STAIRS: Towards Efficient Full-Text Filtering and Dissemination in a DHT Environment

Weixiong Rao[†] Ada Wai-Chee Fu[†] Lei Chen[‡] Hanhua Chen[♯]

[†]Department of Comp. Sci. and Eng.
The Chinese University of Hong Kong
{wxrao,adafu}@cse.cuhk.edu.hk

[‡]Department of Comp. Sci. and Eng.
Hong Kong University of Sci.and Tech.
leichen@cse.ust.hk

[♯]School of Computer Sci.
Huazhong University of Sci. and Tech.
huanhua@hust.eud.cn

***Abstract*— Nowadays contents in Internet like weblogs, wikipedia and news sites become "live". How to notify and provide users with the relevant contents becomes a challenge. Unlike conventional Web search technology or the RSS feed, this paper envisions a personalized full-text content filtering and dissemination system in a highly distributed environment such as a Distributed Hash Table (DHT). Users can subscribe to their interested contents by specifying some terms and threshold values for filtering. Then, published contents will be disseminated to the associated subscribers. We propose a novel and simple framework of filter registration and content publication, STAIRS. By the new framework, we propose three algorithms (default forwarding, dynamic forwarding and adaptive forwarding) to reduce the forwarding cost and false dismissal rate; meanwhile, the subscriber can receive the desired contents with no duplicates. In particular, the adaptive forwarding utilizes the filter information to significantly reduce the forwarding cost. Experiments based on two real query logs and two real datasets show the effectiveness of our proposed framework.**

I. Introduction

The contents of Internet such as weblogs, wikipedia and news sites are growing in an amazing speed. For example, in July 2006, there were over 1.6 million blog postings every day; the number of blogs worldwide was reported as 50 million and is doubled every 6 months [14]; in Wikipedia, which has 2,163,836 articles in the English version, the increase of average addition per day during the period from 2002-01-01 to 2007-01-01 is from 54 to 1822 [2]. These staggering numbers suggest a significant shift in the nature of Web content from mostly static pages to continuously created and updated documents.

With such "live" contents, how to notify users about their interested contents becomes a highly helpful and a very challenging task. Web search technologies can retrieve Web documents for input queries, however, it is ill-suited to notify users about rapidly-changing contents. Though Google Alerts [3] and Microsoft Live Alerts [1] provide the dissemination service for end users, it is believed that both alert services are based on batch processing through the search engine [9]. Thus, they cannot offer timely dissemination service. Many users now use RSS feeds, which allow them to receive rapid updates from sites of interest. However, one RSS feed only covers an individual site, such as a blog, hence end users need to subscribe to multiple sites in order to satisfy their various interests. Moreover, RSS feeds adopt a topic based publish/subscribe approach. Due to the lack of content-based subscription, articles that are sent by RSS may not correspond to the real interests of subscribers, resulting in false dismissals and false alarms.

In view of the shortcomings of previous approaches such as no timely dissimilation or no content-based services, we aim to design a personalized threshold-based full-text content filtering and dissemination system in a large distributed environment like Peer-to-Peer (P2P) network. In such a system, end users can subscribe their interested contents by entering some *terms* as a filtering condition (i.e. a subscription *filter*). The system will disseminate refresh contents that match a subscription filter to the corresponding subscriber.

To disseminate the relevant documents to a subscriber, we employ a score function such as *term frequency * inverse document frequency* (tf*idf) to measure the relevance between published contents and subscription filters. Only the document with a score higher than a system default threshold or a user specified threshold will be disseminated to users. In this work, we use both terms and thresholds in the filter conditions is based on the following observations: (1) term input has become the de-facto standard for end users to retrieve their interested documents in Web search engines, and the tf*idf score function is widely used in Information Retrieval (IR) context such as vector space model (VSM) to measure the relevance; (2) the whole content space is extremely huge, while the subscription filters are usually composed of a few terms, therefore without setting a threshold value, the matching result set for the filter terms against the content space will be very large; this is especially true for terms that frequently appear in documents.

In a simpler setting, subscribers may adopt a *default* threshold value that is provided by the dissemination system itself. However, some subscribers may specify their own thresholds to meet their specific requirements.

A. Problem Statement

We formalize the key problem of building such a dissemination system.

Problem Statement: We assume that each document d is associated with $|d|$ pairs of $\langle t_i, score(t_i, d)\rangle$, where t_i is a term in d, and $score(t_i, d)$ represents the importance or weight of term t_i in d, which can be pre-computed using the *term frequency * inverse document frequency* (tf*idf) scheme.

Each subscription filter f consists of $|f|$ terms $f = \{t_1, .., t_{|f|}\}$ and a threshold $\mathrm{T}(f)$. The value of $\mathrm{T}(f)$ can be

the default value T or a personalized value greater or less than T. Document d will be disseminated to the subscriber with filter f provided that

$$TotalScore(f,d) = \sum_{i=1}^{|f|} score(t_i,d) \geq \mathrm{T}(f)$$

A matching scheme should enable document d to be disseminated to each interested subscriber at **no or low false dismissal rate** and **low publication cost**, which is determined by the document forwarding cost in a P2P system. Also, the documents should be disseminated in a **timely** manner; otherwise the validity may expire and the documents become useless.

B. Previous Approaches

In distributed IR systems based on DHT look up, such as [23], [10], [18], each document d is stored in the *home nodes* for *all* terms in d. To retrieve the interested documents, an end user enters a query f composed of a few terms, then DHT routes f to the home node for *each* term in f. The returned results from $|f|$ home nodes are intersected and aggregated to determine the final result. If we directly apply the above idea to our problem, filter f will first be registered to $|f|$ home nodes, then d is forwarded to $|d|$ home nodes, and finally d is matched with f in the home node for the common terms that appear in both d and f. There are two problems for such scheme: (1) The cost to forward d to $|d|$ home nodes is high because the number of terms in d, $|d|$, is typically large, for example in our experiments the number of terms per document of one data set is around 6000 on average. (2) Since each filter is registered at the home node of each term in the filter, forwarding document d to the home nodes of all terms in d will produce *duplicated* dissemination.

Previous work KLEE [10] vertically stores the document information to the home node of each term in the document and uses a thresholding approach to answer distributed top-k queries in a *top-down* manner: in each round, only several top candidate items are chosen from each sorted input index-list; and the next two or three rounds try to go down the input index list and choose more candidates to avoid false dismissal. However, the multiple rounds of candidate items fetching between query initiators and document home nodes will be time consuming and cannot guarantee *timely* delivery. Though KLEE proposes an hybrid structure of histogram and bloom filter, HistogramBlooms, to collect the document term information, which is similar to our proposed structure, HiBlooms, in fact, our HiBlooms is used to summarize the filter information (including filter thresholds, filter lengths and query terms). The Bloom filters in each cell of HiBlooms are further grouped by filter length and also are dynamically created to reduce the false positive rate; while in HistogramBlooms, each cell contains one Bloom filter.

Publish/Subscribe systems like [6] register each filter composed of multiple attribute predicates to *one* attribute index with the most selective principle. However, the approach of [6] is not applicable: document d is forwarded to each predicate list associated to each attribute in the schema, that is well-suited for the data schema with several attributes. Following this idea, forwarding d to home nodes for all terms that appear in the dissemination system is extremely costly, which is actually identical to the document broadcast across DHT. In Sieve [8] as a distributed content dissemination system, the counting algorithm suffers from the high latency problem because, for a subscription involving predicates over multiple attributes, the intermediate matching result of each attribute is maintained before the whole subscription is finally evaluated.

For an information dissemination system like SIFT[22] and InRoute [4], the content matching and dissemination are conducted by a centralized indexing structure. Though intended for the large distributed environments, FeedTree [16] and Corona [12] adopt topic or subject-based content dissemination, using the URL address as the subscription topic or subject. With consideration of the high maintenance issue due to a large number of topics in the topic based pub/sub system, [11] devises a dynamical clustering method for reducing the maintenance overhead. Instead, our proposed fine-grained content filtering model is free from topic maintenance while reducing the forwarding cost.

C. Our Approach

To guarantee low forwarding cost, no (or a low rate of) false dismissal, and timely dissemination, we propose a threshold-based approach to exclude useless terms in the documents. Specifically, we require each subscriber to register his/her filter f to the home node of each term in f, this is called "*full*" registration. When a document is published, based on a given threshold (system default or user's specification), only a few terms are selected from the published document for forwarding, this is called "*partial*" forwarding. We call the threshold based full-text filtering and dissemination system that is enabled by the full registration and partial forwarding as *STAIRS*.

