Comparing different architectures for query routing in peer-to-peer networks

Henrik Nottelmann and Norbert Fuhr

Department of Informatics, University of Duisburg-Essen,
47048 Duisburg, Germany, {nottelmann,fuhr}@uni-duisburg.de

Abstract. Efficient and effective routing of content-based queries is an emerging problem in peer-to-peer networks, and can be seen as an extension of the traditional "resource selection" problem. Although some approaches have been proposed, finding the best architecture (defined by the network topology, the underlying selection method, and its integration into peer-to-peer networks) is still an open problem. This paper investigates different building blocks of such architectures, among them the decision-theoretic framework, CORI, hierarchical networks, distributed hash tables and HyperCubes. The evaluation on a large testbed shows that the decision-theoretic framework can be applied effectively and cost-efficiently onto peer-to-peer networks.

1 Introduction

Peer-to-peer (P2P) networks have emerged recently as an alternative to centralised architectures. The major problem in such networks is query routing, i.e. deciding to which other peers the query has to be sent for high efficiency and effectiveness. In contrast to the traditional resource selection problem, this process is inherently decentralised in peer-to-peer networks and based on local knowledge.

The decision-theoretic framework (DTF) [10, 6] computes an optimum selection based on cost estimations (including retrieval quality, time or money). Most other (centralised) resource selection methods follow a heuristic approach and compute a ranking of the digital libraries (DLs), e.g. CORI [4] or the language modelling approach [15]. The latter has been extended towards hierarchical peer-to-peer networks [8, 9], where a hub also stores language models ("hub descriptions") of the hubs directly connected to it. A fixed number of hubs is selected, while unsupervised learning is employed for learning a threshold for the number of selected DL peers. A modified version of the semi-supervised learning algorithm [14] is used for result merging.

This paper presents an extensive discussion of resource selection architectures for peer-to-peer networks. The architectures are classified based on the underlying resource selection approach (in our case, DTF and CORI as a baseline), design choices like the locality of knowledge (e.g. IDF values) and selections (centralised vs. decentralised), as well as the network topology (hierarchical networks with DLs and hubs, distributed hash tables and HyperCubes).

This paper is structured as follows: Section 2 briefly describes DTF and CORI. Then, different topologies are introduced, and the resource selection frameworks are extended towards P2P networks in section 4. The proposed approaches for resource selection in P2P networks have been evaluated in section 5.

2 Resource selection approaches

This section only sketches the decision-theoretic framework (DTF) and CORI.

2.1 The decision-theoretic framework (DTF)

In contrast to other resource selection approaches (e.g. CORI, see section 2.2) which only consider the similarity of the DL to a query, the decision-theoretic framework (DTF) [10, 6] estimates retrieval "costs" from different sources (e.g. monetary costs, computation and communication time, retrieval quality). As the actual costs are unknown in advance, expected costs (for digital library DL_i when s_i documents are retrieved for query q) are regarded instead:

$$EC_i(s_i,q) := E[r_i(s_i,q)] \cdot C^+ + [s_i - E[r_i(s_i,q)]] \cdot C^- + C^t \cdot EC^t(s_i) + C^m \cdot EC^m(s_i) \ , \quad (1)$$

where $E[r_i(s_i,q)]$ is the expected number of relevant documents among the s_i top-ranked documents and $s_i - E[r_i(s_i,q)]$ is the expected number of non-relevant documents, $EC^t(s_i)$ denotes the expected "time" costs, and $EC^m(s_i)$ the expected monetary costs. In addition, C^+, C^-, C^t and C^m are user-specific parameters which allow a user to specify her own selection policy, e.g. cheap and fast results. Non-relevant documents have higher costs (wasted time) than relevant ones, thus $C^+ < C^-$.

A user also specifies the total number n of documents to be retrieved out of m libraries, and the task is to compute an optimum solution (employing the algorithm presented in [6]):

$$s := \underset{\sum_{i=1}^{m} s_i = n}{\operatorname{argmin}} \sum_{i=1}^{m} EC_i(s_i,q). \quad (2)$$

Relevance costs are computed in two steps:

1. First, the expected number $E(\text{rel}|q,DL)$ of relevant documents in DL is computed based on statistical aggregations (called "resource description") of the DL.
2. Then, a linearly decreasing approximation of the recall-precision function is used for computing the expected number $E[r_i(s_i,q)]$ of relevant retrieved documents.

