and returns a result set) needs access to the index database in order to process queries. Most search engines co-locate these two components, so

1. Contact address: The Polytechnic College, Hans Burums v.30, NO-1357 Bekkestua, NORWAY, or anders@fongen.no

all queries and updates to the index database need to be concentrated in and around this site. A centralised site is a candidate for becoming a bottleneck as the size, rate of change and query rate increases [2, 3 p. 146].

With the large growth of both the number of users on the Internet and the amount of information published there, it seems probable that a traditional IR design will not perform well in the future.

1.2. Currency

When following the references shown in a query result set, it often turns out that these information resources are unavailable or have changed contents. Changes in a web page are never reported to the indexing agent, and frequent reindexing of large collections is an expensive operation. Thus, the indexing agent must “guess” which portion of the collection that need frequent re-indexing [4]. As any user of a web search engine has experienced, this is only partly successful: A significant portion of the references in a result set are “dead” links.

1.3. Relevance

Full-text search engines search for words without any context. The search for “bond” returns references to chemistry, finance and film, although it is inconceivable that a user is interested in all the three categories. Finding “hotels in Nice” is impossible with a full-text search engine [5, 6].

1.4. Coverage

The use of web spiders limits the coverage of the search engine to what can be seen through a URL (Uniform Resource Locator), and what is linked to through HTML hypertext references. HTML Framesets (requiring more than one URL to be represented) and database services (showing information as a response to a HTTP POST operation) will never be properly represented in a full-text index. Sullivan [4] calls this “the hidden web”. It is estimated that the best spider tools are able to index no more than 16 % of the entireWeb content [7].

2. Our contribution

Our approach to these problem areas is a combination of distributed metadata clustering and query routing inside a network of forwarding and processing entities, called *members*. We call this network a “Content-Sensitive Infrastructure” (CSI), which will be described in detail during the next section.

The focus of the work is on *scaleability*. We will present a model by which recent research projects can be evaluated in terms of their ability to scale. Many of these systems claim to be scaleable, but fail to analyse or even discuss this property. Our evaluation model suggests the necessary properties for a resource discovery system scaleable to a World Wide Web environment.

Another important property of our project is the use of *metadata*. The use of metadata shows promising solutions to the problems of relevance, currency and coverage.

The rest of the paper is organised as follows: In section 3 we present the design of the Content-Sensitive Infrastructure to some detail. In section 4 we present our framework for evaluation of distributed resource discovery, and explain why most existing design is not scaleable to Internet proportions. Section 5 gives a summary of our formal analysis of the CSI, and in section 6 we discuss some positive side effects of using metadata in resource discovery.

3. The Content-Sensitive Infrastructure

3.1. Members

At the heart of the design of our distributed search system is an infrastructure of network entities (called *members*[1]). Each member has a particular area of interest (its *topic*).

CSI employs a hierarchical means of describing topics whereby any topic can be arbitrarily decomposed into its sub-topics (e.g. computing->programming->java). Since the topics are hierarchical[2], one member can have its topic covered by another member with interest in a broader “parent” topic (figure 1).

Members can accept, process and forward two types of information items: metadata items and query items and are able to act upon the topic information in queries and metadata in such a way that these items are forwarded towards those members with similar areas of interest.

3.2. Mapping to physical network

An instance of a member consists of the CSI servent (server/client - the software plays both roles) and its associated database. One computer may be a host to many members.

1. Since a computer can accommodate several members, the words *member* and *node* has different meanings.
2. Note that the topical hierarchy is an abstract (but static) entity. The CSI is a hierarchy of *real* members (and dynamic).

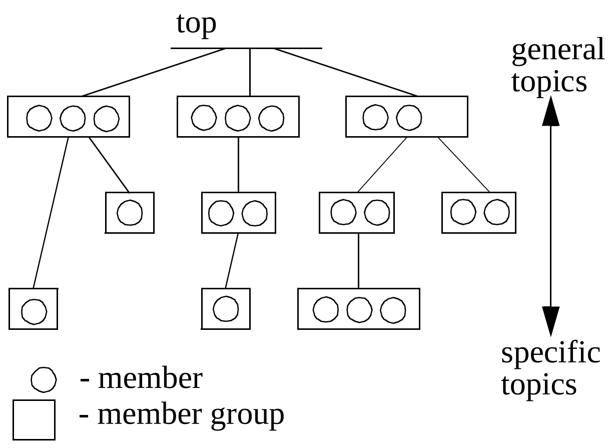
3.3. Member group, tuplets

More than one member can have an interest in a given topic. Together, these members form a member group. We term members of the same member group *tuplets*[1].