In *STAIRS*, we also design the mechanism to avoid duplicate document dissemination. Moreover, we utilize the fact that most of the queries (filters) are expressed with a few terms to remove more useless terms for dissemination. Finally, for users with personalized thresholds, we utilize the filter information to tune the system thresholds and to reduce the number of terms as candidates for distribution. In summary, the contributions of this paper include:

- a new thresholding approach is proposed so that useless terms in a document d are excluded and d is not forwarded to the home nodes of these useless terms, saving a lot of forwarding cost;
- the basic filtering and dissemination framework can be extended to handle personalized subscriptions with dynamical thresholds;
- using the filter information, an adaptive approach can tune the default threshold and select a smaller number of candidate terms to significantly reduce the forwarding cost;
- with real query logs and document corpus, our experimental results verify our analytical findings.

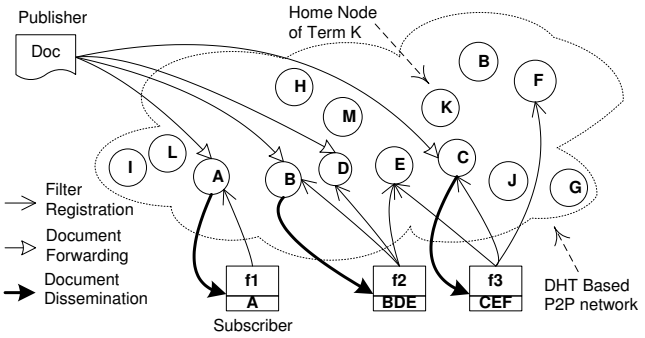


Fig. 1. Basic Framework of Full Registration and Partial Forwarding

The rest of this paper is organized as follows: Section II shows the basic content filtering and dissemination framework with the system default threshold. Section III extends the thresholding approach to meet subscribers' personalized requirements. Section IV shows the adaptive forwarding. Section V evaluates the prototype implementation. Finally, Section VI is a conclusion.

II. Basic Framework

Let us first show an overview of DHT-based P2P networks and then consider the simple case in *STAIRS* where each subscription filter specifies the default threshold.

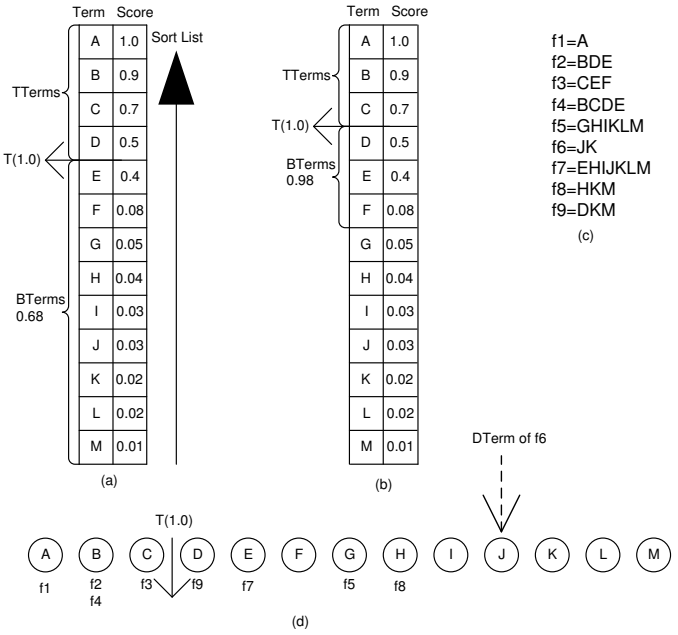
A. An Overview of DHT-based P2P networks

A number of structured peer-to-peer routing protocols have been proposed recently, including CAN [13], Chord [17], Tapestry [24] and Pastry [15]. These P2P systems provide the functionality of a scalable distributed hash-table (DHT), by reliably mapping a given object key (like a term with the string type) to a unique live node in the network. For simplicity, in this paper, we call the node that is responsible for the object key as *home node* of the key. These systems have the desirable properties of high scalability, fault tolerance and efficient routing of queries. In these DHT P2P networks, each node is regularly assigned with equal $O(\log N)$ number of links, and the average lookup hop number is guaranteed with $O(\log N)$ where N is the total number of nodes in DHT.

B. Filter Registration

Each end user subscribes his/her interested documents by a filter $f = \{t_1, ..., t_{|f|}\}$ with a threshold value $\mathrm{T}(f)$. Here we call the number of terms in f, $|f|$, as the *length* of filter f. For the basic framework, we assume that each subscriber adopts a default threshold value $\mathrm{T}(f) = \mathrm{T}$.

When a subscriber sends out the subscription request containing filter f to a DHT based P2P network, f is registered in the home node for each term in f. As a result, f is registered in $|f|$ peers. We call this "*full registration*". In Figure 1, filter $f_2 = \{B, D, E\}$ is registered in three home nodes respectively for B, D, and E; $f_3 = \{C, E, F\}$ is registered in three home nodes respectively for C, E and F.



(a) Document with ⟨term, score⟩ pairs; (b) Optimized for Short Filters;
(c) Disseminating Terms; (d) Short Filters;

Fig. 2. Bottom-up Approach to Select TTerms by Default Threshold $\mathrm{T} = 1.0$

C. Document Publication

Before the documents are published, the computation of term score $score(t_i, d)$ by $tf{*}idf$ requires the global statistical information including total number of documents and the inverse document frequency of some term. In [5], it has shown that in IR scenarios it is enough to have an approximation of the exact idf values. By a set of randomly chosen peers to collect such statistics and merge the results, [19] creates an approximation of the global idf values. Though it is an open problem to approximate idf in pub/sub scenario [20], one possible solution is to use the similar approximate solution as [19] to compute idf, and the precise approximate will be our future work.

Given document $d := \{\langle t_i, score(t_i, d)\rangle|_{i=1,...,|d|}\}$, two steps are required for document publication:

- *Document Forwarding* to forward d to the home nodes of some meaningful terms in d;
- *Document Dissemination* to disseminate d to subscribers with qualified filters f with $TotalScore(f, d) \geq \mathrm{T}$.

In the second step, a filter f satisfying $TotalScore(f, d) \geq \mathrm{T}$ is called a "*qualified filter*". In the computation of $TotalScore(f, d) = \sum_{i=1}^{|f|} score(t_i, d)$ where $t_i \in f$, term t_i need not appear in the document d. If term $t_i \in f$ is not in document d, then $score(t_i, d) = 0$. Only when term $t_i \in f$ is also in document d, t_i will contribute the value of $TotalScore(f, d)$ by the term score $score(t_i, d)$.

Now, we show how to use $TotalScore(f, d) \geq \mathrm{T}$ for document forwarding. If we sort all terms in document d by a *descending* order of $score(t_i, d)$, we can position the threshold value T in the sorted list as illustrated by the

following example. In Figure 2 (a), the term scores are sorted in descending order, the sum of term scores from M to E is 0.68, and the sum from M to D is 1.18. Then the threshold value $\mathrm{T} = 1.0$ is positioned between term D and E. Since the sum of term scores from M to E is less than the threshold value 1.0, we call the terms from M to E the"*Below Threshold Terms*" (BTerms), and the remaining terms in d (i.e. terms from D to A) the "*Threshold Terms*" (TTerms). With BTerm and TTerm, we have two important lemmas about document d and threshold T. The first lemma follows from the definition of BTerm:

Lemma 1: The sum of term scores of any subset of BTerms in document d is less than T.

For example, in Figure 2(a) with BTerms from M to E, for the full combination of BTerms, the sum of term scores, 0.68, is less than the threshold value 1.0; for a partial combination for BTerms like $\{E, F, G\}$, the sum of term scores is less than 0.68.

Lemma 2: Given a document d with $TotalScore(f, d) \geq \mathrm{T}$, the qualified filter f at least contains one TTerms of d.

Proof By contradiction: if no TTerm appears in f, f is composed of only BTerms. By Lemma 1, we find that the sum of all term scores for all terms in f is less than T, contradicting the assumption in Lemma 2. Hence the Lemma holds.■

In Figure 2, two TTerms (B, D) appear in the qualified filter f_2 with $TotalScore(f_2, d) \geq 1.0$.

