Information Retrieval and Filtering over Self-Organising Digital Libraries

Paraskevi Raftopoulou[1,2], Euripides G.M. Petrakis[1], Christos Tryfonopoulos[2], and Gerhard Weikum[2]

[1] Technical University of Crete, Chania, Crete, 73100, Greece
[2] Max-Planck Institute for Informatics, Saarbruecken, 66123, Germany
{paraskevi,petrakis}@intelligence.tuc.gr
{trifon,weikum}@mpi-inf.mpg.de

Abstract. We present iClusterDL, a self-organising overlay network that supports information retrieval and filtering functionality in a digital library environment. iClusterDL is able to handle huge amounts of data provided by digital libraries in a distributed and self-organising way. The two-tier architecture and the use of semantic overlay networks provide an infrastructure for creating large networks of digital libraries that require minimum administration, yet offer a rich set of tools to the end-user. We present the main components of our architecture, the protocols that regulate peer interactions, and an experimental evaluation that shows the efficiency, and the retrieval and filtering effectiveness of our approach.

1 Introduction

Research in the area of peer-to-peer (P2P) data management has lately given considerable attention to Semantic Overlay Networks (SONs) [1–3]. In a SON, peers that are semantically, thematically or socially close (i.e., peers sharing similar interests or resources) are organised into groups to exploit similarities at query time. SONs, while being highly flexible, improve query performance and guarantee high degree of peer autonomy. This technology has proved useful not only for distributed information sharing, but also as a natural distributed alternative to Web 2.0 application domains such as decentralised social networking in the spirit of Flickr or del.ic.ious. Although SONs do not offer accurate location mechanisms like structured overlays (e.g., Distributed Hash Tables-DHTs), they are better suited for loose P2P architectures due to better support for semantics and their natural emphasis on peer autonomy. Query processing in a SON is achieved by identifying which cluster is better suited to answer the query and routing the query towards a peer in that cluster. This peer is then responsible for forwarding the query to the other members of the same cluster.

Currently, document collections are fragmented across different Digital Libraries (DLs) due to copyright issues that prevent the owners to share their documents. To deal with this issue, a number of P2P architectures (most using a DHT as the underlying routing infrastructure) that allow users to transparently search these data collections have emerged as a natural decentralised solution. In this paper, we introduce iClusterDL, a novel P2P architecture supporting self-organising SONs that demonstrates (i) the feasibility of supporting rich query models without resorting to structured overlays, and (ii) the benefits derived from a looser organisation of system components. iClusterDL relies on the idea of organising peers into SONs, where clusters are linked by virtue of containing similar information. Building upon a P2P model, iClusterDL consists of three general types of peers, namely information providers (contributing documents to the network), information consumers (seeking for existing information), and super-peers. Super-peers act as access points for clients and providers. They self-organise into a SON to offer a robust, fault-tolerant and scalable means for routing messages and managing queries. The description of each super-peer is derived from the descriptions of the providers connected to it, allowing super-peers to organise into clusters of similar content. In iClusterDL, both publications and queries are processed using the vector space model (VSM).

iClusterDL is designed to support both *information retrieval* (IR) and *information filtering* (IF) functionality. In an IR scenario a user poses a *one-time query* and the system returns all resources matching the query (e.g., all currently available documents relevant to the query). In an IF scenario a user submits a *continuous query* and waits to be notified about certain future events of interest (i.e., about newly published documents relevant to the continuous query). With today's information explosion, IF becomes a necessary component for DLs since it provides the tools to keep the users informed, while not imposing the effort and cognitive overload of periodic one-time queries.

As an example of an application scenario for iClusterDL let us consider a computer scientist, whose main field of expertise is *information retrieval*, and is mainly interested in retrieving scientific publications on his topic of interest and also follow the work of prominent researchers in the area. He regularly uses the DL of his department and also other DLs to search for new papers in the area. Even though searching for interesting papers this week turned up nothing, a search next week may return new information. Clearly, a system that is able to integrate a big number of relative sources and also capture his long-term information need would be a valuable tool that would allow him to save both time and effort.