3.4. Self-configuration

Any large-scale distributed system must be able to accommodate new processing nodes that join the system without any manual intervention or system configuration. Likewise, the system should allow nodes to leave the system (or crash) without disruption or loss of control. Our CSI design is completely self-configuring, and separate "join"- and "leave"-protocols are responsible for maintaining the structure of the members currently active.

Figure 1: The components and organisation of a Content-Sensitive Infrastructure



It is not necessary that every topic in the hierarchy is explicitly covered by a member, but every topic must be covered at least implicitly through a parent member with interest in a broader category of topics.

The topic is represented internally in the system as a string variable called the *Category of Interest* (COI). The characters of this string shows the *path* from the top of the topic hierarchy down to the member's position in the CSI hierarchy. A COI string can be t.ex. "15TGA", and is only meaningful when interpreted within the context of a "real" topical hierarchy, e.g. the Dewey Decimal Classification system. Also note that any symbol (t.ex. the "G" above) has different interpretations in different positions in the CSI string.

1. We needed a general word for twins, triplets etc. since they are closer related than siblings. We chose *tuplets*.

3.5. Metadata processing

Every item of metadata will have a topic and a unique serial number attributed to it. In addition the item contains a description of “something” (most often a service or a document), and information on where to find it and how to use it. The processing of metadata involves transporting the items to the members having this topic inside its interest (includes members having interest in the corresponding “super-topics”). The members having such interest form a path from one of the top-level member groups[1] to the member group having exactly the same topic of interest as the metadata.

3.6. Query processing

A query is a small, structured[2] document resembling an item of metadata, and the processing of queries involves a *matching* operation between a query and a set of stored metadata, as well as transporting the query to the members expected to have interesting metadata in storage.

3.6.1. Scope of query

To decide which items of metadata is interesting to a query, we employ these rules:

- Metadata associated with specific topics are interesting to queries associated with general topics, given that the topics are in some parent-child relationship.
- Not the other way around

3.6.2. Candidates for processing

A query is initially sent to the member group having the same topical interest. If there is no such member group, the query is sent to the closest parent member group. In this way the rules above are automatically enforced, since this member group holds metadata information about this topic and every sub-topic.

3.6.3. Return of result set

When a query arrives at a member possessing metadata within this topic of interest, the stored metadata will be inspected in order to find metadata similar (above some threshold) to the query. The similarity check may take any information in the query/metadata into account: A list of descriptive keywords, language used or accepted, storage format, client requirements, certifications, age, size, price etc.

1. A top-level member group is one that does not have a parent
2. The structure of all messages in the CSI is based on XML (eXtensible Markup Language)[8].

The metadata found to be a part of the query's result set should be returned to the user agent or the member acting on behalf of it.

3.7. The User Agent

The User Agent (UA) of our system is responsible for the generation of metadata and queries. It assists the user filling out the information, and suggests the topic of the metadata/query based on lexical analysis. It also holds a user profile and a history of previous queries in order to have more information about the user's intentions and preferences.

The UA presents the returned result set in a ranked form. It can inspect the indicated resources on behalf of the user in order to improve the ranking process. It can also launch client software necessary to use the indicated resource.

4. Scaleability issues

From an architectural view, search engines are not very complex. It seems possible to compare the CSI design with alternative designs using an evaluation framework based on analysis of the information flows between its components. The resource of greatest scarcity when applications are distributed over a WAN (Wide Area Network) is the networking capacity. Our chosen perspective is therefore to study the *message complexity* of the different alternatives, i.e. the number of messages sent or received by any node as a function of fulltext volume, index volume, metadata volume and query volume. These four terms are discussed below.

