MinervaDL: An Architecture for Information Retrieval and Filtering in Distributed Digital Libraries*

Christian Zimmer, Christos Tryfonopoulos, and Gerhard Weikum

Department for Databases and Information Systems
Max-Planck-Institute for Informatics, 66123 Saarbrücken, Germany
{czimmer, trifon, weikum}@mpi-inf.mpg.de

Abstract. We present MinervaDL, a digital library architecture that supports approximate information retrieval and filtering functionality under a single unifying framework. The architecture of MinervaDL is based on the peer-to-peer search engine Minerva, and is able to handle huge amounts of data provided by digital libraries in a distributed and self-organizing way. The two-tier architecture and the use of the distributed hash table as the routing substrate provides an infrastructure for creating large networks of digital libraries with minimal administration costs. We discuss the main components of this architecture, present the protocols that regulate node interactions, and experimentally evaluate our approach.

1 Introduction

In this paper we present MinervaDL, a digital library (DL) architecture that supports *approximate* information retrieval and filtering functionality in a single unifying framework. Our architecture is hierarchical like the ones in [24,12,21,26] and utilizes a Distributed Hash Table (DHT) to achieve scalability, fault-tolerance, and robustness in its routing layer. The MinervaDL architecture allows handling huge amounts of data provided by DLs in a distributed and self-organizing way, and provides an infrastructure for creating large networks of digital libraries with minimal administration costs.

There are two kinds of basic functionality that we expect our architecture to offer: *information retrieval* (also known as *one-time querying*) and *information filtering* (also known as *publish/subscribe* or *continuous querying* or *selective dissemination of information*). In an information retrieval (IR) scenario a user poses an one-time query and the system returns all resources matching the query (e.g., all currently available documents relevant to the query). In an information filtering (IF) scenario, a user submits a continuous query (or *subscription* or *profile*) and will later be notified from the system about certain events of interest that take place (i.e., about newly published documents relevant to the continuous query).

Our DL architecture is built upon the Minerva P2P search engine [3,2] and contains three main components: *super-peers*, *providers* and *consumers*. Providers are implemented by information sources (e.g., digital libraries) that want to expose their content to the rest of the MinervaDL network, while consumers are utilized by users to query for

* This work has been partly supported by the DELOS Network of Excellence and the EU Integrated Project AEOLUS.

L. Kovács, N. Fuhr, and C. Meghini (Eds.): ECDL 2007, LNCS 4675, pp. 148–160, 2007.

and subscribe to new content. Super-peers utilize the Chord DHT [19] to create a conceptually global, but physically distributed directory that manages aggregated statistical information about each provider's local knowledge in compact form. This distributed directory allows information consumers to collect statistics about information sources and rank them according to the probability to answer a specific information need. This reduces network costs and enhances scalability since only the most relevant information sources are queried. In MinervaDL, both publications and (one-time and continuous) queries are interpreted using the vector space model (VSM), but other appropriate data models and languages could also be used (e.g., LSI or language models).

As an example of an application scenario for MinervaDL let us consider John, a professor in computer science, that is interested in constraint programming and wants to follow the work of prominent researchers in the area. He regularly uses the digital library of his department and a handful of other digital libraries to search for new papers in the area. Even though searching for interesting papers this week turned up nothing, a search next week may turn up new information. Clearly, John would benefit from accessing a system that is able to not only provide a search functionality that integrates a big number of sources (e.g., organizational digital libraries or even libraries from big publishing houses), but also capture his long term information need (e.g., in the spirit of [24,16,28]). This system would be a valuable tool, beyond anything supported in current digital library systems, that would allow John to save time and effort. In our example scenario, the university John works in is comprised of three geographically distributed campuses (Literature, Sciences and Biology) and each campus has it own local digital library. In the context of MinervaDL, each campus would maintain its own super-peer, which provides an access point for the provider representing the campus' digital library, and the clients deployed by users such as John. Other super-peers may also be deployed by larger institutions, like research centers or content providers (e.g., CiteSeer, ACM, Springer, Elsevier), to provide access points for their users (students, faculty or employees) and make the contents of their digital libraries available in a timely way. MinervaDL, proposed in this paper, offers an infrastructure, based on concepts of P2P systems, for organizing the super-peers in scalable, efficient and self-organizing architecture. This architecture allows seamless integration of information sources, enhances fault-tolerance, and requires minimum administration costs.