In our example scenario, consider a university comprised of geographically distributed campuses that maintain their local DLs. In the context of iClusterDL, each campus implements and maintains its own super-peer, which serves as an access point for the provider representing the campus DL, and the clients deployed by end-users (e.g., students or faculty). Other super-peers may also be deployed by larger institutions, like research centers or content providers (e.g., CiteSeer, ACM, Springer), to provide access points for their employees and make the contents of their DLs available in a timely way. iClusterDL offers an infrastructure, based on concepts of P2P systems, for organising the super-peers in a scalable, efficient and self-organising architecture. This architecture (i) allows for seamless integration of information sources, since different DLs and other content providers offer the same querying interface through the iClusterDL system to the end-user, (ii) enhances fault tolerance, since it avoids centralised components that introduce bottlenecks and single points of failure, and (iii) requires no central administration authority, since each participant is responsible for the administration and maintenance of its own (super-) peers.

To the best of our knowledge, iClusterDL is the first approach towards efficient organisation of DLs in SONs that supports both IR and IF functionality. The proposed architecture is automatic (requires no intervention and minimal administration), general (requires no previous knowledge of the DL contents and works for any type of data model or content), adaptive (adjusts to changes of DL contents), efficient (offers fast query processing) and accurate (achieves high recall).

In the following, we position our paper with respect to related work in Sec. 2. The iClusterDL architecture and protocols for supporting IR and IF functionality are presented in Sec. 3 and Sec. 4 respectively. A critical evaluation of the performance of iClusterDL is presented in Sec. 5, followed by issues for further research in Sec. 6.

2 Related Work and Background

This section provides a brief survey of the technology related to data organisation and retrieval in SONs, and related research in the context of IR and IF for the DL domain.

2.1 Semantic Overlay Networks

Initial IR approaches implementing SON-like structures to support content search in a distributed collection of peers include the work of Klampanos et al. [4], where an architecture for IR-based clustering of peers is proposed. In this architecture, a representative peer (hub) maintains information about all other hubs and is responsible for query routing. The notion of peer clustering based on similar interests rather than similar documents is introduced in the work of Spripanidkulchai et al. [5]. In a similar spirit, Parreira et al. [6] introduce the notion of "peer-to-peer dating" that allows peers to decide which connections to create and which to avoid based on various usefulness estimators. Additional work on peer organisation using SONs is based on the idea of "small-world networks". Schmitz [7] assumes that peers share concepts from a common ontology, and this information is used for organising peers into communities with similar concepts. Along the same lines, Li et al. [8] propose creating a self-organising semantic small world (SSW) network based on the semantics of data objects stored locally to peers. Other works include the embedding of metric spaces in the SON paradigm, as in [9, 10]. None of the works cited above examines the applicability of SONs in the DL domain, while their focus lies on single-tier architectures.

To the best of our knowledge, DESENT [11] is the only work applying the concept of SONs in a DL domain. In DESENT [11], SONs are organised as a hierarchy of clusters. Each cluster is represented by the so called "cluster gateway" (a single peer within the cluster). In turn, groups of clusters form super-clusters with their own gateways. Each cluster gateway maintains information about all other clusters (and super-cluster) representatives. All communication within a SON is propagated through gateways, thus creating communication bottlenecks at these peers. Contrary, iClusterDL maintains a flat structure of clusters with no representatives and the communication can be routed through any peer within a cluster, thus avoiding the bottleneck problem. Moreover, iClusterDL supports IF functionality (in addition to IR) at no extra communication cost.

2.2 IR and IF in Digital Libraries

Two-tier architectures is a natural solution for addressing architectural issues in DL domains. Lu and Callan [12] study the problem of content-based retrieval in distributed DLs focusing on resource selection and document retrieval. They propose a two-level hierarchical P2P network where DLs are clients that connect to super-peers, which form an unstructured P2P network in the second level of the hierarchy. A recent contribution by the same authors [13] suggests organising super-peers into neighborhoods to devise a method for super-peer selection and ranking. Finally, OverCite [14] is proposed as a distributed alternative for the scientific DL CiteSeer[3], by utilising a DHT infrastructure to harness distributed resources (storage, computational power, etc.).

To the best of our knowledge, P2P-DIET [15] and LibraRing [16] are the first approaches that try to support both IR and IF functionality in a single unifying framework. P2P-DIET utilises an expressive query language based on IR concepts and is implemented as an *unstructured P2P network* with routing techniques based on shortest paths and minimum weight spanning trees. LibraRing [16] provides protocols to support both IR and IF functionality in DLs using DHTs. The DHT is used to make sure that queries meet the matching documents (in the IR scenario) or that published documents meet the indexed continuous queries (in the IF scenario). Contrary to LibraRing, MinervaDL [17] suggests using Chord DHT to disseminate and store *statistics* about the document providers rather than the documents themselves. iClusterDL is the first comprehensive architecture that exploits SONs to provide a framework for self-organised and self-managed DLs that can offer a rich quiver of tools to end-users. The work presented in this paper extends the iCluster protocols [3] to support both IR and IF functionality in the DL domain, by focusing on a two-tier architecture.