4.1. The full-text volume

Any search engine using full-text indices will need an automatic indexing mechanism. An agent generating indices will need to fetch a given web page periodically in case it has changed content or disappeared. The fraction of the total web content that needs to be fetched and reindexed over a time period can be calculated as a function of *grace period* (the time allowed for an index to be out-of-date with the page content) and the probability for an index to be out-of-date beyond the grace period [9]. An example of Brewington and Cybenko's analysis is that a page need to be fetched every 18th day if the index should have a 0.95-probability that it is at most one week out-of-date. This period decreases to 8.5 days if the index should be less than one day out-of-date with the same probability.

With a total web page volume of 800 million pages, each having an average size of 12 kBytes (numbers indicated by Brewington and Cybenko), a reindexing period of 18 days requires the transport of 533 Gigabyte per day into the index-generating agent[1]!

The number of web pages is believed to grow exponentially over time [3].

4.2. The index volume

The structure most commonly used for storing full-text indices is the *inverted list* [10 p. 82]. The size of the inverted list is dependent on the richness of the information contained in the list, as well as the use of stop words in the indexing process. We estimate the size of the index to be less than, but at the same order of magnitude as the full-text volume.

When distributing indices to several search engines, we will know exactly which portion of the index has changed since the last time of distribution and transmit only this portion (possibly compressed). Brewington and Cybenko’s observations indicate that 17 % of the web pages changes more often than every week. It is reasonable to expect that 17 % of the index has to be transported over this period in order to have an index with 1 week “currency” (related to the “original” index, not to the web pages in question. If we allow the indexing process a 1-week grace period, then the transported index is up to two weeks out-of-date).

If the size of the compressed index is 10 % of the total full-text volume, and 17 % of this needs to be transported from the index-generating agent to the search engine every 7 days, an index volume of 23 Gigabyte needs to be transported every day.

The size of the index volume is also expected to grow exponentially over time.

4.3. The query volume

When a user issues a query to a search engine, it can have the form of a series of search words being *mapped* into a set of relevant web pages, or as a metadata description of the desired documents being *matched* to a set of stored metadata descriptions [11]. In both cases the volume of the query is low (100-200 characters) and the size of the response should be roughly the same when using either full-text-based or metadata-based search engines. The network traffic generated by queries and responses is therefore expected to grow linearly with the number of query operations[2].

1. This number corresponds roughly to a data rate of 1.2 Gigabit/s continuous bandwidth throughout 24 hours.

The number of web users is believed to grow exponentially over time [3]. The popularity of search engines is not decreasing, so the number of query operations generated from each user is not expected to decrease. We therefore expect the query volume to grow exponentially over time as well. The most popular search engines report number of queries up to 40-50 million per day[1]. The volume of information transferred is not known.

4.4. The metadata volume

When using metadata, the index information of a document is not distributed across large list structures, but are self-contained descriptions, made through a manual or automatic process. A metadata description of a web page can be generated inside its web server and be sent to a search engine “just-in-time”. A metadata description of a web page can be of any size, but the dominant standards (e.g. Dublin Core) indicate a typical compressed size of 100-200 bytes (500-1000 characters uncompressed).

A metadata-based search engine could receive metadata every time a web page is updated. For the sake of comparable alternatives, we introduce a lower threshold of 7 days between updates, so that our hypothetical metadata-based search engine has a one-week currency. Over a period of 7 days it will receive metadata describing 17 % of the total population of 800 million pages, each the size of 150 bytes. This amounts to a volume of 2.9 Gigabytes per day.

The size of metadata volume is also expected to grow exponentially over time.

4.5. Distribution strategies

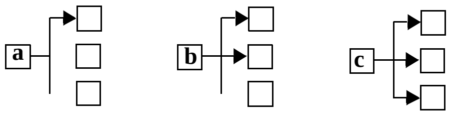
When dealing with distributed resource discovery, the processing model of the architecture has large influence upon the way in which the information volumes need to be transported across the network. We will here present some alternative models, most of them are in use in existing distributed IR projects.

In order to process *queries*, a search engine needs access to an *index*[2] representing the content of a *collection*. In distributed resource discovery there are several collections distributed across several information servers (what is just the case in the World Wide Web). The index can be kept centralised (representing every collection) or distributed (representing a subset of collections).