For the first step, the resource descriptions store the DL size $|DL|$ and the average (expectation) $\mu_t = E[w(d,t)|d \in DL]$ of the indexing weights $w(d,t)$ (for document d and term t). For a query with term weights $a(q,t)$ (summing up to one) and a linear retrieval model, the expected number $E(\text{rel}|q,DL)$ of relevant documents in DL w. r. t. query q can be estimated as:

$$E(\text{rel}|q,DL) = \sum_{d \in DL} Pr(\text{rel}|q,d) \approx \sum_{d \in DL} \sum_{t \in q} a(q,t) \cdot w(d,t) = |DL| \cdot \sum_{t \in q} a(q,t) \cdot \mu_t \ , \quad (3)$$

where $Pr(\text{rel}|q,d)$ denotes the probability that document d is relevant.

In a second step, $E(\text{rel}|q,DL)$ is mapped onto the expected number $E[r_i(s_i,q)]$ of relevant retrieved documents employing a linearly decreasing recall-precision function:

$$\frac{E[r_i(s_i,q)]}{s} = precision(recall) := 1 - recall = 1 - \frac{E[r_i(s_i,q)]}{E(\text{rel}|q,DL)} \ . \quad (4)$$

For DTF, the libraries have to return the probabilities of relevance of the result documents, thus no further normalisation step is required.

2.2 CORI

CORI [4, 5] only considers retrieval quality, by ranking the DLs w. r. t. their similarity to the query. This scheme can efficiently be implemented using a traditional IR system, where documents are replaced by "meta-documents" which are created by concatenating all documents in the DL.

CORI uses a $df \cdot icf$ weighting scheme instead of $tf \cdot idf$, based on the number m of involved libraries, the document frequency df, the collection frequency cf (the number of libraries containing the term) and the collection length cl (the number of terms in the DL). Thus, the belief in a DL due to observing query term t (the "indexing weight" of term t in the "meta-document" DL) is determined by:

$$w(DL,t) := 0.4 + 0.6 \cdot \frac{df}{df + 50 + 150 \cdot \frac{cl}{avgcl}} \cdot \frac{\log(\frac{m+0.5}{cf})}{\log(m+1)} \; . \tag{5}$$

Similar to DTF, a linear retrieval function is used based on $w(DL,t)$. CORI then selects a fixed number of top-ranked libraries, and retrieves an equal number of documents from each selected DL.

CORI also covers the data fusion problem. The library scores C and the document scores $D := Pr(q|d)$ are normalised to $[0,1]$, and then combined for computing the final normalised document score D'':

$$C' := \frac{C - C_{min}}{C_{max} - C_{min}}, \qquad D' := \frac{D - D_{min}}{D_{max} - D_{min}}, \qquad D'' := \frac{1.0 \cdot D' + 0.4 \cdot C' \cdot D'}{1.4} \; , \tag{6}$$

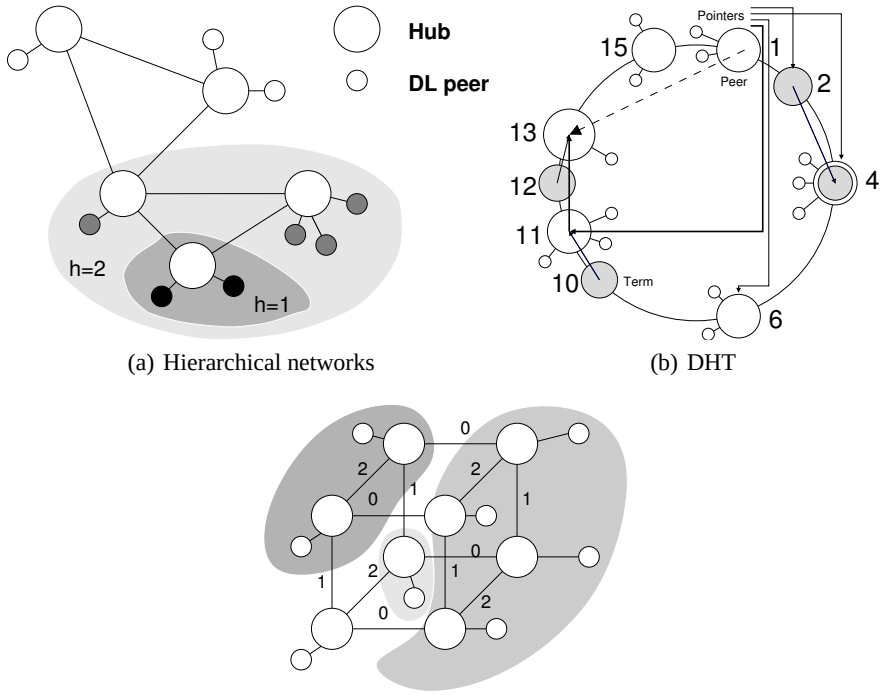
where C_{min} and C_{max} are the minimum and maximum DL scores for that query, and D_{min} and D_{max} the minimum and maximum document score in the DL.