3 iClusterDL Architecture

Figure 1 illustrates an overview of the iClusterDL architecture, composed of three main components: *super-peers*, *providers* and *clients*.

Super-peers represent the message routing layer of the network. Super-peers are nodes with more capabilities than provider or client peers (e.g., more cpu power and bandwidth), are responsible for serving information producers (providers) and information consumers (clients), and act as their access points to iClusterDL network. Super-peers run the *peer organisation* protocol and form clusters based on their likelihood to contain similar content. Each super-peer is characterised by the information content of the providers that use it as an access point, and maintains a *routing index* holding information for short- and long-range links to other super-peers: *short-range links* correspond to *intra-cluster* information (i.e., links to super-peers with similar interests), and *long-range links* correspond to *inter-cluster* information (i.e., links to super-peers having different interests). In the iClusterDL architecture, the role of super-peers is multi-fold: they act as the glue between information producers and information consumers, build a distributed self-organising repository that can be queried efficiently, store continuous queries submitted by information consumers to match them against newly published documents, and serve as a robust, fault-tolerant and scalable routing infrastructure.

Provider peers stand for information sources exposing their contents to the clients of iClusterDL. A provider connects to the network through a super-peer (its access point). Providers store locally their documents and answer one-time

[3] http://citeseer.ist.psu.edu/

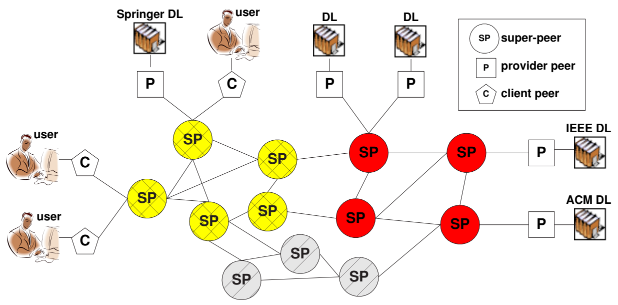


Fig. 1. A high-level view of the iClusterDL architecture

queries. Each document is represented by a vector of terms in the spirit of VSM, which may be either automatically (by text analysis) or manually assigned to each document (e.g., tags or index terms). To identify its *interests*, a provider categorises its documents (each document may belong in multiple categories) using an external reference system (e.g., the ACM categorisation system), an ontology, or unsupervised clustering methods [18]. Since documents are represented by term vectors, naturally a provider's interest is also represented by the *centroid* (i.e., the mean vector of the vector representations of the documents it contains). The interests of a provider are then used to determine the interests of the super-peer that acts as an access point for the provider.

Client peers provide an interface to end-users searching or subscribing for data. Clients can pose one-time queries and receive answers, or subscribe to resource publications and receive notifications about published resources that match their interests. A client connects to the network through a single super-peer (its access point). Clients can connect, disconnect or even leave the system at any time. If clients are not on-line, notifications matching their interests are stored by their access points and delivered once clients reconnect.

4 The iClusterDL Protocols

The main idea behind iClusterDL is to let super-peers self-organise into SONs, and then, address (or monitor) the most promising cluster of super-peers with respect to an one-time (or continuous) query. In this section, we discuss the protocols that specify how peers join and leave the network, how super-peers self-organise into clusters, and how query processing for both one-time or continuous query, and document publication are carried out.

4.1 Provider and Client Join

The first time a provider p_i wants to connect to the iClusterDL network, it has to follow the join protocol. Initially, p_i categorises its documents and stores its interests in To join the network, p_i finds the IP address of a super-peer s_j using out-of-band means (e.g., via a secure web site that contains IP addresses for the super-peers that are currently online), and sends to s_j a NewP= $(id(p_i), ip(p_i), int(p_i))$ message, where $id(p_i)$ is the identifier of p_i created when the provider bootstraps and is used to support *dynamic IP addressing*, and $ip(p_i)$ is the IP address of p_i. Subsequently, s_j adds p_i in its *provider table* (PT_j), which is a table used for identifying the providers that use s_j as their access point. PT_j is used to associate the $id()$ of a provider with its last known IP address, and also stores information such as the status of p_i (connected/disconnected). Finally, s_j sends to p_i an acknowledgement message AckP= $(id(s_j), ip(s_j))$. Once p_i has joined, it can use the connect/disconnect protocol described next to connect to and disconnect from the network.

