MAAY: a self-adaptive peer network for efficient document search

Joaquín KELLER, Daniel STERN
France Telecom R&D
38, rue du Général Leclerc
92794 Issy-Moulineaux - France
{joaquin.keller, daniel.stern}
@rd.francetelecom.com

Frédéric DANG NGOC
IRISA / INRIA-Rennes
Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 RENNES Cedex - France
frederic.dangngoc@rd.francetelecom.com

Abstract MAAY[1] *is a network of peers aimed to search, publish and retrieve documents. Each node (or peer) handles indexes, acts as a document repository/server, and routes search requests. We propose a network topology in which nodes tend to aggregate by interests and thematics. Assuming that a given user only have a limited range of topics of interest, she do not need to search the whole web but only part of it. In* MAAY *that can be found in their neighborhood. The replication and caching is extensively used in* MAAY *to improve availability of documents and to ensure anonymity of authors and publishers. We describe the related work concerning search engines and peer-to-peer systems then we describe the architecture of* MAAY *and its routing, caching and replication algorithms.*

Keywords: peer-to-peer, search engine, adaptive network, distributed search mechanisms

1 Introduction

Classical search engines like Google contribute to make web pages more reachable by allowing search by keywords in (almost) the whole web. However, dynamic content web pages and transient web servers are not well referenced and remain unreachable.

In peer network based systems like Gnutella or KaZaA, the search is performed locally on alive nodes, directly in the documents so freshness of the results is ensured. But since only few nodes are visited during a search only the most replicated files are reachable. Also, unlike the web, these systems make available poorly structured contents: in file sharing systems, "documents" are flat lists of files (plus their meta-data) and there is only one document per node and no links between documents.

Like Gnutella and many others, MAAY is implemented by a network of peers in which each node handles indexes, acts as a document repository/server, and routes search requests. However, the MAAY project intends to build a system that enables searches in both worlds: the structured web of documents and the transient contents. Moreover, MAAY will also make reachable the transient structured documents (*e.g.* personal web servers).

Assuming that users are interested only in a limited range of topics that are somehow related, *à la* Google exhaustiveness is an overkill: a given user will search few of the millions of available keywords and most of the indexed pages are totally irrelevant for her. Consequently they do not need to search the whole web but only a part of it. For example, most people need not to search Amharic or Tagalog documents since they (usually) do not even understand the language nor the culture.

MAAY system does not aim at achieving complete exhaustiveness, but only user perceived exhaustiveness, *i.e.* a user must receive all the relevant (from her point of view) responses to all the queries they issue.

To do so, in the MAAY peer network, peers tend to aggregate when they share keywords, when they give each other relevant responses or when the doc-

[1] MAAY should be pronounced as 'my' in English or as 'mailles'(meshes) in French.

uments they store are hyperlinked. So eventually, for a peer, all the relevant part of the network will tend to be in its neighbourhood. All the contents that might interest a peer are just a few hops away. The peer network "learns", auto-organizes itself, in order to give the user the perception of exhaustiveness.

In the first part, we describe the motivations through related works that drive us to the creation of MAAY. Then in the second part, we explain the principles of MAAY and the algorithms used to aggregate nodes, the local rules to route queries, and the replication and caching policies. We finally conclude by describing how to deploy MAAY on the Internet and the future works to extend it.

2 Related work

2.1 Search engines

The search engines like Altavista, Lycos and the most famous Google [1, 2] allow to seek pages in the whole web according to a set of keywords. They are composed of a crawler, an indexer and a search engine.

The crawler is in charge of the retrieval of all the pages of the web. To do so, it starts by retrieving pages from a list of URLs and then follows the hyperlinks in the pages and do so recursively. But since the web counts more that 3 billions of pages, and the system is centralized, the crawling takes a lot of time: more than one month is required in average for Google to crawl (almost) all the web. This is responsible of the outdated results returned by the search engine.

The indexer analyzes the page to attribute a score for each keywords in the page (according to the frequency of the keyword, its location in the page, its size, ...) and store them using huge indexes.

