Scalable Peer-to-Peer Web Retrieval with Highly Discriminative Keys [⋆]

Technical report – LSIR-REPORT-2006-009

Ivana Podnar, Martin Rajman, Toan Luu, Fabius Klemm, Karl Aberer
[ivana.podnar, martin.rajman, vinhtoan.luu, fabius.klemm, karl.aberer]@epfl.ch

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland

Abstract. The suitability of Peer-to-Peer (P2P) approaches for full-text web retrieval has recently been questioned because of the claimed unacceptable bandwidth consumption induced by retrieval from very large document collections.
In this contribution we present a novel indexing/retrieval model that achieves high performance, cost-efficient retrieval by indexing with *highly discriminative keys (HDKs)* stored in a distributed global index maintained in a structured P2P network. HDKs correspond to carefully selected terms and term sets appearing in small numbers of collection documents. We provide a theoretical analysis of the scalability of our retrieval model and report experimental results obtained with our HDK-based P2P retrieval engine. These results show that, despite increased indexing costs, the total traffic generated with the HDK approach is significantly smaller than the one obtained with distributed single-term indexing strategies. Furthermore, our experiments show that the retrieval performance obtained with a random set of real queries is comparable to the one of centralized, single-term solution using the best state-of-the-art BM25 relevance computation scheme. Finally, our scalability analysis demonstrates that the HDK approach can scale to large networks of peers indexing web-size document collections, thus opening the way towards viable, truly-decentralized web retrieval.

1 Introduction

Contrarily to traditional information retrieval (IR) systems that build upon centralized or clustered architectures, P2P retrieval engines theoretically offer the possibility to cope with web-scale document collections by distributing the indexing and querying load over large networks of collaborating peers. However, while P2P distribution results in smaller resource consumption at the level of individual peers, there is an ongoing debate about the overall scalability of P2P web search because of the claimed unacceptable bandwidth consumption induced by retrieval from very large document

[⋆] The work presented in this report was carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European FP 6 STREP project ALVIS (002068)

collections. In [1] for example, it is shown that a naïve use of structured or unstructured P2P networks for web retrieval leads to practically nonviable systems, as the traffic generated by such systems would exceed the capacity of the existing communication networks. Additionally, a recent study [2] has shown that, even when carefully optimized, P2P algorithms using traditional single-term indexes in structured P2P networks do not scale to web size document collections. Similarly, even for more sophisticated schemes, such as term-to-peer indexing [3, 4] or hierarchical federated architectures [5, 6], there is little evidence on whether these approaches can scale to web sizes.

The design of scalable models for full-text IR over P2P networks therefore remains an open issue. We argue that any solution to this problem should at least verify the following three properties: (1) it should support unrestricted multi-term queries; (2) it should provide retrieval performance comparable to state-of-the-art centralized search engines; and (3) it should scale to very large networks, possibly consisting of millions of peers. In addition, as the natural P2P solution for processing document collections that reach unmanageable sizes is to increase the number of available peers, we focus on use case scenarios in which the maximal number of documents each peer contributes to the global network can be assumed constant which again makes bandwidth consumption the major concern.

This paper introduces a novel indexing model that maintains indexing at document granularity and is characterized by the following central property: We carefully select the keys used for indexing so that they consist of terms and term sets that are *discriminative* with respect to the document collection, i.e. appear in a limited number of documents. Such keys, which may be seen as highly-selective multi-term queries associated with precomputed answer sets, enable efficient retrieval because of the short size of the associated posting lists.

However, as retrieval efficiency often comes at the price of an increased indexing cost, a theoretical analysis of the scalability of our indexing/retrieval model is crucial to asses the feasibility of the proposed approach, in particular, because the size of the key vocabulary can easily become unmanageable as it theoretically grows with $2^{|T|}$, where $|T|$ is the size of single-term vocabulary. We therefore perform a scalability analysis to demonstrate the viability of our model and to point out the salient properties that make it superior to existing alternative solutions. However, as the theoretical scalability analysis essentially concentrates on asymptotic properties, it provides few evidence about the practical feasibility of our approach in more realistic usage scenarios. In this perspective we have used our P2P retrieval engine to carry out an experimental study and gather empirical data about our indexing/retrieval method. These experiments confirm that the growth of the key vocabulary, as well as the size of the global index remain bounded with realistic upper bounds. Furthermore, the measured retrieval performance is comparable to the one achieved with a centralized single-term engine using the best state-of-the-art BM25 relevance computation scheme. Finally, the analysis of the total traffic generated during both indexing and retrieval demonstrates the potential of the key-based indexing approach to achieve orders of magnitude traffic reduction.

In summary, the main contributions presented in this paper are the following: (1) we introduce a novel indexing/retrieval model for full-text P2P search that relies on global key-to-document indexing to overcome the retrieval scalability problems encountered by existing solutions with respect to retrieval costs; (2) we provide a fully worked out theoretical scalability analysis of the proposed model; (3) we report experimental results obtained with our distributed prototype that confirm the practical feasibility of our key-based approach.

The rest of the paper is structured as follows: Section 2 analyzes related work in the area of full-text IR in P2P networks. In Section 3 we present our model and describe the key-based indexing and retrieval mechanisms. Section 4 presents the scalability analysis, while experimental results obtained with our truly-distributed prototype implementation are given in Section 5. Finally, Section 6 provides a general conclusion and sketches out possible future steps.

2 Related Work

There are two architectural concepts for designing P2P retrieval engines in the area of IR: a) federated engines in unstructured P2P networks, and b) global inverted index in structured P2P networks. The first strategy [7] relies on peers maintaining indexes of their local document collections. Such indexes are in principle independent, and a query is broadcasted to all the peers generating an enormous number of messages, while more advanced approaches restrict the amount of messages by random walks. The second strategy [8] distributes the global document index over a structured P2P network and each peer is responsible for a part of the global vocabulary and their associated posting lists. Queries are processed by retrieving posting lists associated with query terms from the global P2P index.

A number of solutions have been proposed to cope with the scalability problem of federated engines using the principle of answering a query at two levels, the peer and document level. First, a group of peers with potentially relevant local collections is detected; second, the query is submitted to the identified peers which return answers from their local indexes; and finally, the retrieved answers are merged to produce a single ranked answer set. Some engines use term-to-peer indexing where the indexed units are peers instead of individual documents: PlanetP [3] gossips compressed information about peers' collections in an unstructured network, while MINERVA [4] maintains a global index with peer collection statistics in a structured overlay to facilitate the peer selection process. Orthogonal solutions are proposed based on hierarchical P2P overlays where a backbone P2P network maintains a directory service which routes queries to peers with the relevant content [7, 6, 9].