2. Local- or LAN-caching of query results is not considered here.
1. http://www.searchenginewatch.com/
2. At this stage of the discussion, the word “index” includes full-text indices (e.g. inverted lists) and metadata.

In the following analysis, the symbol *F* denotes the Full-TextVolume, *I* denotes the Index Volume,*M* the Metadata Volume and *Q* the Query Volume. The symbols used are explained in figure 2.

Figure 2: The symbols used in this discussion: (a) shows an information flow distributed across several recipients, (b) shows a flow where two or more recipients receive the same information (multicast), (c) shows a flow where everyone receives the same information (broadcast)



4.5.1. Centralised index (centralised search engine)

In this case the entire query volume is processed in a central search engine. Also, in order to keep the index current, the entire index volume (from an indexing agent) or metadata volume must be transported into the same site (figure 3). The message complexity of the search engine in this case is *O(Q+M)* or *O(Q+I)*. Since both *Q*, *I* and *M* are expected to have exponential growth rate, this design is not future-proof. “Traditional” search engines like Google *(www.google.com)* are using this computational model[1].

Figure 3: The information flow using centralised index. SE=Search Engine, IA=Indexing Agent

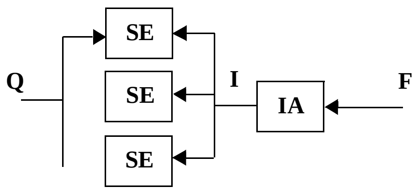
$$\begin{tikzcd} {} \arrow[r, "Q"] & \boxed{SE} & \boxed{IA} \arrow[l, "I"'] & {} \arrow[l, "F"'] \end{tikzcd}$$

4.5.2. Replicated index (distributed search engines)

Several search engines can use the same index at different sites provided that the index is replicated between them. In this case the query volume will be distributed between *n* search engines, but every one need to receive the full index or metadata volume (figure 4). The message complexity becomes *O(Q/n+I)* or *O(Q/n+M)*. But an indexing agent need to send replica to several sites, making the message complexity *O(nI)* for this component[2]. AltaVista *(www.altavista.com)* is using a variation of this distribution strategy. There are search engines on different continents, but they do not give the same result to a given query. Only parts of the index database seems to be replicated between the search engines.

1. Any references to the design of commercial search engines are based on assumptions, since the internals of these systems are not open to the academic community.
2. The workload of replication can also be shared between the search engines, making the message complexity slightly different.

Figure 4: The information flow using replicated index



4.5.3. Distributed index (distributed search engines)

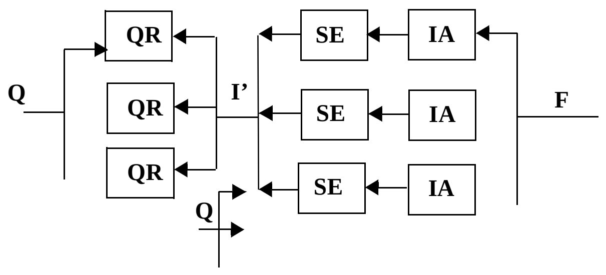
When the index is distributed, query processing must consult possibly several indices, depending on which collections are considered interesting. The IR literature refers to this problem as the *collection fusion problem.* In the area of distributed resource discovery this is called *Query Routing*, since some entity in the system (a Query Router) must decide which search engines a query should be forwarded to. This decision is based on *Forward Knowledge* in the Query Router, a type of information describing the characteristics of the collection either as full-text indices (called *Centroids* in [12, 13], other approaches are found in [5, 14]), or as a topical classification [15]. As these characteristics change (which they constantly do when the collection is a web server), a new flow of information is necessary, keeping the Query Routers up-to-date with current Forward Knowledge. The volume of this flow (called *I'*) has not been analysed, but it is expected to grow in proportion with the Index volume (exponential over time). Figure 5 shows the principles of this model.

Using this model, the “ideal” message complexity of a search engine becomes $O(Q/n+I/m)$ or $O(Q/n+M/n)$, and that of the Query Router becomes $O(Q/m+I')$ (where m is the number of query routers). This is only true under the situation where the query is forwarded to only one search engine. Under conditions where queries are sent to several search engines, the complexity of the search engine grows towards $O(Q+I')$.