The search engine answers queries of users by looking at the indexes. It returns all the documents pertinent to the query and sorts them according to their score for the keywords of the query. In order to improve the relevance of results, Google uses in addition the Pagerank factor to sort the pages by their popularity. Since the more a page is popular, the more it interests people, so popular pages are more relevant for most users. Although, Google works quite well, we have often experienced modifying the request in order to find the expected results. In fact, the problem has two origins. First the web pages are very numerous, so thousands or millions of pages can be returned if popular words are used in the query. Second the words can have several meanings depending on the language and the context.

The centralization of search engines can imply a bottleneck and a single point of failure. Moreover, they can easily log queries of users and identify them with their IP address.

2.2 Semantic web

The semantic web [3] proposes to associate to each document a description (or meta-data) to let applications have a better understanding of the document. As queries and documents can be "understood" by a semantic search engine, the results should be more relevant.

But semantics are not universal and a document can be interpreted in different ways. For example, a website about Saddam Hussein can be relevant for mustache fetishists, an interpretation far from universal but still relevant for a minority. Software that automatically generate meta-data, or users that annotate documents manually like in the Annotea project[2] just give their own interpretation that will not satisfy everybody.

2.3 Gnutella

Gnutella [4] is a protocol used to search files on a network of peers. The search algorithm floods the network using a TTL (Time To Live). In order not to saturate the network, the TTL is limited to a small value so just a small part of the peers are visited. In addition, the topology of the network does not depend on the semantic of peers and since the search is done locally on the visited peers, there is little chance for a peer to be in contact with peers satisfying its research unless this information is heavily replicated. Nonetheless, The good point of gnutella is that transient nodes can be reached.

[2] `http://www.w3.org/2001/Annotea/`

2.4 Freenet

Freenet [5] is a peer-to-peer network that enables the publication, replication and retrieval of data while protecting the anonymity of both author and reader.

In Freenet, each document has an associated ID generated by a hash function. The system allows only the retrieval of documents by their associated ID, not the search by keywords. The routing of request and the publish process make the documents having a close ID aggregate but not for documents close semantically.

The publishing process and the retrieval process contribute to the replication of documents in the network, so the publishers of documents are quickly hidden. The requester is also hidden since the forwarding of request avoids peers to know if a received request has been sent by the last sender of just forwarded.

3 MAAY

3.1 Principles

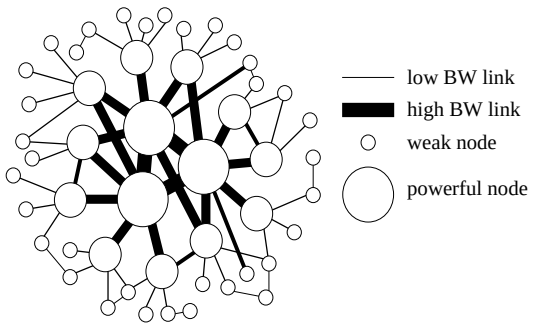
The MAAY network is composed of several MAAY peers. Every peer has a node ID (generated by a SHA1[3] hash function that allows to identify uniquely them and to handle problems raised by possible IP address changes. Each peer handles a document repository which contains the intentional[4] files that it wants to share (such as pages on a website) and also other cached documents (it will be explains later). Each document have a document ID also generated by SHA1 applied on its content. The peer keeps several information about the document such as title, size, and nodes that have a replicate of it. Every peer also uses an internal search engine to index all the documents in its repository and to search a document according to a set of keywords. A result for a search is composed of all the documents satisfying the queries, their descriptions, their relevances according to the keywords of the query (depending on the frequency of the keyword in the document, its location in the document, its size, ...) and document excerpts containing the keywords.

Like other peer-to-peer systems, every MAAY peer P has several neighbours. These neighbours are simply known to P by their IDs. The knowledge process is inherent to MAAY: P learns neighbours IDs either when it receives responses to its (own or forwarded) queries, or when it receives a query (whether it will answer it or not). As peers are not "permanently" connected, they cannot know if their neighbours are still alive, but it does not matter since they usually have lot of neighbours and the queries are simultaneously sent to several peers.