Since large posting lists are the major concern of global single-term indexing, both [8] and [10] have proposed top-k posting list joins, Bloom filters, and caching as promising techniques to reduce search costs for multi-term queries. However, a recent study [2] shows that single-term indexing is practically unscalable for web sizes even when sophisticated protocols using Bloom filters are combined to reduce retrieval costs. A distributed top-k approach [5] is a viable solution for bandwidth scalability, but the open problem is related to retrieval performance of such a solution. Therefore, our approach comes as a completely novel solution for global document-level indexing in structured P2P networks. The idea of indexing term sets is somewhat similar to the set-based model [11] that indexes term sets occurring in queries. In contrast to our indexing scheme, the set-based model has been used to index frequent term sets in a centralized setting.

We have presented the general indexing idea and the architecture of our P2P retrieval engine in [12], and in this paper we focus on scalability aspects stressing that the presented theoretical scalability analysis is among the first in the field: Currently, the reasoning about system viability mostly relies on simulations and few comparative analysis are available. [13], for example, provides a performance study of structured, unstructured and hierarchical web retrieval solutions and reports that in terms of bandwidth the hierarchical solution performs slightly better than the other two architectures, but structured P2P offers the best response time. In that respect our engine also offers good response time, while it significantly reduces bandwidth required for retrieval.

3 Indexing and Retrieving with Highly Discriminative Keys

Let us consider a structured P2P network with N peers $P_i, 1 \leq i \leq N$ and a possibly very large document collection $\mathcal{D}$, consisting of M documents $d_j, 1 \leq j \leq M$. M is referred to as the size of $\mathcal{D}$, while the total number of term occurrences denoted by D is referred to as the sample size of $\mathcal{D}$. Furthermore, T denotes the term vocabulary in $\mathcal{D}$.

In the P2P network, each of the peers P_i plays two complementary roles. First, P_i stores a fraction of the global document collection $\mathcal{D}$, denoted by $\mathcal{D}(P_i)$. Then it contributes to build, store, and maintain the global inverted index that associates indexing features to the documents of $\mathcal{D}$. The fraction of the global index under the responsibility of P_i consists of all the keys and associated posting lists (i.e. document references) that are allocated to P_i by the Distributed Hash Table (DHT) built by the P2P network.

As far as indexing is concerned, each peer P_i is responsible for two complementary tasks. First, it is responsible for indexing $\mathcal{D}(P_i)$, i.e. for computing the indexing keys and associated posting lists that can be locally derived from $\mathcal{D}(P_i)$, and for inserting them into the global P2P index. Second, P_i is responsible for maintaining its fraction of the global index. More precisely, P_i maintains pairs of the form $(k, PL(k))$, where k is a key that P_i is responsible for and $PL(k) = \{d_j \in \mathcal{D} | k \in d_j\}$ is the posting list associated with k. Notice that a $(k, PL(k))$ pair stored in the fraction of the global index under the responsibility of P_i has no a priori reason to be the one that P_i extracts from $\mathcal{D}(P_i)$.

As far as retrieval is concerned, each peer P_i is responsible, when receiving a query q, for interacting with the global network in order to retrieve the list of documents from $\mathcal{D}$ that contain indexing keys that maximally overlap with q.

A detailed description of the above-mentioned indexing and retrieval models is given below. Additional information on more system-related aspects of these models can also be found in [14].

3.1 Indexing Model

The central principle underlying our indexing model is quite intuitive and is depicted in Figure 1. Instead of indexing with single-terms, which might lead to potentially very large posting lists as it is the case for the naïve approach, we index with selected terms and term sets, hereafter called the *keys*, that occur in at most DF_{max} documents, where DF_{max} is a parameter of our model. The crucial characteristic of this new indexing method is that it leads to an increase in the total number of indexing features (keys), but, at the same time, strictly limits the size of the associated posting lists to DF_{max}, which bounds the traffic generated during retrieval. This approach is fully in line with the general properties of P2P networks that can easily store large amounts of data (provided that enough peers are available), but must be carefully controlled wrt the volume of information transmitted between the peers.

However, even if P2P networks can provide very large storage capabilities, if no special care is taken, the set of indexing keys can still become unmanageable in size because it theoretically grows with $2^{|T|}$. Thus, the major issue we have to cope with is to find a key computation mechanism which generates key sets of scalable size, while preserving a good retrieval performance. Such a key generation mechanism, relying on the combination of adequately defined key filtering methods, is presented hereafter.

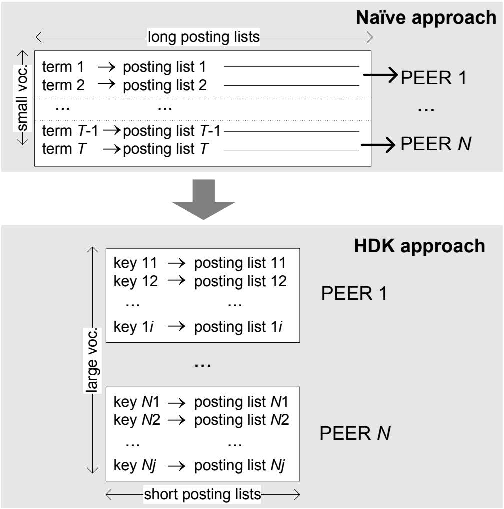

Fig. 1. The basic idea of HDK indexing

Definition 1. *A key k is defined as any set of terms $\{t_1, t_2, \ldots, t_s\}, t_i \in T$. The set of keys that can be extracted from a document collection $\mathcal{D}$ is denoted by $K_{\mathcal{D}}$ (or simply K), and the number of terms present in a key is referred to as the size of the key.*

The set of all keys of size s is denoted by K_s and $K = \bigcup_{s=1}^{s_{max}} K_s$, where s_{max} is the maximal size of a key in $\mathcal{D}$. Notice that size-1 keys correspond to single-terms and therefore $K_1 = T$.