Contrary to approaches like LibraRing [24] that focus on *exact* retrieval and filtering functionality (e.g., by disseminating documents or continuous queries in the network), in MinervaDL publications are processed locally and query or subscribe to only selected information sources that are most likely to satisfy the user's information demand. In this way, efficiency and scalability are enhanced by trading faster response times for some loss in recall, achieving *approximate* retrieval and filtering functionality. MinervaDL is the first approach to provide a comprehensive architecture and the related protocols to support approximate retrieval and filtering functionality in a digital library context. In the following sections, we position our paper with respect to related work, and discuss the MinervaDL architecture and an related application scenario. Subsequently, we present the protocols that utilize node interactions and experimentally show the efficiency of our approach both in terms of retrieval effectiveness and message efficiency. Finally, in the last section we give directions for future work.

2 Related Work

In this section, we survey related work in the context of information retrieval and filtering in P2P networks. Initially we focus on retrieval approaches in super-peer networks and structured overlay networks as these are the two areas conceptually closer to our approach. Subsequently we discuss work in the area of P2P IF and position our work with respect to the approaches presented here.

IR in super-peer networks. In [12] the authors study the problem of content-based retrieval in distributed digital libraries focusing on resource selection and document retrieval. They propose to use a two-level hierarchical P2P network where digital libraries are clients that cluster around super-peers that form an unstructured P2P network in the second level of the hierarchy. In a more recent work, [13] uses also an unstructured P2P architecture to organize the super-peers and uses the concept of neighborhood to devise a method for super-peer selection and ranking. In a similar fashion, [11] proposes an architecture for IR-based clustering of peers in semi-collaborating overlay networks.

ODISSEA [21] was one of the first attempts to utilize a DHT in a super-peer environment, by focusing on architectural issues of building a P2P search engine. There, a two-tier architecture is again adopted, and the lower tier nodes of the system are implemented on top of Pastry DHT. Later, OverCite [20] was proposed as a distributed alternative for the scientific literature digital library CiteSeer. This functionality was made possible by utilizing a DHT infrastructure to harness distributed resources (storage, computational power, etc.).

IR in structured networks. With the advent of DHTs as a remedy for the node location problem that existed in unstructured networks, a significant number of approaches tried to support VSM on top of structured overlays. Meteorograph [9] was one of the early papers to deal with the problem of similarity search over structured P2P overlays. pSearch [23] was the first P2P system that used LSI to reduce the feature vectors of the documents. In pSearch the authors propose the usage of a multi-dimensional CAN to efficiently distribute document indices in the P2P network. In [18] a similar approach is proposed, and Chord DHT is used to index the documents and route the queries to appropriate peers. While most of related papers utilize a DHT to route the queries to appropriate peers, Minerva [3] follows a different approach. In Minerva the structured overlay offers a conceptually global, but physically distributed directory, that maintains IR-style *statistics* and *quality of service information*. This information is exploited by querying peers, and most relevant ones according to resource selection algorithms [2] are contacted. The research presented in this paper extends the Minerva approach with information filtering functionality and adapts the protocols to a DL environment.

IF in P2P networks. New approaches that use a DHT as the routing infrastructure to build filtering functionality have lately been developed. Scribe [17] is a topic-based publish/subscribe system based on Pastry. Hermes [16] is similar to Scribe because it uses the same underlying DHT (Pastry) but it allows more expressive subscriptions by supporting the notion of an event type with attributes. pFilter [22] uses a hierarchical extension of CAN DHT to filter unstructured documents and relies on multi-cast trees to notify subscribers. VSM and LSI can be used to match documents to user queries. Finally, supporting prefix and suffix queries in string attributes is the focus of the

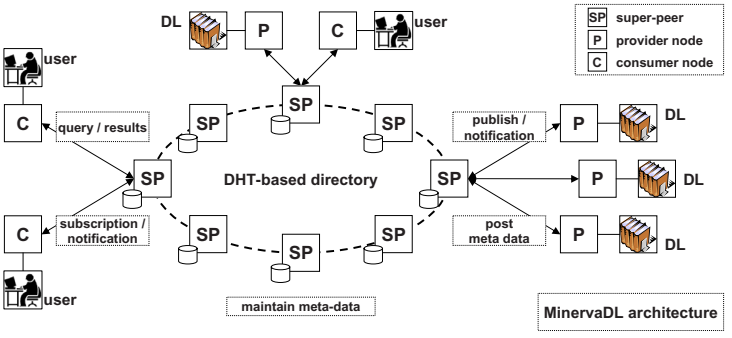


Fig. 1. A high level view of the MinervaDL architecture

DHT-Strings system [1], which utilizes a DHT-agnostic architecture to develop algorithms for efficient multi-dimensional event processing.