The interest of a peer is defined by a set of keywords. In order to define its own interest, a peer P looks at the topics of all its previous requests and also at the intentionally stored documents. To know a neighbour P''s interest, P looks at the keywords used in all the queries sent by P' that go through P or reponses received from P. A peer that has run a request Q - so that have received lot of responses for its request - is the most capable to answer queries related to the topics of Q. To provide anonymity when downloading a document, a peer may ask (with some probability) a neighbor to do it, So a peer receiving a document query from P' could not know if the query was intentionnaly emitted by P' or just forwarded. But since request are forwarded each time to the peers the most interested by the topic (from its point of view) of the query, we can expect the node to be interested by the queried document.

Topology of MAAY network is organized according to two main "forces". The first one is related to the node capabilities (bandwidth, CPU, disk storage) in order to make nodes aggregate in clusters of "servers".

Figure 1. Network topology of peers related to nodes capabilities



As shown in figure 1 peers are more likely to be

[3] http://www.w3.org/PICS/DSig/

[4] put by the user itself with its knowledge

linked with powerful peers so biggest nodes will agregate together. Smaller node will connect, in a start like topology, to aggregates of bigger nodes. This helps in load balancing and in reducing the network diameter improving the latency of the system.

The second force corresponds to interest similarities. In figure 2, there are three kinds of music; each peer is located in a semantic position according to its music interest. We can notice in this

Figure 2. Network topology of peers related to nodes interests

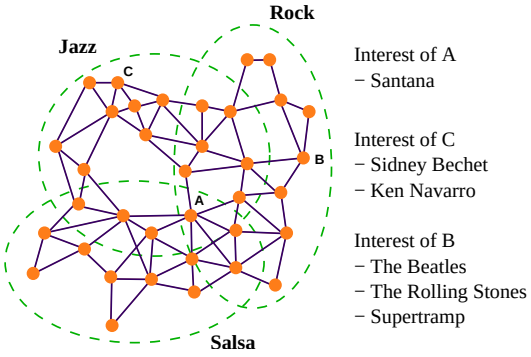


figure that each peer is connected to other peers sharing common music interests and the more two peers are distant, the less they share common interest.

These two forces combine in order to self-organize the network in such way that neighbors are concerned by the same topics and optimize node load and path lengths.

3.2 Data structures

In MAAY the cache is extensively used. Each peer P maintains in several tables the most useful information extracted from the received requests and responses:

- description of documents (title, size, nodes which can provide the files)
- the keywords that describes the documents and their relevance for the document
- information on nodes (node ID, current IP address, bandwidth, CPU, disk storage)
- the interest of nodes (node ID, keywords).

From this information several scores useful for the routing are computed. Table 1 summarizes these scores. A line represents a full table whose columns are the shaded cells.

Table 1. MAAY Information System

MAAY table	DocID	Keywords	Node	Score
Node interest for kw		■	■	■
Affinity with node			■	■
Popularity of doc	■			■
Doc is on node	■		■	■
Kw elevance in doc	■	■		■
Kw popularity		■		■

Each score is stored in a different table:

- the **interest of a peer for a keyword** is computed using all the previous requests done by a peer on this keyword but also on the number of responses supplied by the peer for this keyword.
- the **affinity with a node** represents the amount of common interests shared with a node P'. To do so, P uses the scores above describing the interest of P' for each of its keywords and compare them with its own interest. Node capabilities are also considered since the more a node is powerful, the more it is attractive.
- **the popularity of a document** depends on the number of times the document is downloaded (and so replicated) and also on the date of its last download.
- the **availability of a document on a node** represents the probability of the peer to have the document available on its repository. This value depends on the popularity of the document, the interest of the peer for the document and the document recency.
- the **relevance of a set of keywords for a document** can be obtained either by asking its internal search engine or by asking a neighbour (which may have the results in its own table).
- **keyword popularity** depends on the number of uses of the keyword in the received or emitted request and on the date of its last use.

The keywords associativity between two keywords is calculated by looking at the multi-keywords queries (*e.g.* maille mustard). The more

a keyword is used with another, the more they are related. The same applies to keywords that often appears together in documents. If the keyword is definitely unknown, the peer calculates the keyword distance to its known keywords by using the Levenshtein distance (also known as the edit distance) to use the closest know keyword instead. This distance takes into account the number of insertions, substitutions and deletions necessary to change a word into another.