1) Forwarding by TTerms: By Lemma 2, we conclude that at least one TTerms of document d appear in a *qualified* filter f. Meanwhile, from Section II-B, filter f is "*fully* registered" in the home node of each term in f. The terms in a filter f can be (i) TTerm of d, (ii) BTerm of d, or (iii) not in d. Thus "full registration" can guarantee that a qualified filter f with $TotalScore(f, d) \geq \mathrm{T}$ is certainly registered in the home node for some TTerm of d.

As a result, when document d is published, we can forward d only to the home node of each TTerm in d because the qualified filters must be registered in the home node of some TTerm. Compared with the "*full-forwarding*" approach that forwards document d to the home node of each term in d, our "*partial-forwarding*" can reduce the forwarding cost. In Figure 1, when document d in Figure 2(a) is published, d is only forwarded to 4 home nodes: for TTerm A, B,C and D, instead of 13 home nodes by "*full forwarding*": for the terms from A to M.

Algorithm 1 Selection (threshold T, document d)

Require: $|d|$ term scores $score(t_i, d)$ of document d are reverse sorted with $score(t_1, d) \geq ... \geq score(t_{|d|}, d)$;
1: $Doc\_Score(d) \leftarrow 0$;
2: **for** $i = |d|$ to 1 **do**
3: $Doc\_Score(d) \leftarrow Doc\_Score(d) + score(t_i)$
4: **if** $Doc\_Score(d) \geq \mathrm{T}$ **then**
5: **return** $t_1, ...t_i$ as selected TTerms;
6: **end if**
7: **end for**
8: **return** null;

Algorithm 1 lists the pseucode to select TTerms by threshold T for document forwarding. In the bottom-up loop (line 2-7), the sum of term scores $score(t_i, d)$ starts from the tail term with the least term score $score(t_{|d|}, d)$. Once the sum from $score(t_{|d|}, d)$ to $score(t_i, d)$ is larger than the threshold T, the remaining terms from t_1 to t_i are returned as TTerms (line 5). Finally if the sum of all term scores is smaller than T, no term is selected as a TTerm.

2) Dissemination Without Duplicates: When document d reaches the home node for some TTerm, for all filters f locally registered in the home node, $TotalScore(f, d)$ is computed to check whether the condition $TotalScore(f, d) \geq \mathrm{T}$ is met. If met, document d should be disseminated to the subscriber of f; otherwise, no dissemination takes place. Note that by "full registration", filter f will be registered in the home nodes of all terms appeared in f. It could be a problem: when f may contain multiple TTerms, there could produce a duplicate dissemination. For example in Figure 1, with two TTerms B, D, f_2 is registered in the home nodes for B and D. When d is forwarded to two home nodes for B and D, document d may be duplicately disseminated to the subscriber of f_2 by both home nodes.

To avoid duplicated dissemination, we introduce a *dissemination rule*. First let us define some terms. Given a document d, we call a term in filter f with the highest term score in d a *dissemination term* (DTerm) of f for d, denoted as $t_{f,d}$. In case of a tie in the highest score, we can break the tie by some rule such as choosing the term that appears the earliest in the document. The home node for $t_{f,d}$ is treated as the *dissemination node* to deliver d to f. With the definition of DTerm $t_{t,f}$ we give the following dissemination rule:

Dissemination Rule: *Given a qualified filter f composed of one or more TTerms, document d is disseminated to the subscriber specifying f only by the home node of the DTerm of filter f for d.*

To illustrate the dissemination rule, we may set up document d of Figure 1 with the term scores of Figure 2 (a). When d is forwarded to the home node for term B, it can be observed that filter $f_2 = \{B, D, E\}$ contains two TTerms B and D. Since the term score for B is higher, d will be disseminated to the subscriber specifying f_2 via the home node for B. Meanwhile, when d is forwarded to the home node for D, it can be similarly observed that f_2 contains two TTerms B and D. Based on the dissemination rule, the home node for D will not disseminate d to the subscriber of f_2. Similarly, document d is uniquely disseminated to the subscribers of filters f_1, f_2 and f_3 by the dissemination nodes A, B and C, respectively. In addition, Figure 2(d) shows the dissemination term of all filters from f_1 to f_9 for document d.

Note that in our dissemination rule, *no nodes are required to remember the history information about the disseminated documents*. Instead, in [8], the subscribers have to remember the history information to avoid duplicates when documents are forwarded from multiple home nodes, and the communication cost used to forward duplicated documents is wasted in vain.

D. Optimization for Shorter Filters

From two real query history logs from a popular commercial search engine and www.search.com, we find that the major input queries are typically composed of only a few terms (See Figure 6(a) in Section V). If these short queries are used as the filters in *STAIRS*, we can further improve the forwarding cost by forwarding d to home nodes of a very small number of TTerms.

Recall the definition of TTerms and BTerms in Section II-C, we actually assume that a filter is composed by *arbitrary* number of terms, and require the sum of term scores for BTerms in d to be less than the default threshold value T. With consideration that filters are composed of at most $|f|_s$ short terms, we can redefine the TTerms by the default threshold T as follows: in the bottom-up sorted list of document d, the sum of term scores for $|f|_s$ consecutive terms should be less than T; then we use the term t_h with the highest term score among such $|f|_s$ consecutive terms to indicate the term in d is TTerm or BTerm: any term in d having a higher term score than t_h is a TTerm; otherwise it is a BTerm.

For example in Figure 2(b), suppose the maximal number of terms in any filter is $|f|_s = 3$. Since the sum of term scores for 3 terms including F, E and D is only 0.98, we may indicate that any term t_i with term score smaller than 0.5 as BTerm; and the remaining terms as TTerm. Compared with Figure 1(a) where the sum of term scores for all terms from the bottom line to term E is less than $\mathrm{T} = 1.0$, now we only require the sum of term scores for $|f|_s$ consecutive terms is less than $\mathrm{T} = 1.0$. Obviously, by filtering with a small number of terms, we boost the BTerms to a higher position in the sorted list and achieve a smaller number of TTerms than Figure 2(a); forwarding d to home nodes of a smaller number of TTerms will further reduce the forwarding cost. The experiments of Section V show that this strategy can greatly reduce the number of TTerms and the forwarding cost.

Similar to Lemmas 1 and 2, we derive the following results for the new definition of BTerms and TTerms:

Lemma 3: The sum of term scores by any combination of $|f|_s$ BTerms in document d is less than T.

Lemma 4: Given a document d with $TotalScore(f, d) \geq$ T, a qualified filter f composed of at most $|f|_s$ terms contains at least one TTerm of d.

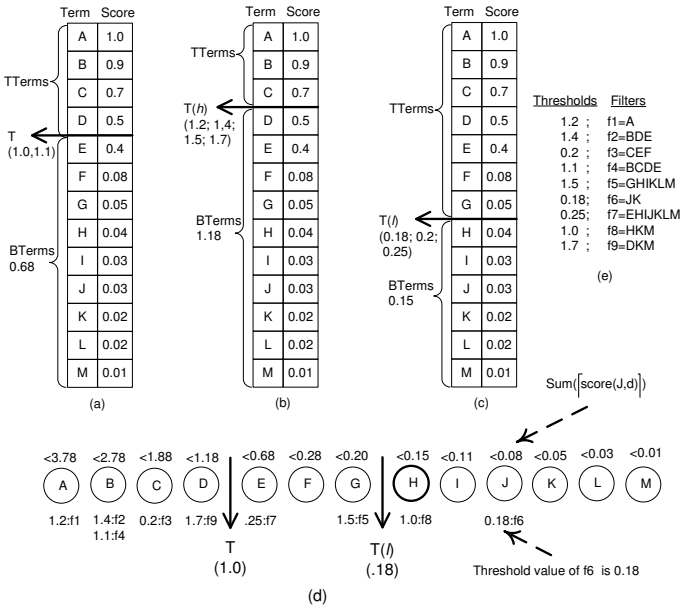
Algorithm 2 lists the pseucode to utilize the shorter filters for document forwarding. Note During the for loop from step 8 to step 13, $Doc\_Score(d)$ is re-initiated to aggregate the term score of at most $|f|$ terms. If $Doc\_Score(d) \geq$ T is met, the remaining terms from t_1 to t_j are returned as the selected TTerms.