None of the works discussed above provides a comprehensive architecture and the related protocols to support both IR and IF in DLs using DHTs. P2P-DIET [10] and LibraRing where the first approaches that tried to support both functionalities in a single unifying framework. P2P-DIET utilizes 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. An extension of P2P-DIET [7] considers a similar problem for distributing RDF meta-data in an Edutella [15] fashion. LibraRing [24] was the first approach to provide protocols for the support of both IR and IF functionality in DLs using DHTs. In LibraRing, super-peers are organized in a Chord DHT and both (continuous) queries and documents are indexed by hashing words contained in them. This hashing scheme depends heavily on the data model and query language adopted, and the protocols have to be modified when the data model changes [24]. 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). In this way the retrieval effectiveness of a centralized system is achieved, while a number of routing optimizations (such as value proxying, content based-multicasting, etc.) are used to enhance scalability.

Contrary to the LibraRing approach, in MinervaDL the Chord DHT is used to disseminate and store *statistics* about the document providers rather than the documents themselves. Avoiding per-document indexing granularity allows us to improve scalability by trading recall for lower message traffic. This approximate retrieval and filtering approach relaxes the assumption of potentially delivering notifications from every producer that holds in all the works mentioned above and amplifies scalability. Additionally, it allows us to easily support different data models and query languages, without modifications to the protocols, since matching is performed locally in each node.

3 MinervaDL Architecture

Figure 1 shows a high level view of the MinervaDL architecture, composed of three types of nodes: *super-peers*, *consumer nodes*, and *provider nodes*.

Super-peers run the DHT protocol and form a distributed directory that maintains statistics about providers' local knowledge in compact form. The Chord DHT is used to partition the term space such that each directory node is responsible for the statistics of a randomized subset of terms. Directory nodes are super-peers, nodes with more capabilities than consumer or provider nodes (e.g., more cpu power and bandwidth) that are responsible for serving information consumers and providers and act as their access point to the MinervaDL network. When the number of super-peers is small, each node can easily locate others in a single hop by maintaining a full routing table. When the super-peer network grows in size, the DHT provides a scalable means of locating other nodes. Super-peers can be deployed by large institutions like universities, research centers or content providers (e.g., CiteSeer, ACM, Springer, Elsevier) to provide access points for their users (students, faculty or employees) or digital libraries.

Consumer nodes are utilized by users to connect to the MinervaDL network, using a single super-peer as their *access point*. Utilizing a consumer node allows users to pose one-time queries, receive relevant resources, subscribe to resource publications with continuous queries and receive notifications about published resources (e.g., documents) that match their interests. Consumer nodes are responsible for selecting the best information sources to query (resp. monitor) with respect to a given one-time query (resp. continuous query). If consumer nodes are not online to receive notifications about documents matching their submitted continuous queries, these notifications are stored by their access point and are delivered upon reconnection.

Provider nodes are implemented by information sources that want to expose their content to the MinervaDL network. Provider nodes use a directory node as their access point and utilize it to distribute statistics about their local resources to the network. Providers answer one-time queries and store continuous queries submitted by consumers to match them against new documents they publish. More than one provider nodes may be used to expose the contents of large digital libraries, and also an integration layer can be used to unify different types of DLs.

4 The MinervaDL Protocols

In this section, we explain in detail the way consumers, providers and super-peers join and leave the network, publish new documents, submit one-time or continuous queries and receive answers and notifications for their submitted queries. We use functions $key()$, $ip(n)$ and $id(n)$ to denote the key, the IP address and the identifier of node n respectively. $key(n)$ is created the first time a consumer or provider joins the MinervaDL network, and is used to support dynamic IP addressing and $id(n)$ is produced by the Chord hash function and is used to identify a super-peer within the Chord ring.