Since the cache is limited we have to choose the information to wipe out. In a general manner, the peer deletes first the information that do not interest it. The first thing it can erase is the document content itself if it is no more requested or not interesting for the user; it keeps only the information describing it. Then, if more room is needed, keywords relevance entries in document table, related to documents or keywords rarely used in queries, are deleted. And finally nodes entries and their associated interest can be erased when they become unavailable or no longer interesting.

3.3 Requests

There are two types of request in MAAY: the search request and the download request.

A search request may concern three different criteria:

- on one or more **keywords**
  e.g. keywords = "(computer or laptop or notebook or PDA)
  and Compaq"
- on the **document ID**
  e.g. docId = 76e6a307955d1417
- on the **node ID**
  e.g. nodeId = 28a06b40de688626

The node ID is used to identify a peer even if it changes its IP address. It can be a combination of these three.
e.g. keywords = "car and Renault" and
nodeId = 28a06b40de688626

A download request is composed of the document ID and also keywords describing it which allows us to find it.
e.g. docId = 28a06b40de688626 and keywords ="chain mail"

There are three types of response for a search request:

- the **list of nodes** that are interested by the topics of the query
- the **list of document IDs** and their description (title, size, date,...)
- the **relevance of keywords in document**

3.4 Routing

Unlike Freenet, CHORD [6] and CAN [7] on which routing is done on ID, MAAY routes the queries according to the keywords of the query [8, 9, 10, 11]. When a peer forwards or sends a request, it sends it to a set of peers and also searches in its own repository thanks to an internal search engine. To choose the set of peers, it looks at its tables in order to rank nodes according to: their interest for the topic of the query, their capabilities (bandwidth, storage, ...) and their affinity with the peer. The responses are then sent back to the last sender peer and so on until the originator of the request is reached. We use a TTL like Gnutella but the requests do not flood the network since only part of the neighbor are contacted. In order not to have too much answers that can generate lot of traffics, the number of peers contacted depends on their levels of interest since the more they are interested, the more results they can provide.

In some cases, a peer does not know any neighbours interested in the request. It then will look for keywords associated to this request' keywords and use them instead for the routing. In order to find documents related to rare keywords, we define a process of document declaration as follows. When a peer P needs/wants to notify documents (whether they are new or not), it sends to its neighbours search requests on the topics (set of keywords) related to these documents. This enables others to be informed of P's (new) interests but also for P to meet others peers interested by these topics. If any words in the query are unknown to them (new emerging words for example), the query is routed according to the other known words. Search request sent by publisher P_p and search query sent by a searcher P_s are routed to the peers the most interested by the topics of the queries. We can expect that these peers will put P_p and P_s in relation. For example, suppose that a peer P would like others to know that it has documents about Maay (that is a

rare keyword at the moment). Since MAAY is also a search engine, it sends a search request to find nodes interested by Maay *and* search engine. But since no nodes have heard about Maay, the request is routed on nodes interested by search engine. By receiving the request, they become aware that P is interested (and so will be able to provide results) about Maay and search engine. So if another peer searches for Maay, that it associates to search engine, it just sends the same type of request. This request is then routed to nodes interested by search engines. And we can expect these nodes to have received the request from P before theirs, so they can forward the request to P.

To improve the search mechanism, a peer receiving a search request can (with a probability $p < 1$ to ensure anonymity) adds its own ID and IP address to the message before forwarding it. Therefore a peer receiving the request would know more peers interested by the topics of the query by looking at the list of nodes in the message. This information is also useful in case of a peer that logs out during the search process: instead of forwarding the responses to the failed peer, it is forwarded to the previous peer in the list.

Since the routing is not done on ID like in Freenet, it is difficult to find documents according to their IDs. To solve this problem, the peer emits its request with the document ID and also adds all its known document-related keywords. The nodes forwarding the request do the same (they may have new keywords related to this document). So on the one hand information on this document is improved, in the other hand it is forwarded to the nodes most interested by (or having) the document.