E. Discussion

Full registration is the key of *STAIRS* to enable partial forwarding with less forwarding cost. Documents d_1 and d_2, even with the same term list, may produce different sorted lists for term scores. A term t_i may be a TTerm in d_1, but it may be a BTerm in d_2. Full registration guarantees that we need only forward each document to the home node for its TTerms, which is locally determined by the document publisher itself, without any a priori knowledge of subscriber filters.

Algorithm 2 Selection_Short (threshold T, document d, length $|f|$)

Require: $|d|$ term scores $score(t_i, d)$ of document d are reverse sorted with $score(t_1, d) \geq ... \geq score(t_{|d|}, d)$;
1: **for** $i = |d|$ to 1 **do**
2: $Doc\_Score(d) \leftarrow 0$;
3: **if** $i - |f| > 0$ **then**
4: $jUpper \leftarrow i - |f|$;
5: **else**
6: $jUpper \leftarrow 1$;
7: **end if**
8: **for** $j = i$ to $jUpper$ **do**
9: $Doc\_Score(d) \leftarrow Doc\_Score(d) + score(t_j)$
10: **if** $Doc\_Score(d) \geq$ T **then**
11: **return** $t_1, ...t_j$ as selected TTerms;
12: **end if**
13: **end for**
14: **end for**
15: **return** null;



(a) Default Threshold; (b) Higher Thresholds; (c) Lower Thresholds; (d) Guided Forwarding; (e) Personalized Filters.

Fig. 3. Personalized Subscription

III. Personalized Subscription

In this section we extend the basic framework of *STAIRS* by allowing each subscriber to specify his/her personalized threshold.

Though the system sets up a default threshold that may work well for general users, some subscribers may prefer to specify their own thresholds, which may be higher or lower than the default value T. In Figure 3(e), some subscribers use the default value 1.0, some specify higher thresholds $1.1, 1.2, 1.4, 1.7$; others specify smaller values $0.18, 0.2, 0.25, 0.8$.

1) Default Forwarding For High Thresholds: With such dynamic thresholds, the challenge in the P2P environment is that the publisher cannot know in advance the thresholds

specified by subscribers.

Given a higher threshold T_h, the set of TTerms by T_h is a *subset* of the set of TTerms by the default value T. For example, in Figure 3(b), TTerms by $\mathrm{T}_h = 1.2$, i.e. terms $\{A, B, C\}$, are subset of TTerms by $\mathrm{T} = 1.0$, i.e. terms $\{A, B, C, D\}$. As a result, forwarding document d to TTerms by the default threshold T can also satisfy the requirement of T_h.

After d is forwarded to the home node for each TTerm by T, d is matched with each locally registered subscription filter f by checking whether $TotalScore(f, d) \geq \mathrm{T}_h$ is met. If true, the document is disseminated by the dissemination rule.

2) Dynamic Forwarding For Low Thresholds: Unfortunately the set of TTerms for a lower threshold $\mathrm{T}_l < \mathrm{T}$ is a *superset* of the set of TTerms by the default value T. Then just forwarding the published document d to home nodes for TTerms by T will miss some TTerms by T_l. For example in Figure 3(c), terms $\{E, F, G\}$ are missed if d is only forwarded to home nodes for TTerms by T (i.e. terms from A to D).

Recall the dissemination rule in Section II-C.2, it states that document d is disseminated to the subscriber only via the home node of the dissemination term (DTerm) $t_{f,d}$. Given term $t_{f,d}$ having term score $score(t_{f,d}, d)$, we define $Sum(\lceil score(t_{f,d}, d)\rceil)$ as the sum of all term scores smaller than $score(t_{f,d}, d)$. In Figure 3(d) the dissemination term in filter $f_6 = \{J, K\}$ is term J with term score 0.03. The scores of terms from J to M are equal to or less than 0.03, and $Sum(\lceil score(J, d)\rceil) = (0.03 + 0.02 + 0.02 + 0.01) = 0.08$.

If f is a qualified filter of d, two prerequisites must be met:

$$\begin{cases} (1) TotalScore(f, d) \geq \mathrm{T}_l \\ (2) TotalScore(f, d) \leq Sum(\lceil score(t_{f,d}, d)\rceil) \end{cases}$$

The first prerequisite is the basic filtering requirement. The second prerequisite holds by contradiction: if $TotalScore(f, d) > Sum(\lceil score(t_{f,d}, d)\rceil)$, filter f must contain a term score larger than $score(t_{f,d})$, contradicting the definition of dissemination term. Next from (1) and (2), we get:

$$\mathrm{T}_l \leq Sum(\lceil score(t_{f,d}, d)\rceil)$$

In Figure 3(d), consider d and filter f_6 with threshold 0.18 ($t_{f_6,d}$ is term *J*). Since $\mathrm{T}(f_6) = 0.18$ and $Sum(\lceil score(t_{f_6,d}, d)\rceil) = Sum(\lceil score(J, d)\rceil) = 0.08$, we can infer that document d does not satisfy f_6, violating $\mathrm{T}_l \leq Sum(\lceil score(t_{f,d}, d)\rceil)$. For illustration, Figure 3(d) shows the value of $Sum(\lceil score(t_{f,d}, d)\rceil$ for each term when such term is used as dissemination term. Following the condition $\mathrm{T}_l \leq Sum(\lceil score(t_{f,d}, d)\rceil)$, we similarly find that d does not satisfy filter f_5, f_8 and f_9.

The above prerequisites can be used to guide the document forwarding. For any two terms t_i and t_j in document d, we can easily infer that

IF $score(t_i, d) > score(t_j, d)$,
THEN $Sum(\lceil score(t_i, d)\rceil) > Sum(\lceil score(t_j, d)\rceil)$

Suppose filter f_i and f_j specify the same threshold T_l and have the dissemination term t_i and t_j for document d. The probability for $\mathrm{T}_l < Sum(\lceil score(t_i, d)\rceil)$ will be higher than that of $\mathrm{T}_l < Sum(\lceil score(t_j, d)\rceil)$. It means that d satisfies filter f_i with a higher probability than f_j. Therefore we consider forwarding d to the home node for t_i with a higher probability than forwarding d to home node for t_j. Let us define the *coverage* of forwarding d to the home node for BTerm t_k (with respect to T), denoted by $P_{t_k,d}$, as:

$$\mathrm{P}_{t_k,d} = \frac{Sum(\lceil score(t_k, d)\rceil)}{\sum_{x=1}^{|d_{bt}|} Sum(\lceil score(t_x, d)\rceil)}$$

In the above equation t_k is a BTerm by the default threshold T and $|d_{bt}|$ is the number of such BTerms. In Figure 3, by the above equation, the coveraged P_{t_k} to forward d to the home noded for $E, F, ..., M$ respectively are 37.68%, 27.71%,18.74%,...,0.099%.

Among all BTerms, we chose b BTerms, $t_1, ...t_b$, with top values of $\mathrm{P}_{t_i,d}$ where b is the smallest number to make sure that $\sum_{i=1}^{b} \mathrm{P}_{t_i,d} \geq \mathbf{P}$, where **P** is a system threshold. The value of b will be minimized for reducing the forwarding cost. If **P** is set to 1, then d is forwarded to the home nodes for all BTerms and no filters will be missed, or the total coverage will be 100%. In Figure 3(c), given $\mathbf{P} = 90\%$, term E, F, G will be chosen since $\mathbf{P}_E + \mathbf{P}_F + \mathbf{P}_G = 90.93\%$.

For convenience, we call these chosen BTerms as determined by the given probability **P** the PTerms. As a result, with dynamic thresholds, document d will be forwarded to home nodes for TTerms and PTerms. The union of the TTerms and PTerms form the PTTerms.

Discussion: In a DHT like P2P environment, it is not easy to setup a reasonable value of **P**: with a small value of **P**, many documents are missed if most subscribers specify low filter thresholds; with a larger value of **P**, much forwarding may be wasted if most subscribers specify high filter thresholds. To tackle this problem, in Section IV, we shall propose an adaptive forwarding technique to make sure the false miss dismissal rate and document forwarding cost are further reduced.