4.1 Provider and Consumer Join

The *first time* a provider P wants to connect to the MinervaDL network, it has to follow the join protocol. P hast to find the IP address of a super-peer S using out-of-band means (e.g., via a secure web site that contains IP addresses for the super-peers that are currently online). P sends to S a NEWPROV($key(P), ip(P)$) message and S adds P in

its *provider table* (PT), which is a hash table used for identifying the providers that use S as their *access point*. Here, $key(P)$ is used to index providers in PT, while each PT slot stores contact information about the provider, its status (connected/disconnected) and its stored notifications (see Section 4.7). Subsequently, S sends to P an acknowledgement message ACKNEWPROV($id(S), ip(S)$). Once P has joined, it can use the connect/disconnect protocol described next to connect to and disconnect from the network. Consumers use a similar protocol to join the MinervaDL network.

4.2 Provider and Consumer Connect/Disconnect

When a provider P wants to connect to the network, it sends to its access point S a CONNECTPROV($key(P), ip(P), id(S)$) message. If $key(P)$ exists in PT of S, P is marked as connected. If $key(P)$ does not exist in PT, this means that S was not the access point of P the last time that P connected (Section 4.8 discusses this case). When a provider P wants to disconnect, it sends to its access point S a DISCONNECTPROV($key(P), ip(P), id(S)$) message and S marks P as disconnected in its PT.

Consumers connect/disconnect from the network in a similar way, but S has also to make sure that a disconnecting consumer C will not miss notifications about resources of interest while not online. Thus, notifications for C are stored in the *consumer table* CT of S and wait to be delivered upon reconnection of C (see Section 4.7).

4.3 Maintaining the Directory

In MinervaDL, the super-peers utilize a Chord-like DHT [19] to build-up a distributed directory, while each provider P uses its access point to distribute per-term statistics about its local index to the directory using POST messages. At certain intervals (e.g., every k publications or time units) the provider has to update its statistics in the directory. We now describe the updating process done by a provider.

Let $T = t_1, t_2, \ldots, t_n$ denote the set of all terms included in the documents a provider P has published after the last directory update. For each term $t \in T$, the provider computes statistics: (i) the maximum term frequency of occurrence within P's document collection (tf_t^{max}); (ii) the number of documents in its document collection containing t (df_t); (iii) the size of P's document collection (cs). Using its IP address, P forwards the POST($key(P), ip(P), tf_t^{max}, df_t, cs, t$) message to the super-peer S that is P's access point to the directory. Next, S uses the Chord `lookup()` function to forward a modified POST message (including in addition S's IP address $ip(S)$ and identifier $id(S)$) to the super-peer node responsible for identifier $H(t)$ (i.e., this node is responsible for maintaining statistics for term t). This node S_t stores the received POST message in its local statistics table ST to be able to provide the term statistics to nodes requesting them.

4.4 Submitting an One-Time Query

In this section we show how to answer one-time vector space queries. Let us assume that a consumer C wants to submit a query q containing terms $t_1, t_2, \ldots, t_n$. The following

steps take place. In step one, C sends to its access point S a SUBMITQ($key(C), ip(C), q$) message. In the second step, for each term $t \in q$, S computes $H(t)$ to obtain the identifier of the super-peer responsible for storing statistics about term t. Then, it sends GETSTATS($key(C), ip(C), t$) message by using the Chord `lookup()` function.

In step three each super-peer S_t that receives a GETSTATS message, searches its local statistics table ST for term t to retrieve a list L of provider nodes storing documents containing the term t. Each element in list L is a tuple $(key(P), ip(P), tf_t^{max}, df_t, cs, t)$ containing contact information about providers and statistics about terms contained in documents that these providers publish. Subsequentlyt, S_t creates a RETSTATS($id(S_t), ip(S_t), L$) message and sends it to consumer C using $ip(C)$ included in the GETSTATS message. In this way, the consumer receives provider statistics for all query terms.

In step four, C uses the scoring function $sel(P, q)$ described in Section 5.1 to rank the providers with respect to q and identify the $top - k$ providers that hold documents satisfying q. Subsequently, C creates a GETRESULTS($ip(C), key(C), q$) message and forwards it, using the contact information associated with the statistics, to all provider nodes selected previously. Once a provider node P receives a GETRESULTS message containing a query q, it matches q against its local document collection to retrieve the documents matching q. The local results are ranked according to their relevance to the query to create a result list R. Subsequently, P creates a RETRESULTS($ip(P), R, q$) message and sends it to C. In this way, C collects the local result lists of all selected providers and uses them to compute a final result list that is then presented to the user. To merge the retrieved result lists, standard IR scoring functions (e.g., CORI [4] or GlOSS [8]) are used.