3 Network topologies

This section introduces different topologies of the peer-to-peer network, which will be evaluated in section 5. In any topology, a direct neighbour of a peer is another peer if and only if there is a (direct) connection link. All other peers are called remote peers. The distance between two peers is the minimum number of hops (i.e., messages) required to go from one peer to the other.

3.1 Hierarchical networks

Most of the work reported in this paper—including the two other topologies—is based on a partition of peers into DL peers (sometimes also called "leaves") and hubs. DLs are typically end-user machines which answer but do not forward queries, while hubs are responsible for routing and, thus, high-end computers with high bandwidth which are nearly permanently online. Each DL peer is connected to at least one hub but not to other DL peers, which reduces the number of messages during query routing (i.e., resource selection). This results in a simple yet reasonable and efficient topology, called hierarchical peer-to-peer networks (see figure 1(a)).



(a) Hierarchical networks (b) DHT

(c) Partitions induced by HyperCubes

Fig. 1. Different peer-to-peer network topologies

3.2 Distributed hash tables

Distributed hash tables (DHTs) are similar to traditional hash tables, which provide fast insert and lookup functionality for keys based on hash functions. The DHT system Chord [16] (see figure 1(b)) maps peers and keys onto numbers using the same hash function. Peers are ordered in a ring according to the hash values, and each peer is responsible for storing the values for all keys mapped onto its hash value (and all values lower than the following peer). Each peer maintains a list of peers in exponentially increasing distance, which allows for efficient routing in $O(\log m)$ hops for m peers.

DHT have already been used for storing the complete index (of all documents) [7]. Here, the keys are terms, and the values are inverted lists (pairs comprised of document identifiers and the indexing weight). Thus, all documents form a single (but distributed) collection.

Similarly, DHTs can be employed for storing resource descriptions, Keys are terms, and the values are inverted lists (pairs of DL identifiers and e.g. average indexing weights). Once the inverted lists are fetched, a centralised selection can take place (see section 4.1). This contrasts other DHT approaches, where either the whole document

index is kept in the DHT [7], or DHTs are used for resource selection in unstructured networks (Minerva [1]).

3.3 HyperCubes

Hierarchical networks can contain cycles, which decrease efficiency as peers are contacted multiple times. In addition, estimating the quality of a hub connection is more difficult. HyperCube graphs (HyperCubes for short) [13] can be used to overcome this problem. A (binary) HyperCube is a regular d-dimensional structure, where each peer is connected to exactly d other peers (one per dimension, see figure 1(c)). Messages arriving via a connection on dimension $l \in \{0,1,\ldots,d-1\}$ can only be forwarded to peers on strictly higher dimensions $l' > l$. A consequence is that the dimensions define (starting from an arbitrary peer) a spanning tree on the network, which ensures that there is exactly one path from one peer to another peer. It also corresponds to a clearly defined partition of the whole network. Positions of missing peers are filled with "virtual peers" (see [13] for details), which are then replaced by "shortcuts" to all original peers which can be contacted through virtual peers only. As a consequence, peers can be connected to more or less than d neighbours.

4 Query routing in peer-to-peer networks

Query routing (i.e., resource selection) is a crucial task in peer-to-peer networks, as contacting all connected peers does not scale [11]. This section introduces several competing approaches for resource selection in peer-to-peer networks, which all can be used with DTF or CORI.

4.1 Centralised selection

A simple selection strategy is to use the P2P network for a "cost estimation collection phase", where the query is flooded in a Gnutella-like way through the hub network. Each hub computes cost estimations of its neighbour DLs, and sends them to the hub starting the routing process. Then, DTF or CORI are employed for a single central selection, and the selected DL peers are notified directly. As the hub starting the routing process can only compute a selection when all cost estimations have arrived, synchronous messages are employed. This centralised selection strategy yields a global optimum, but is rather inefficient (see [11] for Gnutella, and section 5.2 for hierarchical networks).