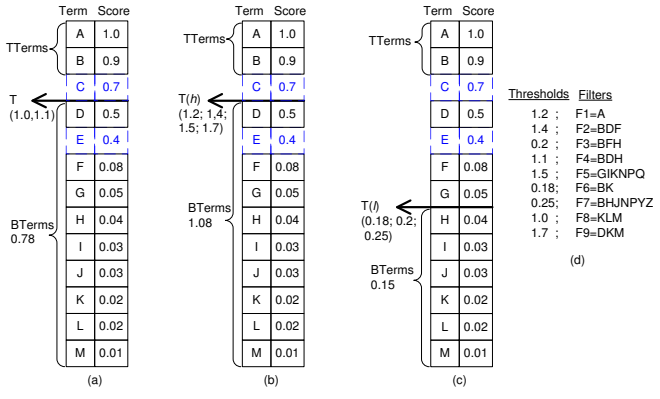
IV. Adaptive Forwarding

So far, dynamic forwarding only relies on the default threshold and the value of P to select PTTerms, in this section we show that this may result in high forwarding cost. We then describe how *STAIRS* can utilize some summarized filter information to further reduce the forwarding cost.

A. Problems with the lack of user filter information

Without knowing the user filter thresholds and filter terms, the forwarding cost of dynamic forwarding may be high caused by selecting many TTerms, which is illustrated by the following scenarios:

- There can be problems when the default threshold is set either too high or too low compared to thresholds that users prefer. If the default threshold is too low, there will a lot of unwanted document forwarding since the TTerms based on T is excessive; if it is too high, false dismissal will become a problem.
- Not all terms in a document d must appear in the registered filters. For example in Figure 4 (a) terms C and E do not appear in 9 filters f_1, ..., f_9. Since E



(a) Default Threshold; (b) Higher Thresholds;
(c) Lower Thresholds; (d) Personalized Filters.

Fig. 4. Adaptive Forwarding

is missing in these filters, we skip the term score of E during the BTerm selection and compute the sum of term scores from E to M excluding term E as 0.78, less than $\mathrm{T} = 1.0$. Thus, three remaining terms A, B and C are chosen as TTerms. Moreover, since term C is also missing in the filters, we only select A and B as TTerms, discarding C.

- Suppose in Figure 4 (d), we setup the filter length $|f|_s = 6$ to optimize the forwarding cost for short filters. However, not all terms in a filter containing 6 or more terms must appear in the published document. For example in Figure 4(d) filters f_5 and f_7 contain at least 6 terms each, and for f_5, only 3 terms (G, I and K) appear in document d; for f_7, only 3 terms (B, H and J) appear in d. Thus we may safely reset $|f|_s = 3$ for the document to generate less TTerms than with $|f|_s = 6$.
- A filter with a low threshold T_l may contain a TTerm. In Figure 4 (c) the 3 filters f_3, f_6 and f_7 have thresholds T_l smaller than the default $\mathrm{T} = 1.0$, we may select 6 terms from H to M as BTerms and let the 7 remaining terms (from A to G) be TTerms. However, we notice that B, as a TTerm, appears in the above 3 filters. Thus in the home node of B, f_3, f_6 and f_7 are matched with document d. It means that using T_l to select TTerms is too conservative when there are TTerms in these filters. Using $\mathrm{T} = 1.0$ to select TTerm B can also have document d disseminated to subscribers of filters f_3, f_6 and f_7.

As a summary, not knowing the filter information (thresholds, filter lengths and query terms) may create either a high false dismissal rate or a high forwarding cost. In the next sections, we show how to select a few number of TTerms to reduce the forwarding cost by using a proposed structure to summarize the filter information.

B. Selecting a smaller number of TTerms by HiBlooms

To summarize the filter information, we use a hybrid structure of histogram and bloom filters, named HiBlooms. The histogram is used to capture the distribution of filter thresholds, and the bloom filter is to encode the query terms in those filters whose thresholds fall in the cell (or bucket) range of the histogram. In details, suppose the equi-width histogram maintains b buckets or cells with range $[lb_i, ub_i)$, those filters with threshold inside $[lb_i, ub_i)$ are further grouped by filter length, and all query terms in each group are encoded by *one* compact synopsis, represented by a Bloom filter. Thus, the whole HiBlooms structure contains $b * g$ bloom filters where b and g is the number of buckets and grouped filter length, respectively.

Algorithm 3 Selection_BF (bloom filter bf, threshold T, length $|f|$, document d)

Require: $|d|$ term scores $score(t_i, d)$ of document d are reverse sorted with $score(t_1, d) \geq ... \geq score(t_{|d|}, d)$;
1: $d' \leftarrow$ null;
2: **for** $i = 1$ to $|d|$ **do**
3: **if** $bf.lookup(t_i)$ is true **then**
4: add $\langle t_i, score(t_i, d)\rangle$ to d'
5: **end if**
6: **end for**
7: **return** Selection_Short(T,$|f|$, d');

After receiving the request to publish document d, the node selects the TTerms via the HiBlooms structure as follows: for each bloom filters associated with a grouped filter length $|f|_g$ and a bucket with threshold range $[lb_i, ub_i)$, we follow the approach in Section II-D to select the TTerms by the filter length $|f|_g$ and a threshold T (we consider the lower bound lb_i as the threshold). During the selection of TTerms, we check whether the term t_i in d appear in the bloom filter: if so, the term score $score(t_i)$ will contribute to the calculation of $TotalScore(f, d)$, otherwise we directly consider such terms as BTerms. With the HiBlooms structure, the number of selected TTerms can be significantly reduced, correspondingly giving a lower forwarding cost. Moreover, the membership checking of the bloom filter does not produce false negatives.

Algorithm 3 shows the pseucode to select TTerms by the bloom filter, threshold T and length $|f|$ to select TTerms. As shown before, each bloom filter bf in the HiBlooms structure is associated with a grouped filter length $|f|$ and a threshold T (i.e. the lower bound lb_i). First (line 1-6), we initiate a document d' with null, then add $\langle t_i, score(t_i, d)\rangle$ to d' where $t_i \in d$ appear in the bloom filter. Since $score(t_i, d)$ is reverse sorted in the original document d, $score(t_i, d')$ is also reverse sorted. After that, we call function select_short in Algorithm 2 to select TTerms.

C. Maintaining HiBlooms structure

Suppose each bloom filter in the HiBlooms structure uses k hash functions and allows m bits, by the following Equation [7]:

$$\left(\frac{1}{2}\right)^k \approx (0.6185)^{m/n} \quad (1)$$

We may find the value of n that minimizes the false positive rate produced by the bloom filters, where n is the number of terms that are encoded by the bloom filter. With b buckets and g groups, the HiBlooms structure can summarize a total

number $n*b*g$ terms. However, when the number of encoded terms in the bloom filter is much larger than the value of $n*b*g$, the overall false positive rate of the HiBlooms structure will be high.

Recall the approach in Section II-D to select TTerms, we notice that the input parameters with *low filter thresholds* and *long filter lengths* may select *more* TTerms and result in a *higher* forwarding cost. Thus, given some threshold τ and filter length ℓ, the HiBlooms structure assigns more bloom filters to *precisely* summarize those filters f with $\mathrm{T}(f) \leq \tau$ and $|f| \geq \ell$; and relax the strict summary for the remaining filters. More specifically, when a filter f with $\mathrm{T}(f) \leq \tau$ and $|f| \geq \ell$ is encoded by some bloom filter, we need to check whether the number of query terms encoded by such a bloom filter is larger than the value of n in Equation 1. If true, an extra bloom filter is created in order to precisely encode more query terms. On the other hand, the filters f with either $\mathrm{T}(f) \geq \tau$ or $|f| \leq \ell$ are always encoded to a single bloom filter, whether the number of encoded query terms by such bloom filter is more than n or not.

To estimate the value of τ and ℓ, we first use an equi-width histogram to collect the statistic information of *all* filters (but with no query term information). In the equi-width histogram, the filters with thresholds falling within a bucket range is grouped by the filter length, and the number of filters for each group is collected to show the overall filter distribution (threshold value and filter length). When such kind of histogram is ready, some threshold value τ and some filter length ℓ can be approximated, so that the total number of query terms in those filters with a lower threshold than τ and a longer filter length than ℓ is less than the value of $n*b*g$ where n is computed by Equation 1.

We leverage the scalable aggregation tree [21] to collect the overall histogram structure for the threshold $\mathrm{T}(f)$ and filter length $|f|$ to approximate τ and ℓ. Each node in the aggregation tree can propagate the local equi-width histogram (which is used to collect local filter information) up to their parents. The parents merge the received equi-width histograms with their own local equi-width histogram and continue the propagation until the root builds the overall histogram, and then propagate the histogram across the whole aggregation tree until each node receives the histogram. By the value of ℓ and τ, each node summarizes the filter information by the HiBlooms structure. Then the local HiBlooms structure of each node is propagated and merged by bit union operations across the scalable aggregation tree, and propagated down so that all nodes receive the overall HiBlooms structure.