Clients use a similar protocol to join the iClusterDL network. When a client c_i wants to connect for the first time, it sends a NewC= $(id(c_i), ip(c_i))$ message to its access point s_j, and is subsequently added to the *client table* (CT_j) of s_j. Similarly, CT_j is used to store contact and status information for c_i, along with non-delivered notifications (e.g., due to c_i being offline).

4.2 Provider and Client Connect/Disconnect

A provider p_i connects to the network by addressing a ConnectP= $(id(p_i), ip(p_i))$ message to its access point s_j. If $id(p_i)$ exists in PT_j then p_i is marked as connected, otherwise this means that s_j is not the access point of p_i, and that p_i wants to *migrate* to another super-peer. In this case, the connection request is rejected and the provider p_i has to run the join protocol described in the previous section. When a provider p_i wants to disconnect, it sends to its access point s_j a DisconnectP= $(id(p_i), ip(p_i))$ message and s_j marks p_i as disconnected in its PT_j. Clients connect or disconnect from the network in a similar way, but s_j has also to make sure that a disconnected client c_i will not miss notifications about resources of interest while not online. Thus, notifications for c_i are stored in the client table CT_j of s_j and wait to be delivered upon reconnection of c_i.

4.3 Super-Peer Join

To join the iClusterDL network, a super-peer s_i must compute the list of its interests $int(s_i)$ as $\bigcup_{\forall p_j \in PT_i} int(p_j)$ in order to identify itself in the network. For each distinct interest int_{ik} in $int(s_i)$, s_i maintains an individual routing index RI_{ik}, which contains short-range links that point to super-peers with similar interests, and long-range links that point to super-peers with different interests. During the joining procedure of s_i, the routing index is initialised as follows: s_i collects in RI_{ik} the IP addresses of λ randomly selected super-peers or routing index entries of neighboring super-peers. These links will be refined according to the interest int_{ik} of s_i, using the super-peer organisation protocol described in the next section.

4.4 Super-Peer Organisation

Super-peer organisation proceeds by establishing new connections and by discarding old ones, producing this way clusters of super-peers with similar interests. Each super-peer s_i *periodically* initiates a *rewiring procedure*. For each interest int_{ik}, s_i computes the intra-cluster similarity NS_{ik} (as a measure of cluster cohesion) as:

$$NS_{ik} = \frac{1}{|RI_{ik}|} \cdot \sum_{\forall s_j \in RI_{ik}} \mathsf{Sim}_{ij}^{int_{ik}}, \qquad (1)$$

Procedure Rewiring($s_i, int_{ik}, t_F, \theta, m$)
A procedure initiated by a super-peer s_i whenever its neighborhood similarity NS_{ik} drops below a predefined threshold θ.

input: super-peer s_i with interest int_{ik} and routing index RI_{ik}
output: updated routing indexes

```
1:  compute
    $NS_{ik} = \frac{1}{|RI_{ik}|} \cdot \sum_{\forall s_j \in RI_{ik}} \mathsf{Sim}_{ij}^{int_{ik}}$
2:  **if** $NS_{ik} < \theta$ **then**
3:    $P \leftarrow \{\ \}$
4:    create FindPeers() message
5:    send FindPeers() to
      $m$ random neighbors of $s_i$
6:    let $s_j$ be a neighbor of $s_i$ receiving
      FindPeers() and $int_{j\kappa}$ the interest
      of $s_j$ that is most similar to $int_{ik}$
7:    update $R_{j\kappa}$ with information from $P$
8:    $P \leftarrow P :: \langle (id(s_j), ip(s_j), int_{j\kappa}) \rangle$
9:    $t_F \leftarrow t_F - 1$
10:  **do** the same for the neighbors of $s_j$
11: **repeat** until $t_F = 0$
12: return list $P$ to $s_i$
13: update $R_{ik}$ with information from $P$
```

(a) The self-organisation protocol

Procedure CQuery_Routing(cq, s_i, t_q, θ, m)
A super-peer s_i compares the continuous query cq towards its interests, decides whether to store it in its local continuous query data structures and forwards cq to the super-peer network.

input: query q issued by super-peer s_i and threshold θ
output: updated continuous query data structures

```
1:  compare $cq$ against interests $int_{in}$,
    where $1 \leq n \leq l$, and select $int_{ik}$
    that is the interest of $s_i$ most
    similar to $cq$
2:  initiate message CQUERY()
3:  **if** $sim(cq, int_{ik}) \geq \theta$ **then**
4:    store $cq$ in local data structures
5:    forward CQUERY() to all short-range
      links in $RI_{ik}$
7:  **else**
      forward CQUERY() to $m$ neighbors
      of $s_i$ that are the most similar to $cq$
8:  $t_{cq} \leftarrow t_{cq} - 1$
9:  **do** the same for the neighbors of $s_i$
10: **repeat** until $t_{cq} = 0$
```