Size filtering. As keys can be interpreted as selective queries associated with precomputed answer sets, the average size of a query submitted by users to the retrieval system is a crucial parameter for our indexing model. For example, in the case of web retrieval, the current average query size is estimated to be between 2 and 3 terms. We therefore define a maximal size s_{max} for the size of the keys to be considered. Furthermore, limiting the size of the considered keys does not have any substantial impact on the global indexing quality because, for a well chosen value of s_{max}, most of the user queries have a size smaller than or equal to s_{max}, and for the few queries of size bigger than s_{max}, the retrieval mechanism described in Section 3.2 is applied.

Proximity filtering. This filtering method uses the notion of textual context to reduce the number of generated keys. More precisely, we only keep keys containing terms that all appear in the same textual context, e.g., the same sentence, paragraph, or fixed-size document window. The underlying argumentation is that words appearing close to each other in documents are good candidates to also co-occur in queries. For example, the analysis presented in [15] reports the importance of text passages, considered as more relevant to user queries than the full documents. A similar reasoning is used in a recently proposed method for static index pruning [16] which indexes 'significant sentences', i.e. phrases appearing in similar contexts. In our indexing model, we use the simplest textual context, a fixed-size window, and consider as keys only term sets exclusively consisting of terms occurring in a window of size w, where w is a parameter of our model.

Notice that size and proximity filtering only rely on local information, i.e. they do not require the knowledge of the global document collection. They can therefore be performed fully independently by each peer P_i on its local document collection $\mathcal{D}(P_i)$. For the rest of this paper, we only consider terms and term sets that result from size and proximity filtering.

Definition 2. *For any given document collection $\mathcal{D}$, $K_{\mathcal{D},sw}$ (or simply K_{sw}) is the set of keys of maximal size s_{max} that exclusively consist of terms occurring in $\mathcal{D}$ within at least one document window of size w.*

Discriminative and non-discriminative keys. Each key $k \in K_{sw}$ has an associated document frequency $df_{\mathcal{D}}(k)$ corresponding to the number of documents in the collection $\mathcal{D}$ that contain k in a window of size w.

Given a document frequency threshold DF_{max} such that $1 \leq DF_{max} \leq M$, we use the key document frequencies to classify the keys into two distinct categories: the *discriminative* keys and *non-discriminative* keys.

Definition 3. $K_{\mathcal{D},d} = \{k \in K_{sw} | df_{\mathcal{D}}(k) \leq DF_{max}\}$ *is the set of* discriminative keys *(DKs), i.e. the keys that appear in at most DF_{max} documents and therefore have a high discriminative power wrt $\mathcal{D}$.*

Definition 4. $K_{\mathcal{D},nd} = \{k \in K_{sw} | df_{\mathcal{D}}(k) > DF_{max}\}$ *is the set of* non-discriminative keys *(NDKs), i.e. the keys with low discriminative power wrt $\mathcal{D}$.*

To simplify the notations, and because our analysis is focussed on the use of a single collection $\mathcal{D}$, $K_{\mathcal{D},d}$ and $K_{\mathcal{D},nd}$ will be simply denoted by K_d, and K_{nd} respectively.

Notice that the DKs (resp. NDKs) verify the following *subsumption property*: Any key containing a DK of smaller size is also a DK. Any key contained in an NDK of bigger size is also an NDK. In addition, for a key to be globally non-discriminative (i.e. non-discriminative in the global document collection $\mathcal{D}$), it is sufficient that it is *locally* non-discriminative, i.e. that it is non-discriminative in any of the local document collections $\mathcal{D}(P_i)$. These properties are important for the redundancy filtering method described below.

Redundancy filtering. This filtering method relies on the subsumption property of the DKs to further reduce the number of generated keys. If a key k_1 contains a discriminative key k_2 of smaller size, then k_1 is also discriminative and the answer set $PL(k_1)$, which is contained in $PL(k_2)$, can be produced by local postprocessing of $PL(k_2)$. In other words, k_1 is practically redundant with k_2 and therefore does not need to be stored in the global index.

Definition 5. *A key k is* intrinsically discriminative *iff it is discriminative and all its sub-keys of strictly smaller size are non-discriminative.*

In other words, redundancy-based filtering consist in considering only intrinsically discriminative keys for indexing. This again strongly reduces the number of generated keys, but, due to the subsumption property, fully preserves the indexing exhaustiveness, i.e. all the answer sets that can be generated with an index consisting of discriminative keys can also be generated with an index restricted to the intrinsically-discriminative keys. In addition, the notion of intrinsically-discriminative key also provides a more precise way for defining the s_{max} threshold used for size filtering. Indeed, if we take s_{max} to be the maximal size of the intrinsically-discriminative key in $\mathcal{D}$, then it is guaranteed that size filtering also preserves indexing exhaustiveness.

Definition 6. *A key k is* highly discriminative *iff (1) $|k| \leq s_{max}$ (size filtering); (2) $k \in K_w$ (proximity filtering); and (3) k is intrinsically discriminative (redundancy filtering). In the rest of this paper, highly discriminative keys will be referred to as HDKs.*

Notice that HDKs verify the following central property: Any key that is locally highly-discriminative (i.e. highly-discriminative in any of the local document collections $\mathcal{D}(P_i)$) is, either globally highly-discriminative, or globally non-discriminative.

Computing the global index. The goal of the indexing algorithm is to produce, for any given global document collection $\mathcal{D}$ split over N peers, all the keys that are either globally non-discriminative, or globally highly-discriminative, and to associate with them the corresponding global posting lists. Full posting lists are stored for HDKs, while the posting lists for NDKs are truncated to their top-DF_{max} best elements.

Since the indexing process is computationally intensive, peers share the indexing load to collaboratively build the global index. Each peer P_i performs its local indexing in several iterations, starting by computing single-term keys, then 2-term keys, ..., and finally s_{max}-term keys. For any current key-size s, P_i computes its local size-s HDKs and NDKs and inserts them in the global network, along with the, possibly truncated, associated local posting lists.

At the global level, the P2P network maintains the global posting lists, i.e. updates the top-DF_{max} posting lists for the NDKs if necessary, and, if any of the inserted local HDKs become globally non-discriminative, notifies the peers that have submitted such key so that they update their local knowledge for the production of the local keys of bigger size. The computation of the local size-s HDKs only requires knowledge about the global document frequencies of the local size 1 and size $s-1$ NDKs, as these are the only ones required for the size s key generation. The global knowledge about HDK and NDK frequencies is maintained in the global P2P index. As the number of the local NDKs is very small wrt to the one of the local HDKs, this is a crucial property for guaranteeing the computational efficiency of our indexing algorithm. Further details about the computational mechanism and a formal algorithm are provided in [12].