In order to send queries to only one (or a few) search engines, the collections must be distinct and different. When using an indexing agent under World Wide Web conditions, there is no obvious method by which to find collections sufficiently different so that the Query Routers can discriminate well between them[1]. An approach where the collections (web servers) themselves generate the index (inverted lists or metadata) rather than letting an indexing agent rely on arbitrary hyperlinks seems to be promising.

1. This problem is closely related to an *information clustering* problem.

Figure 5: The information flow using distributed index and Forward Knowledge. QR=Query Router



4.5.4. Using static forward knowledge

The only way to avoid sending the *I'* volume to every Query Router is by relying on *Static Forward Knowledge* i.e. a Forward Knowledge that never changes. *Static collections* can be characterised by static forward knowledge, but static collections do not exist in the World Wide Web. Static forward knowledge can be obtained if each collection contains information with a given *class*, e.g. information related to a specific topic. By attributing a *topic* to the query, a query router can forward the query to the correct collection. Such collections are hard to find on the World Wide Web, so this is not a realistic solution.

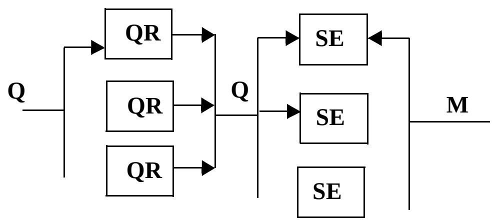
4.5.5. Using metadata associated with a *topic*

The use of metadata, however, enables us to process queries on a *metadata collection* instead of a full-text index. A metadata collection can represent information from several collections. It is thus possible to make “topical” metadata collections that contain information about resources only within a given topic. This approach requires that topics can be attributed to both metadata and queries.

One metadata item that describes a particular internet resource (e.g. a document) can be stored inside a metadata collection together with metadata descriptions of similar resources. Since the collections can be stored on different nodes in the network, we call this process *distributed metadata clustering.*

Queries can in the same manner be routed to a search engine based on the topic attributed to it. Both the *Q* volume and the *M* volume can be distributed across several nodes using this approach, and the *I'* volume do not exist, since we only use static forward knowledge. See figure 6.

Figure 6: The information flow using distributed metadata clustering and static Forward Knowledge.



The message complexity of the query router is under ideal conditions *O(Q/n)* and of the search engines *O(M/m)*. The CSI employs this model.

4.6. The design of forwarding protocols

All the research projects known to us inside Internet resource discovery fall into one of the categories mentioned above. None of them are using distributed metadata and query routing with static forward knowledge, except Hinds [16], which relies on static collections. On the basis of the given analysis we therefore claim that none of these products will be able to scale to the future Internet proportions.

During the construction of the software algorithms and the networking protocols, it turned out to be very difficult to avoid traffic bottlenecks that limits the theoretical scaleability shown above. A thorough formal analysis of the design was necessary in order to prove that the design is scaleable in practice. The need for fault-tolerance, self-configuration and flexible number of members also needed to be addressed.

5. Analysis of the Content-Sensitive Infrastructure

The scaleability of the CSI was formally analysed using probability theory. By assuming that members, metadata and queries have a COI according to some probability distribution, it is possible to express as a function of member count:

- the probability for at least one member to have a given COI
- the average size of a member group with a given COI
- the probability that a member group has children interested in a given sub-COI (sub-topic)
- the fraction of the metadata volume arriving at a given member
- the fraction of the query volume arriving at a given member

The full analysis is omitted in this paper for the sake of brevity, but it can be summarised as follows:

- The size of the member groups contributes to the distribution of the metadata volume (*M*). The message complexity of a member when processing metadata is $O(M/n)$, where n is the number of members.

- The number of member groups contributes to the distribution of the query volume (Q). The message complexity of a member when processing queries is $O(Q \cdot h(n))$, where the function h is a monotonically decreasing function with a horizontal asymptote at a positive small number (that number is inversely proportional with number of different topics).
- The fault-tolerance (robustness towards transport error) of the CSI increases with growing number of members.