(b) The continuous query routing protocol

Fig. 2. Pseudocode for the super-peer protocols

where $|RI_{ik}|$ is the number of short-range links (super-peers in the neighborhood of s_i) with respect to interest int_{ik}. If NS_{ik} is greater than a threshold θ, then s_i does not need to take any further action, since it is surrounded by similar super-peers. Otherwise, s_i initiates a cluster refinement process by issuing FindPeers$= (id(s_i), ip(s_i), int_{ik}, P, t_F)$ message, where P is a list initially empty and t_F is the time-to-live (TTL) of the message. Notice that both θ and t_F are system parameters that are tuned upon system bootstrapping. A super-peer s_j receiving the message computes the similarity between its interest int_{jy} with interest int_{ik} in FindPeers() message, appends to P the interest resulted in the maximum similarity value, reduces t_F by 1 and forwards FindPeers() message to its neighboring super-peers. When $t_F = 0$, the FindPeers() message is sent back to the message initiator s_i. During the message forwarding procedure, a message recipient s_j chooses to forward FindPeers() message to a set of m randomly chosen super-peers contained in s_j's routing index. To further clarify the rewiring procedure, Fig. 2(a) presents the pseudocode for super-peer organisation.

A super-peer s_j receiving FindPeers() message may also collect information about new super-peers with similar interests by examining the interests of previous message recipients. This new information may then be used to update the routing index $RI_{j\kappa}$ of s_j by replacing old short-range links corresponding to super-peers with less similar interests with new links corresponding to super-peers with more similar interests.

4.5 IR: Processing One-Time Queries

Let us assume that a client c_i wants to submit a query q, where q is a term vector. Initially, c_i sends a SubmitQ= $(id(c_i), ip(c_i), q)$ message to its access point s_j. Upon receival of a SubmitQ() message, s_j compares q against its interests $int_{jn}, 1 \leq n \leq \ell$, and selects the interest int_{jk} for which $int_{jk} = max(sim(q, int_{jn}))$ holds (i.e., the most similar interest to the query). If $sim(q, int_{jk}) \geq \theta$, then s_j creates a Query= $(id(s_j), ip(s_j), id(c_i), ip(c_i), q, t_q)$ message, where t_q is the query TTL, and forwards it to all its short-range links in RI_{jk}, thus *broadcasting* the message to its neighborhood. This broadcasting happens because s_j has identified that the query is close to its interests and thus, also close to the interests of its neighbors.

If $sim(q, int_{jk}) < \theta$, s_j forwards the Query() message to the m super-peers with interests most similar to q. In this case m is usually small, and this query forwarding technique is referred as *fixed forwarding*, since the query is forwarded through a limited number of paths until it finds a similar cluster. All forwarding super-peers execute the aforementioned protocol and reduce t_q by one at each step of the forwarding procedure. Notice that the value of t_q in the case of broadcasting is not the same with that of fixed forwarding; typically t_q is smaller when broadcasting (since we need to reach super-peers only a few hops away) and larger when performing a fixed forwarding (since we need to explore regions of the network that are possibly far away from the initiating super-peer).

Apart from query forwarding, each super-peer s_k, for which $sim(q, s_k) \geq \theta$ holds, applies the following procedure for retrieving documents similar to q. It constructs a FindRes= $(id(s_j), ip(s_j), q)$ message and sends it to all providers with interests similar to q, by examining its provider table PT_k. Once a provider peer p_l receives a FindRes() message, it matches q against its local document collection to retrieve all documents matching q. The provider peer ranks the local results according to their relevance to the query, creates a result list r_l of the form $\langle d, m(d), Sim(q, d)\rangle$, where d is a pointer to a document, $m(d)$ are metadata about d and $Sim(q, d)$ is the similarity between q and d, and sends a RetRes= $(id(p_l), ip(p_l), r_l)$ message to s_j (notice that s_j is the super-peer that initiated the querying procedure on behalf of client c_i). In this way, s_j collects the local result lists of the relevant providers contacted and uses them to compute a final result list R that is sent to the client peer c_i and presented to the user.

4.6 IF: Processing Continuous Queries

In the following, the protocols of Sect. 4.5 above are adjusted appropriately to support IF functionality.

