International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)

International Conference on Information Systems and Computing (ICISC-2013), INDIA.

EFFECTIVE MULTI KEYWORD SEARCH OVER P2P NETWORK USING OPTIMIZED BLOOM FILTER SETTINGS

G.Jayalakshmi[1], M.Vijayalakshmi[2]

[1]*Student, Department of CSE, Velammal Engineering College, Chennai, India*
[2]*Assistant professor, Department of CSE, Velammal Engineering College, Chennai, India*

Email: jayalakshmivec@gmail.com/viji13s@gmail.com

Abstract

P2P network is a popular technology used for sharing and searching files on the computers connected to the network. Current search mechanisms of peer-to-peer (P2P) systems can well handle a single keyword search problem. Other than single keyword search, multi keyword search is very popular and useful in many file sharing applications. For multi keyword search, the solution which merges the result of each keyword search incurs a lot of data traffic across wide area network .Existing methods use bloom filter (BF), an efficient data structure which is effective in reducing the traffic cost. Though applying BF is not complex, getting optimal results in terms of communication cost is not trivial. In this paper, optimal settings of bloom filter in terms of traffic cost is achieved by the global statistical information of the keywords involved, not by minimizing the false positive rate as claimed by previous studies. We also propose an intersection order optimization strategy based on BF for AND queries to estimate the size of intersection between the sets to minimize the search latency.

***Index terms*— Bloom filter, DHT, multi keyword search, P2P**

I. INTRODUCTION

With increased popularity of peer-to-peer (P2P) [1] file sharing applications such as Gnutella2 and Napster [2], it has become one of the best medium for sharing files and for searching desired data among peers [1]. It is a well known network for sharing and maintaining information on the peers than in a centralized repository.

In previous study [3], Existing p2p search engines are based on a distributed hash table (DHT) which maintains every individual keyword and maps those keywords with the documents across the network containing those keywords. Single keyword search is easy to perform as it simply uses the keyword based index to retrieve the documents, where each keyword in a query is retrieved using DHT lookups. Other than single keyword search multi keyword search is more useful and popular in real time applications. Existing methodology i.e., Multi keyword using DHT lookups performs multi keyword based search by simply merging the results of each single keyword search causing large amount of data traffic across the wide area network. For example, consider a two keyword query "cloud computing". This two keyword query is decomposed in to "cloud" and "computing". Then each of the keyword will perform a single keyword search i.e., both the keywords are searched separately. Then the results of both the keywords are retrieved separately merged together to retrieve the results containing both the keywords.

It's done by performing the distributed intersection/union operations which are considered as the result of multi keyword searching. This approach leads to unacceptable data traffic across network[1].

Current works on multi keyword search [4] focus on decreasing the number of messages across peers during searching process. Bhattacharjee et al [4] proposed an orthogonal technique using result caching to avoid data movement. But this method handles searching through inverted index of intersection operations, which will generate extremely high network traffic. To reduce such network traffic, Reynolds et al [7] used techniques namely bloom filters, caches, incremental results.

Bloom Filter (BF) plays an important role in reducing network traffic in terms of multi keyword search compared to previous techniques. A BF is an efficient data structure to represent a set S, which can handle well queries such as "is the element x in set S" [1]. By sending a bloom filter i.e., an encoded document set, rather than raw document sets among each participating peers helps in reducing the communication cost effectively. Using BF is not a difficult task but achieving optimal results in terms of communication cost is not trivial. Using BF with the goal of minimizing false positive rate will raise even more traffic cost [5].

In this paper, we focus on achieving optimal setting of Bloom Filter in terms of communication cost.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

We propose global statistical information of the keywords can reduce the communication cost, not the minimized false positive rates as claimed by previous studies [7]. Also intersection order between sets is more important for multi keyword search hence we design an optimal order strategies for queries with AND an OR operators based on BF settings.

By performing simulations on TRECWT10G [8], to evaluate the performance of this design, we can show that our design can reduce the search traffic of existing approaches by 73% [1].

This paper contains the following sections. Section 2 discusses the related work. Section 3 provides an overview of system design. Section 4 describes the optimization strategy design for intersection operation. Section 5 describes the how to gather global statistical information of the keywords. Section 6 discusses the virtual host technique. Section 7 briefed the scalability issues. Section 8 discusses the real time application YaCy and section 9 concludes the paper.

II. Literature Survey

One of the major issues in decentralized P2P search engines is how to effectively search and retrieve the desired results. There are two types of searching techniques exists: a full text federated search over unstructured P2P networks [9] and DHT-based search over structured P2P networks.