DHTs can significantly reduce the cost estimation collection phase in centralised selection. Instead of flooding the hub network, the resource descriptions of all DLs are extracted from the DHT with one lookup for each query term. In contrast to earlier approaches, we combine DHTs with hierarchical networks, thus the DHT ring is formed by the (fast and high-bandwidth) hubs only. The DHT contains resource descriptions in the form of inverted lists (DL identifiers plus average indexing weight), either for DLs (like in Minerva [1]) or for hubs. In the latter case, the inverted lists are much shorter; however, the quality of hub descriptions is lower than for DL descriptions, and

a second selection step would be required for selecting DLs without the DHT. Costs are estimated based on the returned inverted lists. Centralised selection in hierarchical networks involves each hub. In contrast, $O(l \cdot \log m)$ hops for l terms and m peers are required for the DHT, which is in most cases much more efficient. Effectiveness is not affected, as the same cost estimations are computed in both strategies.

An alternative solution is to leverage the hierarchical network topology in a two-step selection algorithm, where the DHT stores only descriptions of all hubs, and DLs are still connected to hubs (topology "dht-hubs"). A queried peer first computes a selection of hubs, and then calls the selected hubs which themselves compute a local selection, considering only the directly connected DL peers. The advantage is that the inverted lists are much smaller now and consume less bandwidth (remember that the full inverted lists have to be transferred to the queried hub).

A last approach is to exploit the spanning tree property of HyperCubes: As no cycles exist, each hub is contacted exactly once (not multiple times as for "cmu"), thus centralised selection with a HyperCube is more efficient (24 hops vs. 72 hops for the cost estimation collection phase in our case).

4.2 Decentralised selection

In contrast to the approaches mentioned in section 4.1, decentralised selection computes a local optimum selection on every hub receiving the query, by considering locally available descriptions of all DL and hubs in a predefined distance. Coordinating this decentralised routing approach is easier, as no responses have to be collected before a selection can be started; thus, asynchronous messages are sufficient. This decentralised selection method produces an overhead as a cost estimation and selection has to be performed on every hub. On the other hand, this method cuts down the number of hubs that are traversed, and thus saves time and bandwidth. Section 4.3 describes how hub descriptions can be obtained.

Decentralised selection also is the major application area for HyperCubes, as problems with cycles are avoided, in particular for "hc-1", where there is exactly one path from any hub to any DL. The selection should be improved by disjoint hub descriptions, where each DL is included in at most one hub description.

4.3 Computing hub descriptions

A hub description is a representative of the neighbourhood of a hub. Its statistical characteristics are defined by combining the resource descriptions of a set of DL peers.[1] Given the average term weights $\mu_{t,i}$ for each involved DL_i, the average term weights $\mu_{t,H}$ in the description of hub H can efficiently be computed as:

$$\mu_{t,H} = \frac{1}{\sum_j |DL_j|} \cdot \sum_i \sum_{d \in DL_i} w(d,t) = \frac{1}{\sum_j |DL_j|} \cdot \sum_i |DL_i| \cdot \mu_{t,i}. \tag{7}$$

[1] In peer-to-peer networks, co-operative peers can be assumed, so query-based sampling is not considered here; each DL returns its description upon request.

The horizon h defines the maximum distance between the hub H and the DLs to be considered for its hub description (see figure 1(a)). In the simplest case, only neighbour DLs are considered (horizon $h = 1$). With a horizon $h = 2$, also DLs directly connected to a neighbour hub of H have to be considered, and so on.

4.4 Locality of knowledge

Often retrieval and resource selection methods rely on some "global" statistics, e.g. inverse document frequencies (IDF). Early experiments have shown that DL-local IDFs (i.e., each DL computes its own values) are not accurate enough. Two alternatives are investigated, system-wide and hub-global IDF values. System-wide DL values are derived by combining all documents in all DL peers, and thus yield accurate estimates. In a large peer-to-peer network, they are too expensive to acquire, and hub-global IDFs (computed by combining all documents from all peers in a hub's neighbourhood) can be used as an approximation. A similar scheme is applied to CORI with its collection frequencies (CF values).

5 Evaluation

The methods proposed in the previous sections have been evaluated on a large test-bed. This sections describes the setup of the experiments and results in terms of efficiency and effectiveness.

5.1 Experimental Setup

The WT10g collection is used as a basis for our experiments. The topology "cmu" (taken from [8]) employes a hierarchical P2P network, where the WT10g collection is divided into 11,485 collections according to the document URLs; 2,500 collections (containing in total 1,421,088 documents) were chosen randomly, each of them forming one DL peer. Hubs are formed by similarity of the DL peers, each hub is connected to 13-1,013 DLs (379.8 on average). Neighbour hubs are selected randomly so that each hub has 1-7 hub neighbours (3.8 on average). The "dht-dl" and "dht-hubs" topology organise the hubs as a DHT ring (dismissing the "cmu" hub subnetwork). All DLs are kept in the hash table in "dht-dl", while "dht-hubs" stores hub descriptions only (which requires an additional local DL selection step).