V. Experiments

A. Methodology

We evaluate *STAIRS* in an DHT environment by using FreePastry [1] as the simulator with total 10,000 nodes, each filter is registered to the home node of each term in the filter and each document is forwarded to the home node of each chosen candidate TTerm in the document for dissemination.

(1) Subscription Filters

[1]http://www.freepastry.org

Query Parameter	Search.com	Commercial SE
Number of queries	81,497	4,000,000
Average number of terms per query	2.085	2.843
Maximal number of terms per query	11	29
Minimal number of terms per query	1	1

TABLE I
Statistics of Two Query Logs

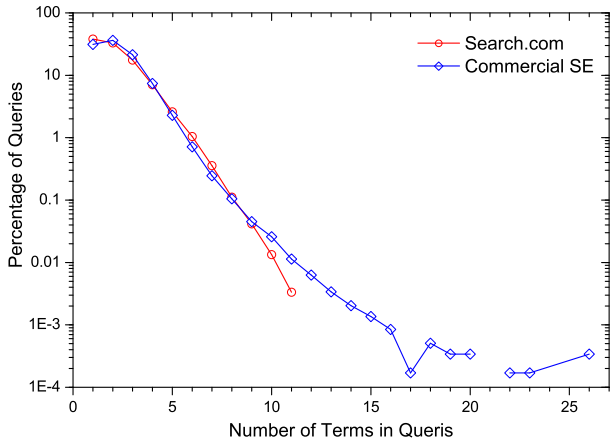


Fig. 5. Number of Terms in Query Logs

The number of queries provided by U.S. National Institute of Standards and Technology (NIST) for the TREC WT10g and TREC AP web test collection are far from enough to be used as the subscription filters in the experiments. Instead, we use two real query logs: (1) a trace log with 81.3 MB input query history file collected within four months from a popular commercial search engine (in short "commercial SE"), which is quite representative for real world systems; (2) a similar query history file from www.search.com with size 1.27MB. Table I summaries the parameters of both query logs. On average, the number of terms per query is 2.085 in search.com and 2.843 in the commercial SE; the largest number of terms in search.com is 11, and that in the commercial SE is 29 terms. The accumulative percentage of filters composed of less than 3 terms in search.com and the commercial SE's query logs is 89.17% and 88.77%, respectively; that of less than 5 terms is 98.84% and 98.42%. Figure 5 plots the number of terms of the two query logs, where the x-axis represents the number of terms, and the y-axis represents the percentage of filters with the corresponding number of terms.

From Table I and Figure 5 it is clear that the input queries in the two query log files are typically composed of very few terms. Hence, if these input queries are used as the subscription filters, there is opportunity for our proposed approach in Section II-D and Section III.

(2) Content Document

We use two data sets:

- one based on Text Retrieval Conference (TREC) WT10G web corpus, a large test set widely used in web retrieval research. The dataset contains around 10 gigabyte, 1.69 million web page documents and a set of queries (we

Document Parameter	Trec WT10g	Trec AP
Total Number of Docs	1,692,096	1,050
Average Number of Terms per Doc	64.808	6,054.9
Maximal Number of Terms per Doc	331	7,320
Minimal Number of Terms per Doc	2	1,303
Total Number of Terms	16,382	48,788
Average Term Frequency	130.31	619.48
Maximal Term Frequency	210,089	1,050
Minimal Term Frequency	1	1

TABLE II
STATISTICS OF THE TREC DATA SETS

mean the "title" field of a TREC topic as a query). The WT10g data was divided into 11,680 collections based on document URLs. Each collection on average has 144 documents with the smallest one having only 5 documents. The average size of each document is 5.91KB. The data set was stemmed with the Porter algorithm and common stop words such as "the", "and", etc. were removed from the data set.

- one based on TREC AP: a text categorization task based on the Associated Press articles used in the NIST TREC evaluations. Compared with TREC WT10G data set, the TRACE AP data set is composed of fewer (only 1,050) articles but with a larger number of terms, on average 6054.9 per article. Table II summarizes the statistics for the test data sets.

By formula $score(t_{i,d}) = \frac{freq_{i,d}}{Max_l(freq_{i,d})} \cdot \log \frac{N}{n_i}$, we compute the score of each term in both data sets. In this formula, $\frac{freq_{i,d}}{Max_l(freq_{i,d})}$ represents the value of term frequencies (tf) and $\log \frac{N}{n_i}$ the inverse document frequencies (idf). In details, $freq_{i,d}$ is the term t_i frequency in document d, and $Max_l(freq_{i,d})$ is the maximal term frequency in document d; n_i is the number of documents containing t_i across the whole data set, and N is the total number of documents.

We plot the term scores of 4 sampled documents in TREC AP and TREC WT10G,respectively, in Figure 6 (a) and (b). From these figures, the term score of TREC WT10G is relatively larger than that of TREC AP due to the effect of $\frac{freq_{i,d}}{Max_l(freq_{i,d})}$. Clearly, the skew term score distribution in both figures is important for the proposed approach: with a given threshold, the high skew can significantly prune the terms with low scores; thus reducing the content forwarding cost.

With no real threshold values available for the subscribers' filters, we randomly generate the threshold values through (1) uniform distribution; (2) exponential distribution with a given mean value. The purpose of using exponential distribution is to study how the adaptive forwarding can adjust the system default threshold to meet the relatively low thresholds by exponential distribution. The generated thresholds are used for filters to reflect subscribers' personalized subscription.

(3) Performance Metric

We use two main metrics to measure the system performance:

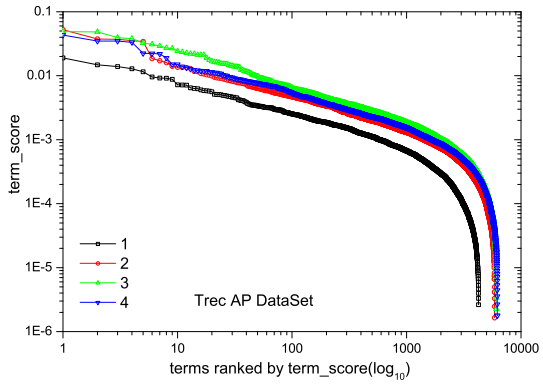
- *Forwarding cost saving rate*: the saving rate is 1.0 minus the ratio between the forwarding cost of our proposed bottom-up threshold approach and that of the full-forwarding. The higher the saving rate, the less forwarding cost is used.
- *Document false dismissal rate*: the rate is 1.0 minus the ratio between the documents that the subscribers actually receive in the filtering and disseminating system and the documents that the subscribers should receive by the specified filters. The higher the false dismissal rate, the more documents are missed.

B. Performance Results of Default Forwarding

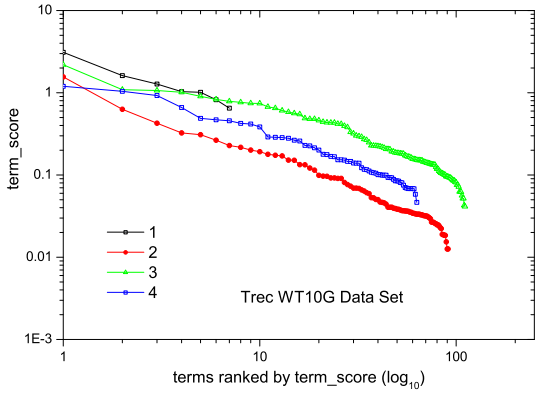
First we conduct the experiments to study the forwarding cost saving by default thresholds. In the experiments, we use TREC AP and TREC WT10G as published documents. For subscriber filters, we use each entry in search.com and the commercial SE query log as a subscription filter, and the default threshold as the filter threshold specified by the subscriber. During the document forwarding, we setup the number of terms in subscription filters as a system parameter $|f|_s$, respectively with the value of 10, 5, 3, 2 and 1. By the value of $|f|_s$, Section II-D may use the sum of $|f|_s$ consecutive terms of each published document to optimize the forwarding cost by selecting a smaller number of TTerms. Section II-C may assume filters with arbitrary terms.