6. Advantages of metadata

This paper initially described a lot of problems inside the area of resource discovery, but have mostly discussed the scaleability. We shortly comment on the other issues:

6.1. Currency

Metadata can be generated by a process local to the collection so that the volume of metadata indices is optimised. Metadata can also include a “life-time”, after which it is considered irrelevant and is deleted from the server.

6.2. Coverage

Metadata can also be generated from a manual process, so that it can represent resources invisible to web spiders.

6.3. Relevance

Metadata can assign a *topic* to the resource. It can also describe the resource in terms of keywords, date stamps, certificates, client requirements and other formal or informal characteristics of the resource. A query conveying the precise requirements of the user can thus be more accurately matched to a selection of resources.

7. Conclusion

Distributed metadata clustering combined with query routing is a promising approach for distributed resource discovery under Internet conditions (size, diversity and rate of change). This paper has described a prototype system based on this idea together with summary of analytical and experimental results. This paper has also presented an evaluation framework for comparing the scaleability of different distribution strategies. The results from using this framework suggests that few existing systems inside this area is scaleable to Internet proportions.

8. Future work

The CSI is in its early experimental stage and only simulated testing has been done so far. The simulation verifies the analytical results. The future work on this project will include the following activities:

- The development of necessary networking code to deploy CSI member nodes in the Internet
- The development of a User Agent
- The study of how a mixed (and evolving) set of metadata representations can exist in the CSI at the same time
- The effect of letting a member change its Category of Interest in order to respond better to local queries. This would resemble a “cooperative caching” system for metadata
- Evaluation of overall system efficiency and effectiveness when all these ideas are working together

9. References

[1] GREY, D.J., P. DUNNE, and R.I. FERGUSON. *AgentSeek: a means of efficiently locating resources on the World Wide Web using mobile, collaborative agents*. in *Workshop2000 on agent-based simulation*. 2000.

[2] KOSMYNIN, A., *From Bookmark Managers to Distributed Indexing: An Evolutionary Way to the Next Generation of Search Engines.* IEEE Communications Magazine, 1997. 35(6): p. 146-151.

[3] KOBAYASHI, M. and T. KOICHI, *Information Retrieval on the Web.* ACM Computing Surveys, June 2000, 2000. 32(2).

[4] SULLIVAN, D., *Crawling Under the Hood, and Update on Search Engine Technology.* Online, 1999. 23(3): p. 30-38.

[5] CHEKURI, C., et al. *Web Search Using Automatic Classification.* in *Sixth International World Wide Web Conference, Poster POS725.* 1997. Santa Clara, California.

[6] SUGIURA, A. and O. ETZIONI, *Query Routing for Web Search Experiments: Architecture and Experiments*. 2000, The ninth International World Wide Web Conference 2000.

[7] SHERMAN, C., *What's new withWeb Search.* Online, 2000. 24(3): p. 27-34.

[8] W3C, *Extensible Markup Language (XML)*. 1998, World Wide Web Consortium.

[9] BREWINGTON, B.E. and G. CYBENKO, *Keeping up with the ChangingWeb.* IEEE Computer, 2000(May 2000).

[10] KOWALSKI, G.J. and M.T. MAYBURY, *Information Storage and Retrieval Systems, theory and implementation. 2nd ed.* 2 ed. 2000: Kluwer Academic Publisher.

[11] KORFHAGE, R., *Information Storage and Retrieval.* 1997, New York: John Wiley & Sons, Inc.

[12] WEIDER, C., J. FULLTON, and S. SPERO, *Architecture of the Whois++ Index Service, RFC1913*. 1996, IETF.

[13] WANG, B., et al., *Standards in the CHIC-Pilot distributed indexing architecture*. 1998, Computer Networks and ISDN Systems 30 (16), pp.1571-1578.

[14] YUWONO, B. and D. L.LEE. *Server Rankning for Distributed Text Retrieval Systems on the Internet*. in *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications*. 1997. Melbourne, Australia.

[15] KIRRIEMUIR, D., et al., *Cross-Searching Subject Gateways*. 1998, D-Lib Magazine January 1998.

[16] HINDS, N., RAVISHANKAR, CV. *Managing Metadata for Distributed Information Servers*. in *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*. 1998. Koala Coast, Hawaii: IEEE Computer Society.