The constructed HyperCube topologies "hc" and "hc-1" comprise of 25 hubs as in "cmu" (with dimension $d = 5$); each hub is connected to 4–14 other hubs (5.8 on average). For "hc", DLs are connected to these 25 hubs as in "cmu". As an alternative, each DL is connected to exactly one hub, randomly chosen out of the "cmu" connections[2] in the "hc-1" topology, so that each hub is connected to 3–254 DLs (100 on average). Thus, "hc-1" completely avoids cycles and yields disjoint hub neighbourhoods, in contrast to "cmu" and "hc".

[2] An alternative solution would be to choose the hub which is most similar to the DL, using a suitable similarity measure.

The WT10g collection only provides 100 topics with relevance judgements. For large P2P networks, more queries are required. Thus, we use 1,000 random queries (from the test set) generated from title fields of documents. Each query contains up to 6 terms, the average is 2.9 terms per query. Unless otherwise noted, $n = 50$ documents are requested. As these queries were created artificially, no relevance judgements are available. Pseudo-relevance judgements[3] were obtained by combining all 2,500 collections into one centralised collection (using system-wide IDF values), and marking the 50 top-ranked documents for each query as "relevant". Thus, the experiments measure how well distributed retrieval approximates the centralised collection. As the same linear retrieval function with the same indexing weights is used for the centralised collection, the DTF and the CORI experiments, these pseudo-relevance judgements form a fair baseline.

BM25[12] indexing weights are used for terms t in documents d:

$$w(d,t) := \frac{tf(t,d)}{tf(t,d) + 0.5 + 1.5 \cdot \frac{dl(d)}{avgdl}} \cdot \frac{\log \frac{|DL|}{df(t)}}{\log |DL|}, \tag{8}$$

with term frequency $tf(t,d)$, document length $dl(d)$ and its average *avgdl*, the library size $|DL|$ and document frequency $df(t)$.

Peer-to-peer networks form graphs, thus it is possible for a query to arrive at the same peer multiple times for the topologies "cmu" (DLs and hubs) and "hc" (only DLs). When this happens to a hub, it can only select DL peers which have not been selected before. A DL has to return additional documents when it is contacted multiple times. In earlier experiments, this approach has proven to be superior to other approaches like ignoring queries which reach a node for the second time, or preventing infinite loops by disallowing selections of a single hub.

All CORI variants select 5 peers (DLs and/or hubs) in each selection step, and retrieve an equal amount of documents for each selected peer. As the CORI-inherent result merging (section 2.2) performs poorly, documents are re-ranked w. r. t. probabilities of relevance, estimated in the same way as for DTF.

Only relevancy costs are investigated, as a good model for measuring time in peer-to-peer networks is missing yet; we defer this to future work.

5.2 Results

Due to space restrictions, we only show tables for hub-global IDF/CF values which can efficiently be computed in peer-to-peer networks, and report results on system-wide values in the text where necessary.

First we set $h = 1$, and compare centralised with decentralised resource selection (Table 1). DTF-d, the decentralised DTF variant, has lower effectiveness than DTF-c, the centralised counterpart with its global knowledge (5-10% worse). Surprisingly, CORI-d performs slightly better than CORI-c. The drawback of the centralised selection is its inefficiency; DTF-d needs about 40% less hops[4], CORI-d 22% less hops

[3] Queries and pseudo-relevance judgements are available at `http://www.is.informatik.uni-duisburg.de/projects/pepper/p2p-wt10g-testbed.zip`.

[4] Hops are the only efficiency measure investigated in this work.

	DTF, cent.	DTF, decent.	CORI, cent.	CORI, decent.
P@5	0.8032 / 0.0%	0.7058 / -12.1%	0.6760 / -15.8%	0.6902 / -14.1%
P@10	0.6630 / 0.0%	0.5721 / -13.7%	0.6144 / -7.3%	0.5663 / -14.6%
P@15	0.5562 / 0.0%	0.4896 / -12.0%	0.4672 / -16.0%	0.4705 / -15.4%
P@20	0.4793 / 0.0%	0.4295 / -10.4%	0.3693 / -23.0%	0.3859 / -19.5%
P@30	0.3792 / 0.0%	0.3456 / -8.9%	0.2520 / -33.5%	0.2705 / -28.7%
MAP	0.2586 / 0.0%	0.2391 / -7.5%	0.1558 / -39.8%	0.1660 / -35.8%
Precision	0.2706 / 0.0%	0.2515 / -7.1%	0.1839 / -32.0%	0.1859 / -31.3%
Recall	0.2582 / 0.0%	0.2417 / -6.4%	0.1562 / -39.5%	0.1665 / -35.5%
#Hops	103.3 / 0.0%	59.9 / -42.0%	77.8 / -24.7%	60.2 / -41.7%