3.2 Retrieval Model

The general idea behind our retrieval mechanism is to consider each of the queries $q = \{t_1, t_2, \ldots, t_{|q|}\}$, where $|q|$ is the size of the query and $t_i \in T$, as a document collection consisting of a unique document (the query itself), and to apply a procedure very similar to the previously described indexing mechanism to identify, in the lattice of query term combinations, the term sets corresponding to global HDKs or NDKs. For the identified keys, the associated postings are retrieved from the global index, and are merged (simple set union) into a single posting list that is subsequently ranked using our distributed ranking implementation.

More precisely, the retrieval mechanism might in theory require the exploration of $(\binom{|q|}{1} + \binom{|q|}{2} + \ldots + \binom{|q|}{s_{max}})$ query term subsets. In practice, due to the smart use of the HDK and NDK related subsumption properties and to the quite limited size of the queries submitted by the users in the case of web retrieval, the number of messages generated during retrieval remains in fact very limited and fully scalable.

4 Scalability analysis

To assess the scalability of the HDK approach, we analyze the indexing and retrieval costs in terms of the number of transmitted postings in the peer network because these make the dominant part of the generated traffic. We are interested in the upper bound on the number of postings associated with the HDK index that have to be transmitted through the network during indexing, and the number of postings transmitted during retrieval. To simplify the analysis, we do not analyze the total traffic between the peers related to P2P network maintenance and query routing and merely analyze the number of postings the network needs to absorb and transmit to examine whether the approach has the potential to scale in P2P overlays.

4.1 Indexing Scalability

The indexing scalability is evaluated for document collection $\mathcal{D}$ of total size D when varying the size of collection sample l, $1 \leq l \leq D$. We assume documents from $\mathcal{D}$ are concatenated and l counts the number of first term occurrences in $\mathcal{D}$. The scalability analysis is based on the distribution of term frequencies because it is currently well explored in the literature [17].

Rare, frequent, and very frequent keys. Each key $k \in K$ has an associated collection frequency $f_{\mathcal{D}}(k)$ corresponding to the number of occurrences of k in the whole collection $\mathcal{D}$. Given the two frequency thresholds F_f and F_r such that $1 \leq F_r \leq F_f \leq D$, we use key frequencies to classify the keys into three distinct categories: the set of *rare*, *frequent*, and *very frequent* keys defined as follows:

Definition 7. $K_{\mathcal{D},r} = \{k \in K | F_r \geq f_{\mathcal{D}}(k)\}$ *is the set of* rare keys*; i.e. keys that occur less that F_r times in $\mathcal{D}$.*

Definition 8. $K_{\mathcal{D},f} = \{k \in K | F_f \geq f_{\mathcal{D}}(k) > F_r\}$ *is the set of* frequent keys*; i.e. keys that are frequent but do not occur more than F_f times in $\mathcal{D}$.*

Definition 9. $K_{\mathcal{D},vf} = \{k \in K | f_{\mathcal{D}}(k) > F_f\}$ *is the set of* very frequent keys*; i.e. keys that occur more that F_f times in $\mathcal{D}$.*

To simplify the notation, $K_{\mathcal{D},r}$, $K_{\mathcal{D},f}$, and $K_{\mathcal{D},vf}$ will be simply denoted by K_r, K_f, and K_{vf} respectively. By definition, $df_{\mathcal{D}}(k) \leq f_{\mathcal{D}}(k)$ and thus we have the following two properties.

Corollary 1. If $DF_{max} = \max[df_{\mathcal{D}}(k) | k \in K_r]$, then all rare keys are discriminative and $K_r \subseteq K_d$.
Proof. As $\forall k \in K_r$, $df_{\mathcal{D}}(k) \leq DF_{max}$, all rare keys are discriminative by Definition 3.

Corollary 2. A frequent key $k \in K_f$ may be discriminative.
Proof. For frequent keys the following properties hold: $f_{\mathcal{D}}(k) > F_r \geq DF_{max}$ and $f_{\mathcal{D}}(k) \geq df_{\mathcal{D}}(k)$. There may be $k \in K_f$ such that $f_{\mathcal{D}}(k) = DF_{max}$ and $f_{\mathcal{D}}(k) = df_{\mathcal{D}}(k)$, and therefore $df_{\mathcal{D}}(k) = DF_{max}$, i.e. k is discriminative.

Zipf model. Zipf laws constitute a parametric function family that provides good fitting function candidates for the approximation between the term collection frequencies and term collection ranks, where a collection rank of a term t in collection $\mathcal{D}$ is defined as the number of distinct terms in $\mathcal{D}$ having a collection frequency larger than $f_{\mathcal{D}}(t)$ [17]. The quality of the zipfian approximations is usually increasing with the size of the collection $\mathcal{D}$. More formally, for a zipf law with a skew a and a scale C, the collection frequency of a term t with a zipf rank r is approximated by $z(r) = C \cdot r^{-a}$. In addition, as the standard zipfian assumption is that the scale parameter depends on the size of the collection while the skew does not, for any term $t \in \mathcal{D}$, if $r(t,l)$ is the collection rank of t within the l first occurrences of $\mathcal{D}$, we can write $f_{\mathcal{D}}(t,l) \approx z_{\mathcal{D}}(r(t,l)) = C(l) \cdot r^{-a}$.

Figure 2 depicts two zipf functions with $a = 1.5$ modeling frequency distributions for two sample sizes. F_f and F_r are independent of collection size and determine the values of term ranks r_f and r_r, $r_f \leq r_r$, that change when increasing l. It is visible that $r_{f1} < r_{f2}$ and $r_{r1} < r_{r2}$ because $l_1 < l_2$.