4.5 Subscribing with a Continuous Query

This section describes how to extend the protocols of Section 4.4 to provide information filtering functionality. To submit a continuous query $cq = \{t_1, t_2, \ldots, t_n\}$, the one-time query submission protocol needs to be *modified*. The first three steps are identical while step four is modified as follows.

C uses the scoring function $pred(P, cq)$ described in Section 5.2 to rank the providers with respect to cq and identify the $top-k$ providers that may publish documents matching *cq in the future*. These are the nodes that will store cq and C will receive notifications from these nodes only. This query indexing scheme makes provider selection a critical component of the filtering functionality. Notice that, in a filtering setting, resource selection techniques like $sel(P, cq)$ described in Section 5.1 and used for one-time querying, are not appropriate since we are not interested in the current document collection of the providers but rather in their future publishing behavior.

Once providers that will store cq have been determined, C creates an INDEXQUERY ($key(C), ip(C), id(S), ip(S), cq$) message and sends it to these providers using the IP addresses associated with the GETSTATS messages C received in the previous step. When a provider P receives an INDEXQUERY message, it stores cq in its local continuous query data structures to match it against future publications. P utilizes these data structures at publication time to find quickly all continuous queries that match a publication. This can be done using e.g., algorithms BestFitTrie [25] or SQI [27].

4.6 Publishing a New Document

Contrary to approaches such as [24] that distribute the documents or the index lists among the nodes to achieve exact retrieval and filtering functionality, publications in MinervaDL are kept locally at each provider. This lack of publication forwarding mechanism is a design decision that offers increased scalability in MinervaDL by trading recall. Thus, only monitored provider nodes (i.e., indexing a continuous query cq) can notify a consumer C, although other providers may also publish relevant documents. As already stated, this makes the scoring function $pred()$ a critical component.

Thus, when a provider node P wants to publish a new document d to the MinervaDL network, the document is only matched against P's local continuous query database to determine which continuous queries match d, and thus which consumers should be notified. Additionally, at certain intervals P creates POST messages with updated statistics and sends them to its access point S.

4.7 Notification Delivery

Assume a provider P that has to deliver a notification for a continuous query cq to consumer C. It creates a NOTIFICATION($ip(P), key(P), d, cq$) message, where d is the document matching cq, and sends it to C. If C is not online at that time, then P sends the message to S, where S is the access point of C, using $ip(S)$ associated with cq. S then is responsible for storing the message and delivering it to C upon reconnection. If S is also off-line then the message is sent to S', which is the access point of P, and S' utilizes the DHT to locate the $successor(id(S))$ (as defined in Chord [19]), by calling function `lookup()`. Answers to one-time queries are handled in a similar way.

4.8 Super-Peer Join/Leave

To join MinervaDL network, a super-peer S must find the IP address of another super-peer S' using out-of-band means. S creates a NEWSPEER($id(S), ip(S)$) message and sends it to S' which performs a lookup operation by calling `lookup(id(S))` to find $S_{succ} = successor(id(S))$, similarly to the Chord joining procedure. S' sends a ACK-NEWSPEER($id(S_{succ}), ip(S_{succ})$) message to S and S updates its successor to S_{succ}. S also contacts S_{succ} asking its predecessor and the data that should now be stored at S. S_{succ} updates its predecessor to S, and answers back with the contact information of its previous predecessor, S_{pred}, and all continuous queries and publications that were indexed under key k, with $id(S) \leq k < id(S_{pred})$. S makes S_{pred} its predecessor and populates its index structures with the new data that arrived. After that S populates its finger table entries by repeatedly performing lookup operations on the desired keys.

When a super-peer S wants to leave LibraRing network, it constructs a DISCONNECTSPEER($id(S), ip(S), id(S_{pred}), ip(S_{pred}), data$) message, where $data$ are all the continuous queries, published resources and stored notifications of off-line nodes that S was responsible for. Subsequently, S sends the message to its successor S_{succ} and notifies S_{pred} that its successor is now S_{succ}. Clients that used S as their access point connect to the network through another super-peer S'. Stored notifications can be retrieved through $successor(id(S))$.

5 Scoring Functions

Selecting the appropriate provider nodes to forward an one-time or a continuous query, requires a ranking function that will be used to determine the most appropriate sources for a given information demand. Although, both IR and IF protocols utilize the same meta-data stored in the distributed directory, the node ranking strategies differ significantly and have to consider varying objectives: in the case of IR, the scoring algorithm has to identify authorities with respect to a given query, i.e., nodes that have already made available a lot of relevant document *in the past*. These nodes should receive high scores, and thus be ranked high in the respective node ranking that will result. For this purpose, standard resource selection algorithms known from the IR literature can be utilized. Section 5.1 discusses our scoring function for one-time querying.