Table 1. Centralised vs. decentralised selection, DTF vs. CORI, topology "cmu", $h = 1$

compared to their centralised counterparts, mainly due to the cost estimation collection phase. Table 1 also shows that CORI is outperformed by DTF in both variants: Set-based precision is 30% (26%), MAP and recall are 40% (30%) lower for centralised (decentralised, respectively) selection. CORI-c is more efficient compared to DTF-c, while there is virtually no difference for decentralised DTF and CORI.

DTF is used in the remainder for resource selection on a hub.

Two different research questions for decentralised selection—the influence of different hub description horizons, and the effect of different topologies, namely "cmu", "hc" and "hc-1"—are covered by table 2. Effectiveness and efficiency dramatically decreases for "cmu" when the horizon h increases. This counter-intuitive result is caused by the large hub neighbourhoods which contain half of the DLs in the network on average; thus it is more difficult to distinguish between good and bad hubs. For the two HyperCube topologies "hc" and "hc-1", however, effectiveness increases while efficiency decreases for larger hub horizons; as each hub (and for "hc-1", each DL) is contained in at most one neighbourhood, hubs can better identify good neighbours. The HyperCube topology "hc" outperforms "hc-1" for all investigated horizons, probably because the accuracy of the IDF values is higher (as more DLs are considered). For system-wide (high-accuracy) IDFs and $h = 3$, also smaller neighbourhoods are useful, thus results for "hc-1" are better than "hc". In addition, "hc" also performs better than "cmu" for $h \geq 2$; there is virtually no difference for $h = 1$.

Table 3 finally compares centralised resource selection in hierarchical networks ("cmu") and in distributed hash tables ("dht-dl" and "dht-hubs"). Effectiveness for "cmu" and "dht-dl" are the same, as the same DL descriptions and therefore the same cost estimations, are used. The different way of acquiring them in this "cost estimation collection" phase determines overall effectiveness: 72 hops are required for "cmu" (as each hub is reached multiple times due to cycles), compared with 6.7 hops (on average) for the DHT. When only hub descriptions are stored in the DHT (i.e., "dht-hubs") and 5 hubs/DLs are selected in each selection step, precision in the first 10 ranks decreases while set-based precision and recall as well as precision in lower ranks are improved. The drawback of this method, however, is the lower efficiency: The cost estimation collection phase requires the same number of hops as for "dtf-all", but about 15 hops to selected hubs have to be added, and the number of overall selected DLs is 25% higher.

(a) Topologies “cmu” and “hc”

	cmu, $h=1$	cmu, $h=2$	cmu, $h=3$	hc, $h=1$	hc, $h=2$	hc, $h=3$
P@5	0.7058 / 0.0%	0.6260 / -11.3%	0.5874 / -16.8%	0.6990 / -1.0%	0.7440 / 5.4%	0.7528 / 6.7%
P@10	0.5721 / 0.0%	0.4891 / -14.5%	0.4475 / -21.8%	0.5772 / 0.9%	0.6306 / 10.2%	0.6411 / 12.1%
P@15	0.4896 / 0.0%	0.4021 / -17.9%	0.3601 / -26.5%	0.4937 / 0.8%	0.5501 / 12.4%	0.5622 / 14.8%
P@20	0.4295 / 0.0%	0.3389 / -21.1%	0.2989 / -30.4%	0.4357 / 1.4%	0.4944 / 15.1%	0.5079 / 18.3%
P@30	0.3456 / 0.0%	0.2492 / -27.9%	0.2167 / -37.3%	0.3539 / 2.4%	0.4156 / 20.3%	0.4309 / 24.7%
MAP	0.2391 / 0.0%	0.1601 / -33.0%	0.1411 / -41.0%	0.2500 / 4.6%	0.3310 / 38.4%	0.3554 / 48.6%
Precision	0.2515 / 0.0%	0.1679 / -33.2%	0.1468 / -41.6%	0.2584 / 2.7%	0.3104 / 23.5%	0.3254 / 29.4%
Recall	0.2417 / 0.0%	0.1565 / -35.2%	0.1345 / -44.4%	0.2479 / 2.6%	0.3077 / 27.3%	0.3256 / 34.7%
#Hops	59.9 / 0.0%	89.9 / 50.2%	95.1 / 58.7%	49.2 / -17.9%	56.8 / -5.2%	57.1 / -4.6%