Figures 7(a) and (c), with search.com query logs as filters, respectively plot the forwarding cost saving for TREC AP and TREC WT10G document corpus; and Figures 7(b) and (d) report the cost saving rates with a commercial SE trace logs as filters, for the same two TREC corpus data sets. By Figure 7, we find that (1) The approach used to optimize the short filters can save the forwarding cost. For example in Figure 7(a) for threshold value 2.0, short filters with 10 terms may achieve 4.26 folds of the forwarding cost saving compared to the arbitrary filters; for threshold 0.1, short filters with 1 term may achieve 7.96 folds of the forwarding cost saving compared to the arbitrary filters. (2) When the default threshold value in the x-axis grows larger, the forwarding cost saving rate of y-axis also increases. This is because when the larger default threshold value is used, a smaller number of TTerms are selected by the higher default threshold value. (3) Since documents in TREC WT10G are relatively short articles composed by a smaller number of terms than TREC AP, for arbitrary number of terms, the cost saving of TREC WG10G is higher than that of TREC AP.

Simultaneously, in the above experiments, we measure the false dismissal rate. Using the search.com query logs as filters, Figure 8(a) and (c) respectively plots the false dismissal rate for TREC AP corpus and TREC WT10g corpus, and Figure 8(b) and (d) for both corpus with a commercial SE's query logs as filters. From Figure 8, we find that: (1) when setting $|f|_s = 10$ or $|f|_s = 5$, the technique used to optimize short filters produce a very low false dismissal rate; even in Figure 8(c) and (d), when the default threshold value is unreasonably set with a large value 8.0, the false dismissal rate is less than 0.0812. This result is useful: it indicates that with larger threshold values, the system can reduce the forwarding cost with a small false dismissal rate as confirmed in Figure 7.

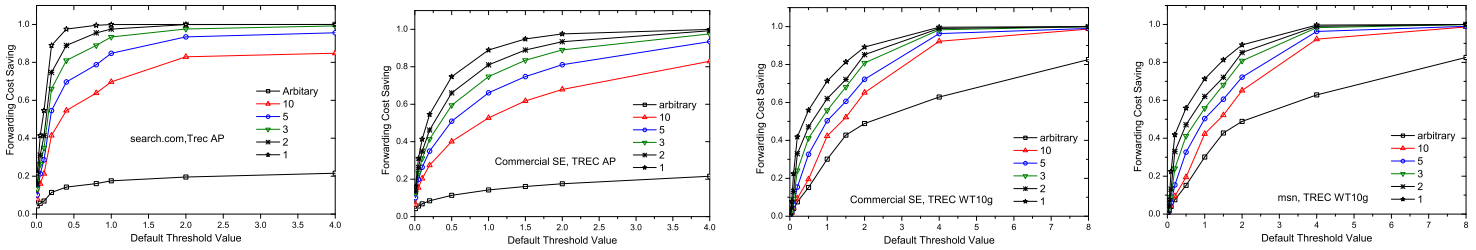


(a) TREC AP



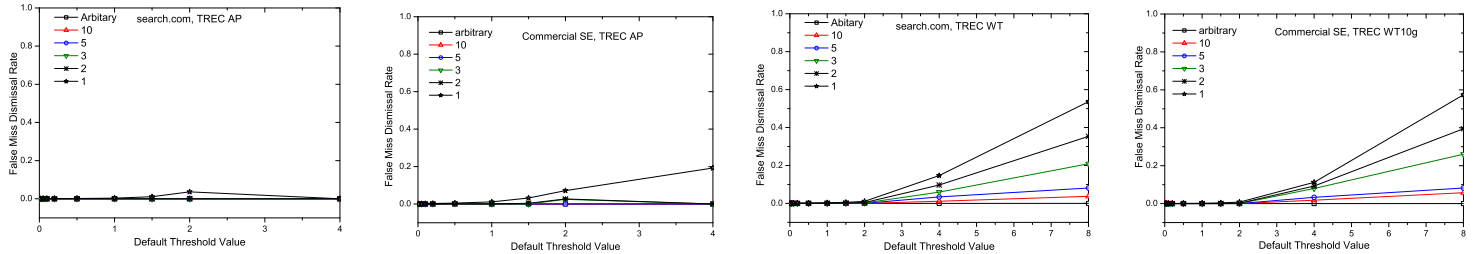
(b) TREC WT10G

Fig. 6. Query Logs and Content Documents



(a) Search.com,TREC AP (b) Commercial SE,TREC AP (c) Search.com,TREC WT10G (d) Commercial SE,TREC WT10G

Fig. 7. Forwarding Cost Saving by Default Thresholds



(a) Search.com,TREC AP (b) Commercial SE,TREC AP (c) Search.com,TREC WT10G (d) Commercial SE,TREC WT10G

Fig. 8. False dismissal rate by Default Thresholds

(2) a relatively higher false dismissal rate is produced when the number of short terms $|f|_s$ is smaller like 3, 2 or 1.

C. Performance Result of Adaptive Forwarding

We find that the results of the various combinations of two query logs and two document corpus follow similar trends; thus in the rest of this section, we report the results by choosing one example combination (search.com as filters and TREC AP as the documents) as the experimental data and study the performance of *STAIR* under several key parameters. Also, we focus on the study of adaptive forwarding as the most effective solution.

The experimental results of default forwarding indicate that the forwarding cost heavily depends on the specified thresholds (either the default threshold or personalized threshold). When the threshold is low, the forwarding cost could be high. Here we conduct the experiment to study the performance of adaptive forwarding that uses the filter information to reduce the forwarding cost.

In this experiment, the thresholds are generated by the exponential distribution with mean equal to 0.1, and the query logs in search.com are used as filters and TREC AP as the documents. During the implementation of HiBlooms structure for adaptive forwarding, we set $k = 4$ hash functions, $m = 512 * 1024 * 16$ bits of bloom vector and $c = 50$ cells (buckets). In each cell, the encoded terms are divided by 5 groups respectively for filters with filter length equal 1, 2, 3, 4 and more than 5 (including 5). As a result, with $33,554$ bits, each group of one bucket in the HiBlooms structure can encode around $4,194$ terms and the total number of encoded terms is $1,048,500$ in the HiBlooms structure, while the false positive rate is roughly equal to 0.024 [7].

Figure 9 plots the forwarding cost saving rate and false dismissal rate, respectively. By setting the accumulative rate $\mathrm{P} = 80\%$, the dynamical forwarding can reduce the false dismissal rate but with the cost of the reduced forwarding cost saving). Compared with the dynamical forwarding with $\mathrm{P} = 10\%$, the forwarding cost saving is a little reduced from

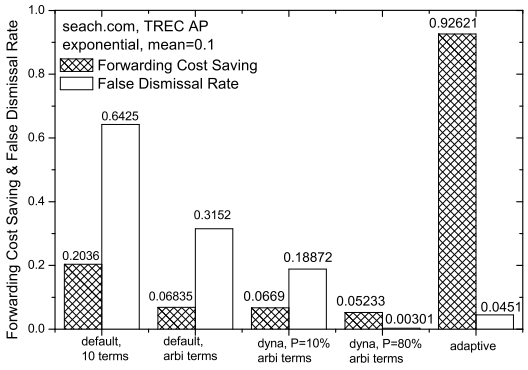


Fig. 9. Performance Study of Default, Dynamical and Adaptive Forwarding

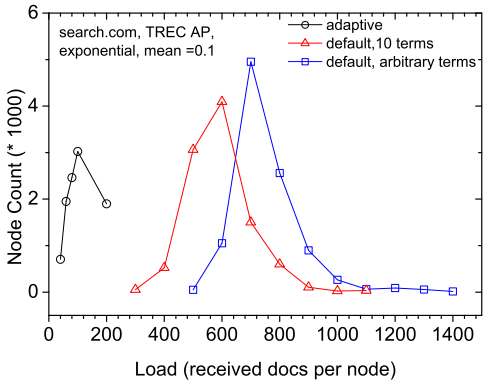


Fig. 10. Load Distribution

0.06835 to 0.0669, yet the the false dismissal rate is reduced from 0.3152 to 0.18872; for the default forwarding, that could be treated as a dynamical forwarding with P $= 0$, the forwarding cost saving for the case of filter length with arbitrary terms, is just reduced to 0.05233, but the false dismissal rate is greatly reduced to 0.00301. This is because: when the term scores in published documents are sorted by the descending order, especially for term score with a skew distribution, the head part of sorted term scores may accumulate a much higher score value than the tail part. Thus with a given cumulative probability P, a few number of terms in the head part with high term score values are selected as PTTerms, correspondingly with a low forwarding cost saving and a small false dismissal rate. More experiments by setting a larger P $> 80\%$ show that the false dismissal rate approaches 0.0 with almost no false forwarding saving.