In contrast, in the IF case the scoring function has to determine the most appropriate provider nodes that given a continuous query cq, will publish documents matching cq *in the future*. In this setting, relying on existing resource selection algorithms similar to the ones utilized for one-time querying leads to low recall, as these algorithms are able to capture past publishing behavior, and cannot adapt to the dynamics of the filtering case. To better illustrate this consider the following example.

Assume two providers, where the first is specialized in soccer (i.e., in the past has published a lot of documents about soccer), although now it is rarely publishing new documents. The second provider is not specialized in soccer but currently it is publishing many documents about soccer. Now, a consumer subscribes for documents with the continuous query *soccer Euro 2008*. A ranking function based on resource selection algorithms would always choose the first provider. To get a higher ranking score, and thus get selected for indexing the query, the second provider has to specialize in soccer, a long procedure that is inapplicable in a filtering setting, which is by definition dynamic. The above example illustrates that our ranking formula should be able to predict the publishing behavior of providers by observing both their past and current behavior and projecting it to the future. To achieve this, we rely on statistical analysis tools to model a provider's publishing behavior. In Section 5.2 we discuss a novel technique that is based on time-series analysis of IR statistics.

5.1 Resource Selection

A number of resource selection methods (e.g., tf/idf based, CORI [5], GlOSS [8] and others) are available to identify authorities with respect to a query q. We use a tf/idf based approach to compute a score for provider P using the formula shown below:

$$sel(P, q) = \sum_{t \in q} \beta \cdot \log\left(df_{P,t}\right) + (1 - \beta) \cdot \log\left(tf_{P,t}^{max}\right)$$

The parameter β can be chosen between 0 and 1 and is used to stress the importance of df or tf^{max} (experiments have shown that $\beta = 0.5$ is an appropriate value in most cases [2]). Using the scoring function presented above we can identify providers that store high-quality documents for query q and thus achieve high recall by querying only a few providers. To further improve recall we have developed overlap-aware node selection strategies [2] and techniques that exploit term correlations [14].

5.2 Publishing Behavior Prediction

Our prediction mechanism collects IR statistics from the distributed directory and treats them as time-series data to perform statistical analysis over them. Statistical analysis assumes that the data points taken over time have some sort of internal structure (e.g., trend etc.), and uses this observation to analyze older values and predict future ones [6]. There exist various approaches that differ in their assumptions about the internal structure of the time-series (e.g., whether it exhibits a trend or seasonality). *Moving average techniques* are a well-known group of time-series prediction techniques that assign equal weights to past observations (e.g., averaging is the simplest form of moving average techniques), and thus cannot cope with trends or seasonality. In our setting, it is reasonable to put more emphasis on a node's recent behavior and thus assign higher weights to recent observations. *Double exponential smoothing* assigns exponentially decreasing weights to past observations and assumes that the time-series data present some trend to predict future values. Since many queries are expected to be short-lived so that no seasonality will be observed in the IR statistics time-series, we do not consider seasonality in our predictions (and thus we do not use *triple exponential smoothing*).

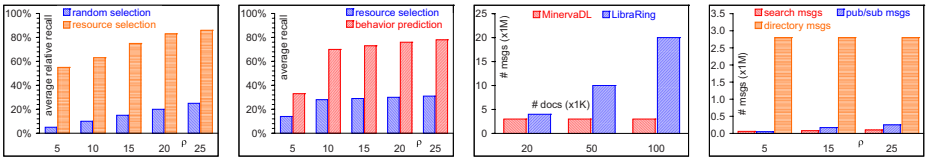
The scoring function $pred(P, cq)$ returns for a score representing the probability that provider P will publish in the future documents relevant to the continuous query cq. In MinervaDL, we use double exponential smoothing to predict the following two statistical values. Initially, for all terms t in cq, we predict the value for $df_{P,t}$ (denoted as $df^*_{P,t}$), and use the difference (denoted as $\delta(df^*_{P,t})$) between $df^*_{P,t}$ and the last received value from the directory to calculate the score for P. $\delta(df^*_{P,t})$ reflects the number of relevant documents concerning t that P will publish in the next period. Secondly, we predict $\delta(cs^*)$ as the difference in the collection size of P reflecting the provider node's overall expected future publishing activity. In this way we model two aspects of the node's behavior; its ability to publish relevant documents in the future and its overall expected publishing activity. The consumer node that submits cq obtains the IR statistics that are needed as an input to our prediction mechanism by utilizing the distributed directory. The following formula is used to compute the prediction score for a provider P with respect to a continuous query cq:

$$pred(P, cq) = \sum_{t \in cq} \log\left(\delta(df^*_{P,t}) + \log\left(\delta(cs^*_P) + 1\right) + 1\right)$$