(b) Topologies “hc” and “hc-1”

	hc, $h=1$	hc, $h=2$	hc, $h=3$	hc-1, $h=1$	hc-1, $h=2$	hc-1, $h=3$
P@5	0.6990 / 0.0%	0.7440 / 6.4%	0.7528 / 7.7%	0.5916 / -15.4%	0.7120 / 1.9%	0.7574 / 8.4%
P@10	0.5772 / 0.0%	0.6306 / 9.3%	0.6411 / 11.1%	0.4488 / -22.2%	0.5743 / -0.5%	0.6157 / 6.7%
P@15	0.4937 / 0.0%	0.5501 / 11.4%	0.5622 / 13.9%	0.3585 / -27.4%	0.4831 / -2.1%	0.5277 / 6.9%
P@20	0.4357 / 0.0%	0.4944 / 13.5%	0.5079 / 16.6%	0.2965 / -31.9%	0.4150 / -4.8%	0.4633 / 6.3%
P@30	0.3539 / 0.0%	0.4156 / 17.4%	0.4309 / 21.8%	0.2182 / -38.3%	0.3239 / -8.5%	0.3750 / 6.0%
MAP	0.2500 / 0.0%	0.3310 / 32.4%	0.3554 / 42.2%	0.1334 / -46.6%	0.2099 / -16.0%	0.2543 / 1.7%
Precision	0.2584 / 0.0%	0.3104 / 20.2%	0.3254 / 26.0%	0.1579 / -38.9%	0.2310 / -10.6%	0.2673 / 3.5%
Recall	0.2479 / 0.0%	0.3077 / 24.1%	0.3256 / 31.3%	0.1409 / -43.2%	0.2197 / -11.4%	0.2614 / 5.4%
#Hops	49.2 / 0.0%	56.8 / 15.4%	57.1 / 16.2%	42.9 / -12.8%	50.6 / 2.9%	51.3 / 4.4%

Table 2. DTF, decentralised selection, Hierarchical networks vs. HyperCubes

This result for “dtf-hubs” cannot be replicated for system-wide IDFs: Here, effectiveness is 10-13% worse w. r. t. all measures.

Indexing and retrieval for the 1,000 queries requires transmitting about 120 million inverted list entry hops [5] for “cmu”, about 75 million entry hops for “dht-dl” (38% less), as the latter topology lacks any redundancy (which is an disadvantage if a hub and all inverted lists on it disappears) while each DL is connected to multiple hubs in “cmu”. In “dht-hubs”, 250 million hops are required. In the retrieval phase, “dht-hubs” is the best performing approach with about 127,000 entry hops, compared to 4,9 million entry hops for “dht-dl” and 8,6 million entry hops for “cmu”.

For $n = 100$ documents to be retrieved using “cmu”, set-based precision decreases (more irrelevant documents are retrieved, and at most 50 of the retrieved 100 documents can be relevant at all), while recall (25-30%) and precision in the top ranks (up to 30%) dramatically improves. For centralised selection, it is close to the optimum; for decentralised selection, it performs about as well as centralised selection for $n = 50$.

With full knowledge (i.e., actual number of relevant documents and system-wide knowledge), centralised DTF performs close to the optimum, with precision and recall >99% (even if only the relevant documents in the DL are known). For decentralised selection, precision drops to 72.8% and recall to 62.6%; the top ranking, however, is still close to the optimum (e.g. P@15 is still >0.9). Thus, further work should concentrate on improvements in estimating the number of relevant documents in the collections.

[5] When an entry is transmitted over multiple hops, each hop is counted individually.

	cmu, cent.	cmu, decent.	dht-dl	dht-hubs
P@5	0.8032 / 0.0%	0.7058 / -12.1%	0.8032 / 0.0%	0.7546 / -6.1%
P@10	0.6630 / 0.0%	0.5721 / -13.7%	0.6630 / 0.0%	0.6468 / -2.4%
P@15	0.5562 / 0.0%	0.4896 / -12.0%	0.5562 / 0.0%	0.5723 / 2.9%
P@20	0.4793 / 0.0%	0.4295 / -10.4%	0.4793 / 0.0%	0.5186 / 8.2%
P@30	0.3792 / 0.0%	0.3456 / -8.9%	0.3792 / 0.0%	0.4405 / 16.2%
MAP	0.2586 / 0.0%	0.2391 / -7.5%	0.2586 / 0.0%	0.3670 / 41.9%
Precision	0.2706 / 0.0%	0.2515 / -7.1%	0.2706 / 0.0%	0.3326 / 22.9%
Recall	0.2582 / 0.0%	0.2417 / -6.4%	0.2582 / 0.0%	0.3328 / 28.9%
#Hops	103.3 / 0.0%	59.9 / -42.0%	38.1 / -63.1%	60.6 / -41.4%