Subscribing with a Continuous Query: Let us assume that a client c_i wants to submit a continuous query cq. Initially, c_i sends a SubmitCQ= $(id(c_i), ip(c_i), cq)$ message to its access point s_j, and s_j initiates a CQuery= $(id(s_j), ip(s_j), id(c_i), ip(c_i), cq, t_q)$ message. The message is forwarded in the super-peer network following the mechanism described in Sect. 4.5. A super-peer s_k that receives a continuous query cq similar to its interests (i.e., $sim(cq, s_k) \geq \theta$), stores cq in its local continuous query data structures to match it against future publications. Super-peer s_k will utilise these data structures at publication time to find quickly all continuous queries that match a publication. This can be done using an appropriate local filtering algorithm such as SQI [19]. Figure 2(b) presents the pseudocode for continuous query routing.

Publishing a New Document: Publications in iClusterDL are kept locally at each provider in the spirit of [17]. This lack of publication dissemination mechanism is a design decision that avoids document-granularity dissemination (e.g., as in [17]) and offers increased scalability by trading recall. Thus, only the corresponding super-peers (i.e., those indexing a continuous query cq) can notify a client c_i, although provider peers using *other* super-peers as access points may also publish relevant documents. When a provider peer p_j wants to publish a new document d to the network, it sends the document to its access point s_k and then, the super-peer s_k is responsible for matching d against its local continuous query database to decide which continuous queries match d and thus, which clients should be notified.

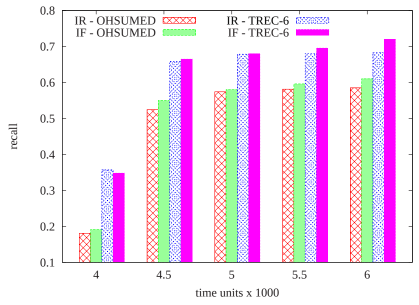
At pre-specified intervals or when the document collection of a provider peer p_j has changed significantly, p_j recomputes its interests and informs its corresponding access point by sending a RefreshInt= $(id(p_j), ip(p_j), int(p_j))$ message. Subsequently, the super-peer that acts as the access point of p_j changes the respective record in its PT and refreshes the list of p_j's interests.

Notification Delivery: Let us assume that super-peer s_i has to deliver a notification for a continuous query cq to client c_j. It creates a Notify= $(id(p_l), ip(p_l), d, m(d), cq)$ message, where d is a pointer to the document matching cq, $m(d)$ are metadata about d and p_l is the provider that published d, and sends it to c_j. If c_j is not online, then s_i sends the message to the access point s_k of c_j, using $ip(s_k)$ associated with cq. Super-peer s_k is then responsible for storing the message in CT_k and delivering it to c_j upon reconnection.

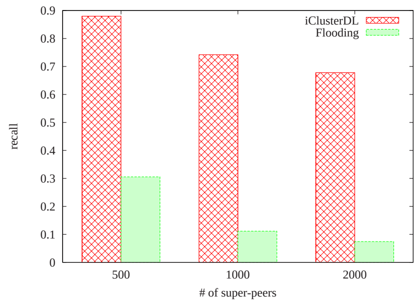
4.7 Discussion

iClusterDL is highly dynamic as it allows for random insertions or deletions of all peer types, as well as for insertions or deletions of new documents. All peer insertions/deletions are performed in a local manner (affecting only entries in local super-peer data structures in the case of provider or client peers, and super-peer routing indexes in the case of super-peers), and the network self-organises to a new stable state after a few iterations of the rewiring protocol. In this way, iClusterDL is based solely on local interactions, requiring no previous knowledge of the network structure or of the overall content in the network. The messaging cost to maintain the network clustered is dampened at query time by the fast and efficient search mechanism. To further improve the search mechanism, iClusterDL maintains a fixed number of long-range links (i.e., links to other clusters) in the routing indexes of the super-peers. These links provide shortcuts to other clusters and prevent them from forming disconnected communities that are inaccessible by others.

Notice that methods assuming one interest per super-peer [20, 7] (specialisation assumption) will not perform well under a DL setting: the description of a super-peer would either reflect the contents of its strongest interest (i.e., the interest of the provider with the largest document collection) ignoring all other interests, or result in a single category representing the averaging over the document collections of the super-peer's providers. This would result in poor retrieval performance as queries (even very specific ones) would be addressing highly incoherent clusters of super-peers. To avoid this, in iClusterDL providers and super-peers use multiple interests obtained by document categorisation.