In the above formula, publication of relevant documents is accented compared to publishing rate. If a node publishes no documents at all, or, to be exact, the prediction of $\hat{cs}^*$, and thus the prediction of $\hat{df}^*$, is 0 then the $pred(P, q)$ value is also 0. The addition of 1 in the $\log$ formulas is used to yield positive predictions and to avoid $\log(0)$.

6 Experiments

In our experimental evaluation we assume a total number of 1000 providers and the same number of consumers, which are connected to a MinervaDL network using 400 super-peers as access points. At bootstrapping, each provider stores 300 documents and is mainly specialized in one out of ten categories (e.g., *Music*, *Sports*, *Arts* etc.) such



(a) IR scenario (b) IF scenario (c) approx. vs exact IF (d) message costs

Fig. 2. Retrieval effectiveness and message costs for MinervaDL

that we have 100 specialized information sources per category. We construct 30 queries[1] containing two, three or four terms that are strong representatives of a certain document category and are used both as one-time and continuous queries. Figure 2 shows the retrieval effectiveness of MinervaDL in terms of recall and its efficiency compared to exact matching approaches.

IR scenario. In figure 2(a), the experimental results of the information retrieval functionality of MinervaDL are presented. We assume that a consumer requests the 30 one-time queries and we are interested in the relative recall to a centralized search engine hosting a joint collection: the ratio of top-25 documents included in the merged P2P result. Thus, we apply the selection strategy as described before and increase the percentage ρ of providers in the system that are requested to answer the query. Figure 2(a) shows that the retrieval performance of MinervaDL manages to retrieve more than 90% of the top rated documents by asking only a fraction of the providers, whereas the baseline approach of random node selection reaches a much lower recall (around 20%).

IF scenario. To evaluate the IF functionality of MinervaDL, we assume that providers publish 30 documents within a specific time period (called publishing round) and we investigate the results after ten publishing rounds. We consider a scenario that models periods of inactivity in the publishing behavior and assume that consumers subscribe with 30 continuous queries and reposition them in each publishing round. In the experiments, we vary the percentage of monitored providers ρ and rank them using the two scoring functions approaches described in Section 5. As a retrieval measurement, we use recall as the ratio of total number of notifications received by the consumers to the total number of published documents matching a subscription. As illustrated in Figure 2(b), the use of behavior prediction improves recall over resource selection as it manages to model more accurately the publishing behavior of providers. In this way, our proposed scoring function manages to achieve a recall of around 80% by monitoring only 20% of the providers in the network. Notice that for resource selection, this number is only 35% which makes it an inapplicable solution to a filtering environment.

Message Costs Analysis. Figures 2(c) and 2(d) show different aspects of our message costs analysis. Here, we assume that in each round, the subscriptions are repositioned and the one-time queries are requested again (e.g., by another consumer). In this setting, we have three types of messages: directory, retrieval, and filtering messages. Figure 2(c)

[1] Example queries are *museum modern art*, or *space model*.

compares MinervaDL with an implementation of the exact matching protocols of LibraRing [24]. We consider the overall message costs as a function of the total number of documents published in the system for both approaches. As shown, message costs of MinervaDL are independent of the number of publications, whereas LibraRing, and all exact matching approaches, is *sensitive* to this parameter since documents have to be disseminated to the network. This imposes a high network cost especially for high publication rates expected by nodes exposing the contents of DLs.

Finally, in Figure 2(c) we can observe that directory messages dominate both retrieval and filtering messages. This is expected since matching and filtering mechanisms are by design local to accommodate high publication rates. This means that building both IR and IF functionality on top of the routing infrastructure imposes no extra cost on the architecture, compared to one that supports only one type of functionality. Since other types of information can also be stored in the directory in the same fashion (e.g., QoS statistics or load balancing information) building extra functionality will increase the added value of the directory maintenance, and come at almost no extra cost.