Table 3. Topologies “cmu”, “dht-dl” and “dht-hubs”, $h = 1$

Policy	Best method
High precision in top ranks (5-10)	“cmu”, centralised DTF; “dht-dl”
High precision in lower ranks (15-30)	“dht-hubs”
High MAP, set-based precision/recall, higher efficiency	“hc”, decentralised DTF, $h = 3$
High MAP, set-based precision/recall, lower efficiency	“dht-hubs”
High efficiency (lowest number of hops)	“dht-dl”

Table 4. Evaluation summary and suggestions

The evaluation has shown that no method outperforms all others for all measures (efficiency, precision in top ranks, MAP, set-based precision and recall). Table 4 summarises the most important results and suggests the best method for a given policy.

6 Conclusion and outlook

This paper investigates different architectures of peer-to-peer information retrieval systems. The architectures are based on the DTF or CORI resource selection approaches and use different strategies for extending them towards peer-to-peer networks (e.g. centralised vs. decentralised selection, system-wide or hub-global statistics). Five different topologies, namely hierarchical networks, HyperCubes (with two variants) and distributed hash tables (DHTs, also with two variants) are investigated.

The extensive evaluation showed that there is no clear winner. Centralised DTF selection in the “dht-dl” topology minimises the number of hops. Set-based precision and recall and precision in the lower ranks can be maximised by switching to HyperCubes (“hc” with $h = 3$, slightly better efficiency) and the DHTs storing only hub descriptions (“dht-hubs”, slightly better effectiveness). Thus, the final goals (high efficiency or effectiveness) designate the choice of the selection approach.

The major drawback of this approach is that hub descriptions are created by considering a hub as a single large library which is created by merging all documents of all DLs in a neighbourhood. This does not take the distance of the involved DLs into account (prohibiting cost estimations w. r. t. time), and ignores DLs in a larger distance.

Future work will concentrate on better descriptions for hubs and methods for approximating unknown distant DLs.

7 Acknowledgements

This work is supported by the DFG (grant BIB47 DOuv 02-01, PEPPER).

References

[1] M. Bender, S. Michel, C. Zimmer, and G. Weikum. Bookmark-driven query routing in peer-to-peer web search. In Callan et al. [3].

[2] J. Callan, G. Cormack, C. Clarke, D. Hawking, and A. Smeaton, editors. *Proceedings of the 26st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, 2003. ACM.

[3] J. Callan, N. Fuhr, and W. Nejdl, editors. *SIGIR Workshop on Peer-to-Peer Information Retrieval*, 2004.

[4] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In E. A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, New York, 1995. ACM. ISBN 0-89791-714-6.

[5] J. French, A. Powell, J. Callan, C. Viles, T. Emmitt, K. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval*, pages 238–245, New York, 1999. ACM.

[6] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, 1999.

[7] M. Harren, J. M. Hellerstein, R. Huebsch, B. T. L. Loo, S. Shenker, and I. Stoica. Complex queries in DHT-based peer-to-peer networks. In *Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[8] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In D. Kraft, O. Frieder, J. Hammer, S. Qureshi, and L. Seligman, editors, *Proceedings of the 12th International Conference on Information and Knowledge Management*, New York, 2003. ACM.

[9] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In Callan et al. [3].

[10] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In Callan et al. [2].

[11] J. Ritter. Why Gnutella can't scale. No, really., 2001. `http://www.darkridge.com/~jpr5/doc/gnutella.html`.

[12] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992.

[13] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. Digital libraries. In *1st Workshop on Agents and P2P Computing*, 2005.

[14] L. Si and J. Callan. A semi-supervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 24:457–49, 2003.

[15] L. Si, R. Jin, J. Callan, and P. Ogilvie. Language modeling framework for resource selection and results merging. In C. Nicholas, D. Grossman, K. Kalpakis, S. Qureshi, H. van Dissel, and L. Seligman, editors, *Proceedings of the 11th International Conference on Information and Knowledge Management*, New York, 2002. ACM.

[16] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, 2001.