Key occurrence probabilities. As we define DF_{max} to be $\max[df_{\mathcal{D}}(k) | k \in K_r]$, by Corollary 1 all rare keys are discriminative. The worst case scenario for our analysis is when $K_r = K_d$ and $K_f = K_{nd}$ as there are no frequent and discriminative keys that are HDKs, since all frequent keys are non-discriminative and have to be expended to form new HDKs. Furthermore, we perform the scalability analysis using collection frequencies instead of document frequencies because, although the HDK index associates postings to documents, the size of the positional index gives an upper bound on

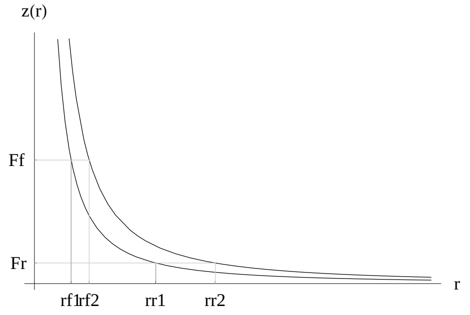

Fig. 2. Zipf function

the HDK index size. First we estimate occurrence probabilities for very frequent and frequent terms using the zipf function.

Theorem 1. *The probability of very frequent term occurrences $P_{\mathcal{D},vf}(l)$ for document collection $\mathcal{D}$ depends on l and can be calculated as*

$$P_{\mathcal{D},vf}(l) = \frac{1 - \frac{F_f}{C(l)}^{\frac{a-1}{a}}}{1 - \frac{1}{C(l)}^{\frac{a-1}{a}}}. \tag{1}$$

The analysis of Equation 1 shows that $P_{\mathcal{D},vf}$ depends on l and when $l \to D$, P_{vf} depends strongly on $\frac{F_f}{C(l)}$. Depending on F_f, the probability of very frequent term occurrence may be extremely high for very large document collections and we therefore do not use such terms for building the key vocabulary. Very frequent terms are further on disregarded from our analysis.

Theorem 2. *The probability of frequent term occurrences $P_{\mathcal{D},f}$ is a characteristic constant of document collection $\mathcal{D}$, i.e. it does not depend on l and can be calculated as*

$$P_{\mathcal{D},f} = \frac{1 - \frac{F_r}{F_f}^{\frac{a-1}{a}}}{1 - \frac{1}{F_f}^{\frac{a-1}{a}}}. \tag{2}$$

The analysis of Equation 2 shows that $P_{\mathcal{D},f}$ depends on the two constants F_f and F_r, and the skew parameter a, but it does not depend on $C(l)$. Therefore, $P_{\mathcal{D},f}$ is independent of sample size and the growing collection size because the skew factor a converges to a constant value when $l \to D$. The probability of rare term occurrences $P_{\mathcal{D},r} = 1 - P_{\mathcal{D},f}$ analogously does not depend on l.

The analysis of term occurrence probabilities reveals an interesting property: The occurrence probability of both rare and frequent terms is independent of collection size and converges to a constant value for large document collections. However, the occurrence probability of very frequent terms does depend on l and may become very large:

Therefore, we are removing an increasing number of very frequent terms from T when building the key vocabulary following the common practice from the IR domain of removing stop words. The only difference is that the set of stop words depends on the constant F_f and will be increasing with l.

The presented occurrence probabilities are related to single-term keys K_1 and to simplify the notation we denote $P_{\mathcal{D},f}$ as $P_{f,1}$. The zipf law can be extended to model frequency distributions for keys of size s, K_s, and the associated key occurrence probability for frequent keys $P_{f,s}$ is a function of the skew parameter a_s that varies for keys of different sizes.

Estimating index size. Here we want to estimate the upper bound on the positional index size $IS_s(D)$ for document collection of size D associated with both rare and frequent keys of size s because it is the upper bound of the HDK index size. For K_1 we have $IS_1 \leq \sum_{k \in K_1} f_{\mathcal{D}}(k_i) = D$, and $\frac{IS_1}{D} \leq 1$. To estimate the index sizes associated with K_s, we assume the independency of frequent term co-occurrence when estimating the number of combinations of s frequent terms within a window of size w.

Theorem 3. *The index size $IS_s(D)$ associated with HDKs and NDKs of size s can be estimated as*

$$IS_s(D) = D \cdot P^2_{f,(s-1)} \cdot \binom{w-1}{s-1} . \tag{3}$$

Thus, $\frac{IS_s(D)}{D} = c$, where c is a constant that can be estimated based on the parameters of our model (w, F_f and DF_{max}) and the skew parameter a_s that describes the document collection. This proves that the upper bound on the index size can be estimated a priori and such estimations are provided for our test collection in Section 5.

4.2 Retrieval Scalability

As our retrieval model applies a rather simple procedure, the scalability wrt retrieval is bounded by D_{max} and the number of keys a query is mapped to. For a query of size $|q|$ in case $|q| \leq s_{max}$ the number of keys that are mapped to a query is $n_k = 2^{|q|} - 1$, while if $|q| > s_{max}$ the number of keys is $n_k = \binom{|q|}{s_{max}} + \binom{|q|}{s_{max}-1} + ... + \binom{|q|}{1}$. The upper bound on the generated traffic is therefore $n_k \cdot DF_{max}$. Since the values for s_{max} and $|q|$ are typically low for web queries, the true upper bound is relatively small. For example, the average size of a query is 2.3 in the Wikipedia query log, and $n_k \approx 3.92$. The experimental results presented in Section 5 show that the retrieval traffic is indeed small and scalable compared to distributed single-term indexing.

5 Experimental evaluation

Experiments have been performed using our prototype retrieval engine built on top of the P-Grid P2P layer [18]. Our implementation is a fully-functional P2P retrieval engine that integrates a solution for distributed maintenance of global key vocabulary with associated document frequencies, calculates HDKs used for indexing in a completely distributed fashion, stores posting lists in a global index for computed keys, and integrates a solution for distributed content-based ranking.

Experimental setup. The experiments were carried out using a subset of documents from the Wikipedia collaborative encyclopedia[1]. It is chosen for its availability of content and a large user base which represents a good case study for the web. The collection statistics are listed in Table 1.

Table 1. Wikipedia statistics

total number of documents M	653,546
size in words D	3 million words
average document size	225 words

Performance analysis. The experiments investigate indexing and retrieval costs, and compare the retrieval performance achieved by our P2P engine to the one obtained by a centralized single-term engine using the top-scoring BM25 relevance scheme. To simulate the evolution of a P2P system, i.e. peers joining the network and increasing the document collection, we started the experiment with 4 peers, and added additional 4 peers at each new experimental run. The total of 28 peers running on Linux Red-Hat PCs with 1GB of main memory and connected via 100 Mbit Ethernet were used in the experiments. The prototype system is implemented in Java. All documents are pre-processed: First we remove 250 common English stop words and apply the Porter stemmer, and then we removed additional very frequent terms. The parameters used in the experiments are listed in Table 2.