Adaptive forwarding of Figure 9 utilizes the HiBlooms structure to summarize the filter information and achieves the highest forwarding cost saving rate 0.92629 and the lowest false dismissal rate 0.0449; instead, without the available filter information, either the default forwarding or the dynamic forwarding can consume more forwarding cost (i.e. the small forwarding cost saving rate) and the low false dismissal rate. From this experiment, we find both the document information (i.e. document terms and term scores) and the filter information (i.e. query terms, filter thresholds and filter lengths) are critical to further reduce the forwarding cost and the false dismissal rate.

Figure 10 plots the load distribution of the above experiment, where x-axis represents the load range from one point value to next one (for example from 100 to 200), and y-axis represents the node count inside the load range of x-axis. For comparison, we plot the load distribution of default forwarding with arbitrary terms, default forwarding with $|f|_s = 10$, and the adaptive forwarding.

Among the load distribution, we are interested in:

- the overall average load per node;
- the overloading issue: we consider one node is overloaded when the node may receive more than 2 times of the average load.

By default forwarding for arbitrary terms, the average load is equal to 664 documents per node, and there are 1.58 % nodes (i.e. 158 nodes) receiving more than 1200 documents; when setting $|f|_s = 10$ for the default forwarding, the average load is 581 documents per node, with only 0.35% nodes (35 nodes) receiving more than 1000 documents. By the adaptive forwarding, the average load is 132 document per node, and no node receives more than 264 documents.

These results are explained as follows: in the document corpus, some terms frequently appear in the documents and some terms rarely appear. Thus, the frequent terms produce high values of inverse document frequency (idf), and correspondingly small values of term scores. By the default forwarding with arbitrary terms to exclude the terms with the small term scores, these frequent terms may be easily selected as TTerms and documents are forwarded to the home nodes for these frequent terms, suffering from the overloading problem. By using the adaptive forwarding, these frequent terms with small term scores can be excluded to receive the forwarded document and be free of the overloading issue. Of course, without excluded any terms, the full-forwarding suffer from the most serious overloading because the home nodes of all frequent terms receive the documents.

D. Sensitivity Study of Adaptive Forwarding

Following the above experiment of adaptive forwarding in Section V-C, we vary the values of total number of nodes N and several key parameters in the HiBlooms structure (including group number g, buckets b, and bits in each bloom filter) to study the sensitivity of adaptive forwarding.

Besides the forwarding cost saving and false dismissal rate, Table III shows the average matching time per document and actual data size of the HiBlooms structure. From this table, we find that the DHT network size N has little effect on the performance result because the filter information is independent of the DHT node size. For the other three parameters, the larger grouping filter length with $g = 20$ may produce better performance for the four metrics, but with rather limited improvement. This result is not hard to explain: $g = 5$ of the default setting in Section V-C may cover most filters because filters in the realistic trace log are typically composed by a few number of terms. More cells with $b = 1000$ in the HiBlooms structure can divide the filter thresholds more precisely and

Parameters	Forwarding Cost Saving	False Dismissal Rate	Avg Matching Time Per Doc (ms)	HiBlooms Size (MB)
default:	0.92629	0.0449	22.69	2.198
case 1: N=80k	0.93152	0.0623	23.86	2.253
case 2: g=20	0.94186	0.0533	26.32	2.576
case 3: b=1000	0.99282	0.0225	71.65	5.971
case 4: m=2^{16}	0.98664	0.0521	10.12	33.155

TABLE III
SENSITIVITY STUDIES OF ADAPTIVE FORWARDING

further save forwarding cost with 0.99282 and reduce the dismissal rate with 0.0225; however, more bloom filters due to a larger value of $b = 1000$ result in the increased matching time with 71.65 ms per document and the whole HiBlooms structure becomes larger with 5.971 MB. Finally, setting more bits for each bloom filter with $m = 2^{16}$ may use less matching time than case 3 with larger cells but the whole HiBlooms structure is as large as 33.155 MB.

VI. CONCLUSION AND FUTURE WORK

In this paper we propose a novel and simple content matching and dissemination framework in distributed hash table, *STAIRS*. Different from the previous alternative solution like "*full registration and full forwarding*", or "*single registration and full forwarding*", *STAIRS* can significantly reduce the content forwarding cost by the proposed "*full registration and partial forwarding*", and subscribers can receive the satisfying contents with no duplicates. As for future work, we expect that *STAIRS* will be integrated with the existing RSS feed approach [16] to provide more expressive content filtering.

ACKNOWLEDGMENT

The research of W. Rao and A. Fu are supported in part by the RGC Earmarked Research Grant of HKSAR CUHK 4118/06E. The research of L. Chen is supported by Hong Kong RGC GRF 611608 and NSFC key project NO. 60736013. The research of H. Chen is supported by National Science Foundation of China (NSFC) under grant No.60433040 and National 973 Key Basic Research Program under grant No.2003CB317003.

We also would like to thank the anonymous ICDE reviewers for their valuable comments that helped improve this paper.

REFERENCES

[1] http://alert.live.com.
[2] http://en.wikipedia.org/wiki/special:statistics.
[3] http://www.google.com/alerts.
[4] J. P. Callan. Document filtering with inference networks. In *SIGIR*, pages 262–269, 1996.
[5] F. M. Cuenca-Acuna and T. D. Nguyen. Text-based content search and retrieval in ad-hoc p2p communities. In *NETWORKING Workshops*, pages 220–234, 2002.
[6] F. Fabret, H.-A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe. In *SIGMOD Conference*, pages 115–126, 2001.
[7] L. Fan, P. Cao, J. M. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, 2000.
[8] S. Ganguly, S. Bhatnagar, A. Saxena, R. Izmailov, and S. Banerjee. A fast content-based data distribution infrastructure. In *INFOCOM*, 2006.
[9] D. Kukulenz and A. Ntoulas. Answering bounded continuous search queries in the world wide web. In *WWW*, pages 551–560, 2007.
[10] S. Michel, P. Triantafillou, and G. Weikum. Klee: A framework for distributed top-k query algorithms. In *VLDB*, pages 637–648, 2005.
[11] T. Milo, T. Zur, and E. Verbin. Boosting topic-based publish-subscribe systems with dynamic clustering. In *SIGMOD Conference*, pages 749–760, 2007.
[12] V. Ramasubramanian, R. Peterson, and E. G. Sirer. Corona: A high performance publish-subscribe system for the world wide web. In *NSDI*, 2006.
[13] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, 2001.
[14] I. Rose, R. Murty, P. R. Pietzuch, J. Ledlie, M. Roussopoulos, and M. Welsh. Cobra: Content-based filtering and aggregation of blogs and rss feeds. In *NSDI*, 2007.
[15] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.
[16] D. Sandler, A. Mislove, A. Post, and P. Druschel. Feedtree: Sharing web micronews with peer-to-peer event notification. In *IPTPS*, pages 141–151, 2005.
[17] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, 2001.
[18] C. Tang and S. Dwarkadas. Hybrid global-local indexing for efficient peer-to-peer information retrieval. In *NSDI*, pages 211–224, 2004.
[19] C. Tang, Z. Xu, and M. Mahalingam. psearch: Information retrieval in structured overlays. In *HotNets-I*, 2002.
[20] C. Tryfonopoulos, S. Idreos, and M. Koubarakis. Publish/subscribe functionality in ir environments using structured overlay networks. In *SIGIR*, pages 322–329, 2005.
[21] P. Yalagandula and M. Dahlin. A scalable distributed information management system. In *SIGCOMM*, pages 379–390, 2004.
[22] T. W. Yan and H. Garcia-Molina. The sift information dissemination system. *ACM Trans. Database Syst.*, 24(4):529–565, 1999.
[23] Y. Yang, R. Dunlap, M. Rexroad, and B. F. Cooper. Performance of full text search in structured and unstructured peer-to-peer systems. In *INFOCOM*, 2006.
[24] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: a fault-tolerant wide-area application infrastructure. volume 32, 2002.