(a) Recall over time for both IR and IF scenarios using two real-life corpora



(b) Recall for iClusterDL and Flooding using the same number of messages

Fig. 3. Evaluation of retrieval and filtering effectiveness

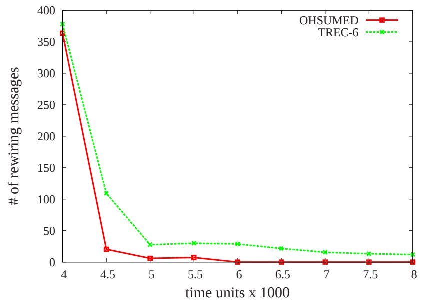
5 Evaluation

In this section, we evaluate the proposed protocols using two real-life corpora with medical and web data, and compare them against a baseline flooding approach. The first corpus is a subset of the OHSUMED TREC[4] document collection that contains more than 30,000 medical articles categorised in 10 different categories. The second dataset contains over 556,000 documents from the TREC-6[5] collection, categorised in 100 categories. Notice that the second dataset has been previously used by [21] to evaluate IR algorithms over distributed document collections for scenarios similar to the ones of the DL domain. In our setting, each provider is mainly specialised in one category, while we pose no restrictions on which providers connect to a super-peer. Thus, a super-peer may have providers with different interests, and be subsequently part of many different clusters. The one-time and continuous queries that were employed are strong representatives of document categories. The size of the network is 2,000 super-peers.

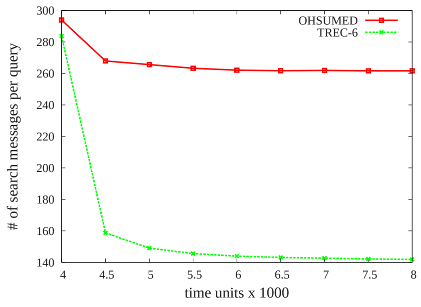
Each super-peer periodically tries to find better neighbors by initiating the rewiring procedure. The base unit for time used is the period t. The start of the rewiring procedure for each super-peer is randomly chosen from the time interval $[0, 4K \cdot t]$ and its periodicity is randomly selected from a normal distribution of $2K \cdot t$. We start recording the network activity at time $4K \cdot t$, when all super-peers have initiated at least once the rewiring procedure. The *network traffic* is measured by recording the number of messages exchanged by the super-peers during rewiring or querying. Finally, the IR (respectively IF) effectiveness is evaluated using *recall* as the percentage of qualifying answers retrieved with respect to the total number of qualifying answers in the network (respectively the percentage of total number of notifications received with respect to the total number of published documents matching a subscription in a time window).

We experimented with different values of similarity threshold θ, message forwarding TTL t_F and query forwarding TTL t_q. We consider that a given parameter value is better than another if it results in better clustering and retrieval for less communication load. The baseline parameter values used for this set of experiments are $\theta = 0.9$, $t_F = 4$ and $t_q = 8$.

[4] http://trec.nist.gov/data/t9_filtering.html
[5] http://boston.lti.cs.cmu.edu/callan/Data/



(a) Organisation messages



(b) Search messages per query

Fig. 4. Message costs for different corpora

Retrieval and filtering effectiveness. Figure 3(a) illustrates the performance of iClusterDL as a function of time for both datasets. Due to the similar query routing protocol, recall for both IR and IF scenarios has similar behavior. When the super-peer network is not yet fully organised into clusters of similar super-peers (i.e., moment $4K$) the queries cannot be routed towards the appropriate super-peers, thus reaching low recall values (around 20% for the OHSUMED corpus). When the network becomes organised into cohesive clusters (i.e., moment $6K$), iClusterDL achieves high values of recall (over 60%) for all examined scenarios. Figure 3(b) illustrates a comparison of the performance of iClusterDL against the flooding approach limited to the same number of messages, for different sizes of the network using the TREC-6 corpus (the results are similar for the OHSUMED dataset). iClusterDL demonstrates significant performance improvement over flooding, which increases with network size reaching up to 8 times better recall. In fact, iClusterDL scales up well for large networks (only 20% decrease of recall for 400% increase in network size).