Table 2. Parameters used in experiments

number of peers N	4, 8, . . ., 28
documents per peer	5,000
size in words l	1,123,000 per peer
DF_{max}	400 and 500
F_f	100,000
w	20
s_{max}	3

Indexing. To quantify indexing costs and the influence of DF_{max} on required storage and bandwidth consumption, we investigate the average number of postings stored per peer (Figure 3) and the average number of postings inserted by a peer into the global index (Figure 4), and compare it to the size of the single-term index. Both curves are increasing for the HDK approach since the document collection sizes in our experiment are rather small, but are expected to reach a constant value as predicted by the scalability analysis. It is visible that a peer stores significantly more postings associated with HDKs

[1] http://www.wikipedia.org/, available for download from http://download.wikimedia.org/

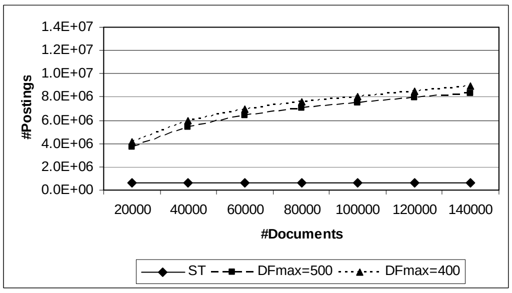

Fig. 3. Stored postings per peer (index size)

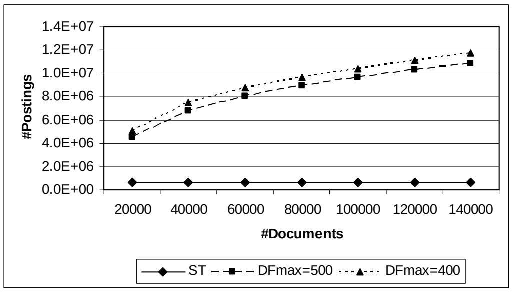

Fig. 4. Inserted postings per peer (indexing costs)

when compared to single-term indexing (13.9 times more for 140,000 documents and $DF_{max} = 400$), but the higher indexing costs are still feasible. The HDK index size can be reduced when increasing DF_{max} because the HDK indexing is approaching single-term indexing. In case when DF_{max} would be equal to the maximum posting list size of a single-term index, the two indexing models would produce equal indexes. However, an increased value of DF_{max} causes higher traffic during retrieval and therefore must be carefully chosen to reflect the networking conditions and P2P engine usage model. The number of inserted postings per peer is larger than the number of stored postings because the P2P index stores top-DF_{max} postings associated with NDKs to improve the retrieval performance. This puts an overhead on the required bandwidth because all peers publish their locally produced top-DF_{max} postings associated with NDKs.

Figure 5 shows the ratio between locally calculated and inserted postings associated with keys of different sizes and D, and is used for comparison with the theoretical scalability analysis. The largest part of the index is currently related to K_2, but the index related to K_3 is slowly growing and is expected to be more substantial for larger collection size. The curves confirm our finding presented in Section 4.1 that $\frac{IS_1}{D} \leq 1$ while $\frac{IS_2}{D}$ and $\frac{IS_3}{D}$ are still growing to reach the constant value for very large D. Using Equation 3, the maximal estimated value for $\frac{IS_2}{D}$ is 12.16 ($a_1 = 1.5$ is fitted from true frequency distribution, and therefore $P_{f,1} = 0.8$) and for $\frac{IS_3}{D}$ the value is 11.35

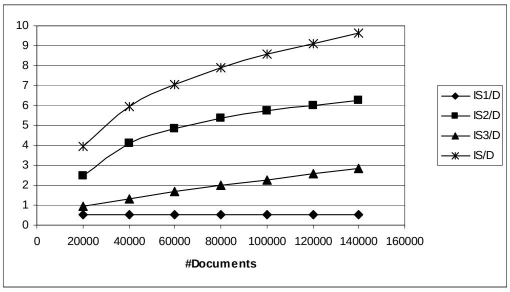

Fig. 5. Ratio between inserted IS and D

($a_2 = 0.9$ and $P_{f,2} = 0.257$). These values are large overestimations compared to experimentally obtained values, especially for keys of size 3 that will become more significant for larger collection sizes (values obtained experimentally are 6.26 and 2.82 respectively). However, the analysis shows that the generated indexing traffic associated with the HDK approach is at most 40.7 times bigger than the one associated with single-term indexing for very large D (single-term indexing produces on average 130 postings per Wikipedia document compared to 5290 postings per document by the HDK indexing).

Retrieval. To evaluate the retrieval performance, queries were extracted from a true Wikipedia query log available for 2 months (08/2004 and 09/2004). We have chosen 3,000 queries from 2,000,000 unique queries that have produced more than 20 hits from the indexed collection. The extracted queries contain on average 3.02 terms, with a minimum of 2 and maximum of 8 terms. Single term queries were not considered because they would generate bounded traffic associated with NDKs.

Figure 6 shows an enormous reduction of bandwidth consumption per query of the HDK-based approach compared to the naïve single term indexing. It shows unscalable linear growth of the retrieval traffic induced by the naïve single-term indexing while for the HDK-based approach the retrieval cost remains constant with a slightly larger traffic for $DF_{max} = 500$. As the major costs associated with P2P IR are related to retrieval, the significant traffic reduction compared to distributed single-term indexing practically demonstrates the effect of the bounded number of index postings to bandwidth consumption during retrieval.

The essential question remains whether the retrieval performance of the HDK approach is satisfactory and comparable to centralized counterparts. Due to the lack of relevant judgment for the used query set, we have compared the retrieval performance to a centralized engine[2] with BM25 relevance computation scheme which is currently considered as one of the top performing relevance schemes [19].