7 Outlook

We are currently implementing MinervaDL and plan to deploy it over a wide-area testbed such as PlanetLab. We are also focusing on the IF case and investigate the automatic adaptation of prediction parameters and the usage of different prediction techniques.

References

1. Aekaterinidis, I., Triantafillou, P.: Internet Scale String Attribute Publish/Subscribe Data networks. In: CIKM (2005)
2. Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Improving Collection Selection with Overlap-Awareness. In: SIGIR (2005)
3. Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Minerva: Collaborative P2P Search (Demo). In: VLDB (2005)
4. Callan, J.: Distributed Information Retrieval. Kluwer Academic Publishers, Dordrecht (2000)
5. Callan, J.P., Lu, Z., Croft, W.B.: Searching Distributed Collections with Inference Networks. In: SIGIR (1995)
6. Chatfield, C.: The Analysis of Time Series - An Introduction. CRC Press, Boca Raton (2004)
7. Chirita, P.-A., Idreos, S., Koubarakis, M., Nejdl, W.: Publish/Subscribe for RDF-based P2P Networks. In: ESWC (2004)
8. Gravano, L., Garcia-Molina, H., Tomasic, A.: GlOSS: Text-Source Discovery over the Internet. In: ACM TODS (1999)
9. Hsiao, H.-C., King, C.-T.: Similarity Discovery in Structured P2P Overlays. In: ICPP (2003)
10. 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)
11. Klampanos, I., Jose, J.: An Architecture for Peer-to-Peer Information Retrieval. In: SIGIR (2003)
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: Losada, D.E., Fernández-Luna, J.M. (eds.) ECIR 2005. LNCS, vol. 3408, Springer, Heidelberg (2005)

14. Michel, S., Zimmer, C., et al.: Discovering and Exploiting Keyword and Attribute-value Co-occurrences to Improve P2P Routing Indices. In: CIKM (2006)
15. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., Risch, T.: Edutella: A P2P Networking Infrastructure based on RDF. In: WWW (2002)
16. Pietzuch, P., Bacon, J.: Hermes: A Distributed event-based Middleware Architecture. In: DEBS (2002)
17. Rowstron, A., Kermarrec, A.-M., Castro, M., Druschel, P.: Scribe: The Design of a Large-scale Event Notification Infrastructure. In: COST264 (2001)
18. Sahin, O., Emekci, F., Agrawal, D., Abbadi, A.: Content-based Similarity Search over Peer-to-Peer Systems. In: Ng, W.S., Ooi, B.-C., Ouksel, A.M., Sartori, C. (eds.) DBISP2P 2004. LNCS, vol. 3367, Springer, Heidelberg (2005)
19. Stoica, I., et al.: Chord: a Scalable Peer-to-Peer Lookup Protocol for Internet Applications. In: ACM TON (2003)
20. Stribling, J., Councill, I., Li, J., Kaashoek, M., Karger, D., Morris, R., Shenker, S.: Overcite: A Cooperative Digital Research Library. In: Castro, M., van Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640, Springer, Heidelberg (2005)
21. Suel, T., et al.: Odissea: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. In: WebDB (2003)
22. Tang, C., Xu, Z.: pfilter: Global Information Filtering and Dissemination Using Structured Overlays. In: FTDCS (2003)
23. Tang, C., Xu, Z., Dwarkadas, S.: Peer-to-Peer Information Retrieval Using self-organizing Semantic Overlay Networks. In: SIGCOMM (2003)
24. Tryfonopoulos, C., Idreos, S., Koubarakis, M.: Libraring: An Architecture for Distributed Digital Libraries based on DHTs. In: Rauber, A., Christodoulakis, S., Tjoa, A.M. (eds.) ECDL 2005. LNCS, vol. 3652, Springer, Heidelberg (2005)
25. Tryfonopoulos, C., Koubarakis, M., Drougas, Y.: Filtering Algorithms for Information Retrieval Models with Named Attributes and Proximity Operators. In: SIGIR (2004)
26. Tryfonopoulos, C., Zimmer, C., Koubarakis, M., Weikum, G.: Architectural Alternatives for Information Filtering in Structured Overlay Networks. IEEE Internet Computing (2007)
27. Yan, T.W., Garcia-Molina, H.: The SIFT Information Dissemination System. In: ACM TODS (1999)
28. Yang, B., Jeh, G.: Retroactive Answering of Search Queries. In: WWW (2006)