Message costs. Figure 4(a) shows the number of messages needed over time for the self-organisation of the super-peers. Initially, the network presents a message overhead in terms of organisation messages, which is greatly reduced after the network organises into coherent clusters. The organisation messages for TREC-6 are higher due to the higher number of clusters created, which in turn is an effect of the higher number of document categories. However, after the organisation of the super-peers, the rewiring protocol is able to maintain an effective super-peer organisation at a small communication cost. Figure 4(b) shows the number of messages per (continuous) query over time. When the network is not yet organised into coherent neighborhoods (left-most points in the x-axis), iClusterDL needs high number of search messages to achieve the recall shown in Fig. 3(a). However, this message overhead is decreased (over 12% decrease for OHSUMED and 100% decrease for TREC-6) as the super-peers get organised into clusters with similar interests (right-most points in the x-axis). The search messages for OHSUMED are higher due to the higher number of peers per cluster.

Figures 4(b) and 3(a) demonstrate that iClusterDL manages to effectively organise the network, as IR and IF performance improves for much less communication overhead.

6 Outlook

We are currently investigating the effect of different system parameters on the clustering and retrieval performance of iClusterDL: the size of the routing index, the number of short/long-range links, and the clustering quality measures.

Acknowledgements

This work was supported in part by EU project Aeolus. Paraskevi Raftopoulou was partially funded by project "Herakleitos" of the Greek Secretariat for Research and Technology.

References

1. Crespo, A., Garcia-Molina, H.: Routing Indices for Peer-to-Peer Systems. In: ICDCS. (2002)
2. Loser, A., Wolpers, M., Siberski, W., Nejdl, W.: Semantic Overlay Clusters within Super-Peer Networks. In: DBISP2P. (2003)
3. Raftopoulou, P., Petrakis, E.: iCluster: a Self-Organising Overlay Network for P2P Information Retrieval. In: ECIR. (2008)
4. Klampanos, I., Jose, J.: An Architecture for Information Retrieval over Semi-Collaborating Peer-to-Peer Networks. In: SAC. (2004)
5. Spripanidkulchai, K., Maggs, B., Zhang, H.: Efficient Content Location using Interest-Based Locality in Peer-to-Peer Systems. In: INFOCOM. (2003)
6. Parreira, J.X., Michel, S., Weikum, G.: p2pDating: Real Life Inspired Semantic Overlay Networks for Web Search. Information Processing and Management **43**(1) (2007)
7. Schmitz, C.: Self-Organization of a Small World by Topic. In: P2PKM. (2004)
8. Li, M., Lee, W.C., Sivasubramaniam, A.: Semantic Small World: An Overlay Network for Peer-to-Peer Search. In: ICNP. (2004)
9. Linari, A., Patella, M.: Metric Overlay Networks: Processing Similarity Queries in p2p Databases. In: DBISP2P. (2007)
10. Sedmidubsky, J., Barton, S., Dohnal, V., Zezula, P.: Adaptive Approximate Similarity Searching Through Metric Social Networks. In: ICDE. (2008)
11. Doulkeridis, C., Noervaag, K., Vazirgiannis, M.: Scalable Semantic Overlay Generation for P2P-based Digital Libraries. In: ECDL. (2006)
12. Lu, J., Callan, J.: Content-based Retrieval in Hybrid Peer-to-Peer Networks. In: CIKM. (2003)
13. Lu, J., Callan, J.: Federated Search of Text-Based Digital Libraries in Hierarchical Peer-to-Peer Networks. In: ECIR. (2005)
14. Stribling, J., Councill, I., Li, J., Kaashoek, M., Karger, D., Morris, R., Shenker, S.: Overcite: A Cooperative Digital Research Library. In: IPTPS. (2005)
15. Idreos, S., Koubarakis, M., Tryfonopoulos, C.: P2P-DIET: An Extensible P2P Service that Unifies Ad-hoc and Continuous Querying in Super-Peer Networks. In: SIGMOD. (2004)
16. Tryfonopoulos, C., Idreos, S., Koubarakis, M.: LibraRing: An Architecture for Distributed Digital Libraries Based on DHTs. In: ECDL. (2005)
17. Zimmer, C., Tryfonopoulos, C., Weikum, G.: MinervaDL: An Architecture for Information Retrieval and Filtering in Distributed Digital Libraries. In: ECDL. (2007)
18. Steinbach, M., Karypis, G., Kumar, V.: A Comparison of Document Clustering Techniques. In: KDD. (2000)
19. Yan, T., Garcia-Molina, H.: The SIFT Information Dissemination System. In: TODS. (1999)
20. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmer, M., Risch, T.: EDUTELLA: a P2P Networking Infrastructure based on RDF. In: WWW. (2002)
21. Xu, J., Croft, W.: Cluster-Based Language Models for Distributed Retrieval. In: SIGIR. (1999)