Figure 7 presents the overlap on top-20 documents retrieved by the HDK-based system and the Terrier search engine. We are interested in the high-end ranking as typical users are often interested only in the top 20 results. The comparison shows significant

[2] Terrier search engine, http://ir.dcs.gla.ac.uk/terrier/

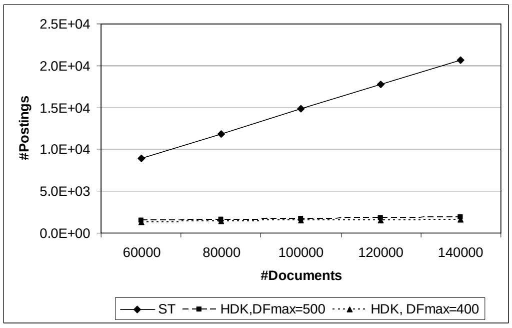

Fig. 6. Number of retrieved postings per query

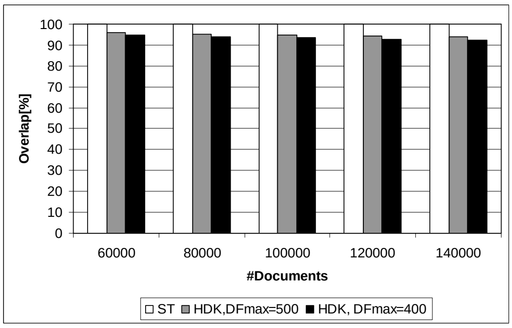

Fig. 7. Top-20 overlap with BM25 relevance scheme

and satisfactory overlap between the retrieved result sets. As expected, the retrieval performance is similar to single-term indexing for larger values of DF_{max}. There is obviously a trade-off between retrieval quality and bandwidth consumption of our indexing strategy because an increased value of DF_{max} results in an increased bandwidth consumption during retrieval, while on the contrary, offers retrieval performance that better mimics centralized engines. However, as bandwidth is the major obstacle for scalable traffic consumption, it is vital to choose an adequate value for DF_{max} taking into account available network capacity.

To investigate the profitability of the HDK-based indexing, we have plotted in Figure 8 the predicted generated traffic associated with both indexing and retrieval comparing the naïve single-term and HDK-based approach. The calculation assumes that indexing is done monthly and the corresponding query load per month is $1.5 \cdot 10^6$ (corresponds to the true number of queries from the query log). The analysis shows that for the whole Wikipedia collection (653,546 documents), the HDK approach would generate 20 times less traffic than the distributed single-term approach, while for 1 billion documents the ratio is around 42. However, as the number of queries is expected to

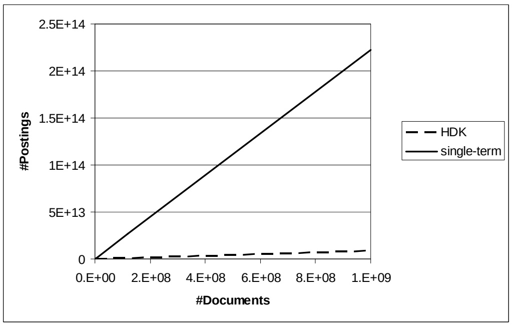

Fig. 8. Estimated total generated traffic

grow for web sizes, this ratio would be significantly larger and in favor of the HDK approach.

6 Conclusion

We have presented a novel HDK-based model for web retrieval within structured P2P networks. This model allows distributed cost-efficient and high performance retrieval as it is confirmed by the presented experimental results showing that our approach achieves a retrieval quality (top-k precision) comparable to the standard single term approach with the best state-of-the-art BM25 relevance computation scheme. The experiments carried out with our truly-distributed prototype provide further evidence that the proposed method is practically viable in large-scale distributed environments and that it produces indexes of realistic size. More importantly, the theoretical scalability analysis proves that, due to its cost-efficient retrieval mechanism, our approach has the potential to scale to very large document collections distributed over large numbers of peers. Finally, our model makes it possible to take into account the characteristics of the used document collection, the nature of the targeted usage model (e.g. the planed frequency of indexing and querying), and the network related capacity constraints, and can adequately adapt the various parameters of the model in order to meet desired indexing and retrieval traffic requirements.

Although an operational, fully distributed prototype has already been designed and tested, a lot of open issues still need to be investigated in more detail: the HDK generation process might integrate more semantics about the indexing keys in order to further reduce the size of the produced global index; the parameters of the model might be more adaptive in order to flexibly take into account changes in the working environment of the prototype; finally, the used distributed ranking procedure might be extended to incorporate more sophisticated, especially query-independent ranking schemes.

In conclusion, we believe that our model and the associated prototype convincingly demonstrate that P2P web-scale retrieval is fully feasible and hope that our work will contribute to the progress in a domain that is generally recognized as crucial for the development of less centralized and therefore more user-centered information dissemination and management techniques.

References

1. Li, J., Loo, B., Hellerstein, J., Kaashoek, F., Karger, D., Morris, R.: The feasibility of peer-to-peer web indexing and search. In: Peer-to-Peer Systems II: 2nd International Workshop on Peer-to-Peer Systems (IPTPS). (2003) 207–215
2. Zhang, J., Suel, T.: Efficient query evaluation on large textual collections in a peer-to-peer environment. In: P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05), Washington, DC, USA, IEEE Computer Society (2005) 225–233
3. Cuenca-Acuna, F.M., Peery, C., Martin, R.P., Nguyen, T.D.: PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. In: 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press (2003)
4. Bender, M., Michel, S., Triantafillou, P., Weikum, G., Zimmer, C.: Improving collection selection with overlap awareness in P2P search engines. In: SIGIR '05, New York, NY, USA, ACM Press (2005) 67–74
5. Balke, W., Nejdl, W., Siberski, W., Thaden, U.: Progressive distributed top-k retrieval in peer-to-peer networks. In: Proceedings of the 21st International Conference on Data Engineering (ICDE 2005). (2005)
6. Lu, J., Callan, J.: Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In: Advances in Information Retrieval, 27th European Conference on IR Research (ECIR). (2005) 52–66
7. Lu, J., Callan, J.: Content-based retrieval in hybrid peer-to-peer networks. In: Proceedings of the twelfth international conference on Information and knowledge management. (2003)
8. Reynolds, P., Vahdat, A.: Efficient Peer-to-Peer Keyword Searching. Middleware03 (2003)
9. Balke, W.T., Nejdl, W., Siberski, W., Thaden, U.: DL Meets P2P - Distributed Document Retrieval Based on Classification and Content. In: 9th European Conference on Research and Advanced Technology for Digital Libraries, (ECDL). (2005) 379–390
10. Suel, T., Mathur, C., Wu, J.W., Zhang, J., Delis, A., Kharrazi, M., Long, X., Shanmugasundaram, K.: ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. WebDB'03 (2003)
11. Pôssas, B., Ziviani, N., Wagner Meira, J., Ribeiro-Neto, B.: Set-based vector model: An efficient approach for correlation-based ranking. ACM Trans. Inf. Syst. **23** (2005) 397–429
12. Podnar, I., Luu, T., Rajman, M., Klemm, F., Aberer, K.: A Peer-to-Peer Architecture for Information Retrieval Across Digital Library Collections. In: To appear in European conference on research and advanced technology for digital libraries (ECDL 2006). (2006)
13. Yong Yang, Rocky Dunlap, M.R., Cooper, B.F.: Performance of Full Text Search in Structured and Unstructured Peer-to-Peer Systems. In: IEEE INFOCOM. (2006)
14. Podnar, I., Rajman, M., Luu, T., Klemm, F., Aberer, K.: Beyond term indexing: A P2P framework for web information retrieval. To appear in Informatica, Special Issue on Specialised Web Search (2006)
15. Salton, G., Allan, J., Buckley, C.: Approaches to Passage Retrieval in Full Text Information Systems. In: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (1993) 49–58
16. de Moura, E.S., dos Santos, C.F., Fernandes, D.R., Silva, A.S., Calado, P., Nascimento, M.A.: Improving web search efficiency via a locality based static pruning method. In: WWW '05:

Proceedings of the 14th international conference on World Wide Web, New York, NY, USA, ACM Press (2005) 235–244

17. Baayen, R.H.: Word Frequency Distributions. Dordrecht, Kluwer Academic Publishers (2001)
18. The P-Grid Consortium: The P-Grid project (2005) http://www.p-grid.org/.
19. Plachouras, V., He, B., Ounis, I.: University of Glasgow at TREC2004: Experiments in Web, Robust and Terabyte tracks with Terrier. In: Proceeddings of the 13th Text REtrieval Conference (TREC 2004). (2004)

Appendix

Theorem 1. Proof.

$$P_{vf}(l) = P\left\{t_i | z(t_i, l) > F_f\right\} = \frac{\sum_{r=1}^{r_f} z(r,l)}{\sum_{r=1}^{T} z(r,l)} \tag{4}$$

If we approximate the sum with an integral of the function $z(r,l)$, $P_{vf}(l)$ can be computed as

$$P_{vf}(l) = \frac{\int_1^{r_f} C(l) \cdot r^{-a} dr}{\int_1^{T} C(l) \cdot r^{-a} dr}. \tag{5}$$

As term frequency distributions are characterized by the presence of large numbers of terms with very low probabilities of occurrence [17], we can disregard hapax legomena in $\int_1^T C(l) \cdot r^{-a} dr$ and approximate it by $\int_1^{T'} C(l) \cdot r^{-a} dr$, where T' is the rank of the first hapax legomena, i.e. $z(T', l) = 1$. Therefore,

$$P_{vf}(l) = \frac{r_f^{(1-a)} - 1}{T'^{(1-a)} - 1}. \tag{6}$$

Using the inverse Zipf function $z^{-1}(y,l) = \left[\frac{C(l)}{y}\right]^{\frac{1}{a}}$, the probability of very frequent terms can be computed as

$$P_{vf}(l) = \frac{1 - \frac{F_f}{C(l)}^{\frac{a-1}{a}}}{1 - \frac{1}{C(l)}^{\frac{a-1}{a}}}. \tag{7}$$

Theorem 2. Proof.

$$P_f(l) = P\left\{t_i | F_f \geq z(t_i, l) > F_r\right\} = \frac{\sum_{r=r_f}^{r_r} z(r,l)}{\sum_{r=r_f}^{T} z(r,l)} \tag{8}$$

Ignoring both very frequent terms and hapax legomena we have

$$P_f = \frac{r_r^{(1-a)} - r_f^{(1-a)}}{T'^{(1-a)} - r_f^{(1-a)}} = \frac{1 - \frac{F_r}{F_f}^{\frac{a-1}{a}}}{1 - \frac{1}{F_f}^{\frac{a-1}{a}}}. \tag{9}$$

Theorem 3. Proof. Let us estimate postings associated with K_2. In the first window $w \in D$, the expected number of frequent term occurrences is $P_{f,1} \cdot w$ while the number of keys of size $s = 2$ that can be generated from the window is $\binom{P_{f,1} \cdot w}{s}$. The created 2-term keys are not necessarily distinct, and therefore we are counting the number of all 2-term keys occurrences (i.e. size of the positional index). By sliding the window one position further, new keys can be generated that consist of the new right-most term, provided that this term is frequent, and frequent terms appearing in any of the $(w - 1)$ remaining positions in the window. The expected number of newly created keys is $P^2_{f,1} \cdot (w - 1)$. There are $(D - w)$ windows when successively shifting the original window by one position to the right and the expected number of 2-term postings that can be generated is

$$IS_2(D) = \binom{P_{f,1} \cdot w}{k} + (D - w) \cdot P^2_{f,1} \cdot (w - 1) \tag{10}$$

As $D \gg w$, (10) can be simplified to

$$IS_2(D) = D \cdot P^2_{f,1} \cdot (w - 1). \tag{11}$$

Analogously, for index sizes $IS_s(D)$, we investigate the number of postings created by sliding the window. Notice that for building keys of size s we are using 2 overlapping frequent keys of size $s - 1$. The new right-most term creates $\binom{w-1}{s-2}$ keys of size $s - 1$ with terms from the previous $(w - 1)$ positions, while the expected number of frequent ones is $P_{f,(s-1)} \binom{w-1}{s-2}$. These keys are combined with $(s - 1)$-size keys built from $(w - 1)$ terms with an addition condition: they have to overlap on $s - 2$ terms. There are $(w - 1) - (s - 2)$ such keys per each newly-created key, and the total number is $\binom{w-1}{s-2} \cdot \frac{(w-1)-(s-2)}{s-1} = \binom{w-1}{s-1}$. The expected index size for K_s is therefore

$$IS_s(D) = D \cdot P^2_{f,(s-1)} \cdot \binom{w-1}{s-1}. \tag{12}$$