If a peer P changes its IP address (after a reconnection and get a new IP address with DHCP for instance), either it contacts its known neighbours to notify them of its new address, or it uses the documents publishing process to find new neighbours. Direct connection to the old reference will fail but since the requests concerning P's interests are routed to its neighbours before reaching P, these neighbors can notify others of its address change.

MAAY routing mechanism allows to adapt to both capabilities and interests of the nodes and therefore should provide efficiently appropriate results.

3.5 Retrieval

Several peer-to-peer systems - *e.g.* KaZaA, OpenNap, new implementations of Gnutella provide multisource downloading that enables to retrieve a file from different locations, each providing a chunk of it, which improves the downloading duration.

To download a document, a peer P must have the document ID and also the ID of the node that can provide the document. To ensure anonymity the following mechanism is applied: the downloader may send the request directly to the provider peer or to another peer (this one may repeat the mechanism); the choice is done according to a certain probability depending on the desired anonymity level. Let P' be the peer which chooses to forward the query to the provider node. A provider peer contacted by P' may return to P' not only the requested document but also a list of peers that could provide a replicate (*i.e.* all peers having in the past downloaded the same document). P' can then download in parallel other chunks of the file on the other nodes, which can in their turn returns other provider peers. Then, P' forwards the file to the downloader peer through the intermediate peers. At the end of the process, P and all the intermediate peers have the requested document in their cache and know many peers having a replicate.

When a peer requests a document download it adds in the request the keywords used when searching the document. This information is used by the provider peer to give a more accurate description for the document.

4 Conclusion

MAAY is a peer-to-peer search system inherently self-adaptive. It takes into account, on the one hand, interests shared by several peers and, on the other hand, the peers networking and computing capabilities. This should allow every peer to find in its neighbourhood almost all relevant responses.

We are now in the phase of improving the algorithms and tuning parameters (affinity, popularity, reliability, ...) that are involved in request routing,

cache and replication policies. This phase will be performed with a simulator.

MAAY deployment on the Internet should be attractive, as a huge number of pages could be indexed by the web servers having a MAAY node. In addition, each web server may index frequently its own pages, which guarantees up-to-date results, and the replication ensures a better availability. From clients point of view, it will improve the relevance of the answers to their requests. A MAAY node comprises two modules. The first one is a MAAY client compatible with the Gnutella protocol that can either link to MAAY peers or to Gnutella peers, and therefore use both the existing resources of Gnutella networks and MAAY functionalities. The second one is an HTTP proxy with several functionalities: logging all the pages visited by the user to define her interest; caching them; indexing them to share them with others; allowing transparent multisource download. It can also be used as a graphical user interface for the MAAY client.

We plan to study the dynamic content search (*e.g.* news pages, weblog) based on the date, to provide a chronological archive of the most interesting pages.

The success of MAAY (or of a similar system) might be the emergence of a new kind of web, richer and more dynamic.

References

[1] C. Ridings and M. Shishigin, "Pagerank uncovered," 2002. [Online]. Available: http://www.supportforums.org/pagerank

[2] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1–7, pp. 107–117, 1998.

[3] W3C, "Semantic web," 2002. [Online]. Available: http://www.w3.org/2001/sw/

[4] C. D. S. Solutions, "The gnutella protocol specification v0.4," 2001. [Online]. Available: http://www.clip2.com

[5] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," *Lecture Notes in Computer Science*, vol. 2009, pp. 46+, 2001.

[6] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable Peer-To-Peer lookup service for internet applications," in *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001, pp. 149–160.

[7] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *Proceedings of ACM SIGCOMM 2001*, 2001.

[8] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," *International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, July 2002ICDCS 2002*, 2001.

[9] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," *International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, July 2002*, 2002.

[10] M. Bawa, R. J. B. Jr., S. Rajagopalan, and E. Shekita, "Make it fresh, make it quick – searching a network of personal webservers," *WWW2003, Budapest, Hungary, May 2003*.

[11] F. M. Cuenca-Acuna and T. D. Nguyen, "Text-based content search and retrieval in ad hoc p2p communities," *Internal Report Department of Computer Science, Rutgers University*, Apr. 2002.