A full text federated search in peer to peer network is implemented in [9]. Federated search engines organizes peer in an ad hoc fashion and a searching method called flooding is used. Each query is tagged with Time to Live (TTL) field in order to limit the number of hops it travels. Methodologies used in this type of search engines perform query search in two levels, the peer level and document level. First the hub which receives a query performs resource selection algorithm to rank and select the peers which are likely to have the results and forwards to them. Selected peers performs full-text document retrieval algorithm and provides list of top ranked results. A hub receives the lists of results from multiple peers and performs merging algorithm to merge them in to a single integrated list of documents and forwards to the client. This approach results in heavy network traffic and communication cost to get relevant results. Hence in PlanetP [10] a global inverted index for each peer is proposed, which contains a mapping "t→p" if term t is in the local index of peer p. PlanetP is a simple, yet powerful system for sharing information. It is powerful because it maintains a content-ranked view of the shared data.

But in practice it is very difficult to collect and store such global index information.

DHT based search engines are based on distributed indexes, and there are two types of distributed index mechanisms exists: single term based and term set based indexes. Single term based distributed index contains single term keywords mapped to the document identifier of the corresponding document in which those keywords present. It retrieves the list of documents/nodes responsible for each single keyword query based on the index.

In [11] hybrid global-local indexing technique is proposed to facilitate the retrieval of documents. In this technique frequent terms of the keywords for each document are stored in the global or local index and other keywords are replicated with the identifier of the document in the posting list. Disadvantage in this approach is few of the stored frequent terms may not be the important representative for documents and such replication strategy may incur unacceptable storage and communication cost. Multi keyword search can also be performed using the global single term based inverted index built on DHT, by looking up the index for different keywords from multiple peers across wide area network. Finally the list of documents containing all the keywords is retuned using distributed intersection operation as result. Even though only few nodes required looking on, travelling across each node with potentially large amount of data leads to heavy bandwidth cost.

Hence Bloom Filter (BF) is proposed to reduce such bandwidth cost incurred by distributed intersection operation. In previous study [7] it is claimed that minimizing the false positive rate, optimal settings of BF can be achieved. Though the communication cost is still unacceptable. Hence we show that minimizing the false positive rate will not make any impact on reduction in communication cost and it is far from achieving optimal BF settings.

Other way to deal with bandwidth cost is pre-computing the term-set-based index, implemented in [12]. This method can significantly reduce the cost and is efficient for multi keyword searching. But the major drawback of this approach is, exponentially growing index size. Podnar et al [13] proposed to index only highly discriminative keyword (HDK) to reduce such index size. But if those keywords may rarely or never used in queries, causing high consumption of bandwidth and storage. In [14] Bender et al proposed to index keyword sets that are frequently issued by users.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

In this design DHT node maintains additional posting list for keywords that are frequently searched by users in previous query logs. Although such term set indexing reduces the scalable cost, distributed intersection operation is required if the query has not searched before.

In proposed work, to reduce the bandwidth cost an effective intersection order optimization strategy in BF is implemented. To reduce search cost, global keyword information is gathered by using a push-synopsis gossip algorithm. And scalability problem can be solved effectively by combining top-k pruning techniques with bloom filter techniques.

III. System Design

In this section we discuss on the system design of P2P network for multi keyword search, and then on techniques by which the communication cost of DHT based multi keyword search can be minimized by using optimal BF settings. Then a distributed algorithm for AND queries and OR queries are discussed. We are proposing an optimization strategy for both AND, OR queries. We utilize push synopsis gossip algorithm for collecting global keyword popularity in order to minimize the communication cost. Then we present virtual host technique to deal with load balancing issues and the suggested solution to the scalability problem.

3.1 Minimization of communication cost for multi keyword search

Before discussing the mechanisms on how to minimize the communication cost for multi keyword search, the following concepts are described.

3.1.1 Multi keyword Search

By analyzing the query logs of any popular web search engine we can observe that 56 percent of the queries contain more than one keyword i.e., at least two keywords. Hence we conclude that multi keyword search is quite common in web search engines.

3.1.2 Bloom Filter

Bloom filter [1] is a probabilistic data structure which is space efficient and is used to check whether an element is present in the set or not. It is represented as bit vector with an array of m bits, initially set to 0.

It uses k different hash functions $\{h_1, h2, h_k\}$. To add an element to the set, an element is hashed using k hash functions to get k array positions and those bits are set to 1. To check whether that particular element is present in the set, the element is hashed by those k hash functions to get the array positions.

If all the bits are set to 1 then the element is a member of set or if the any of the bit is set to 0 then element is definitely not a member of set. Fig 1 presents the overview of how the bloom filter assigns the bit values to 1 or 0 for keywords.

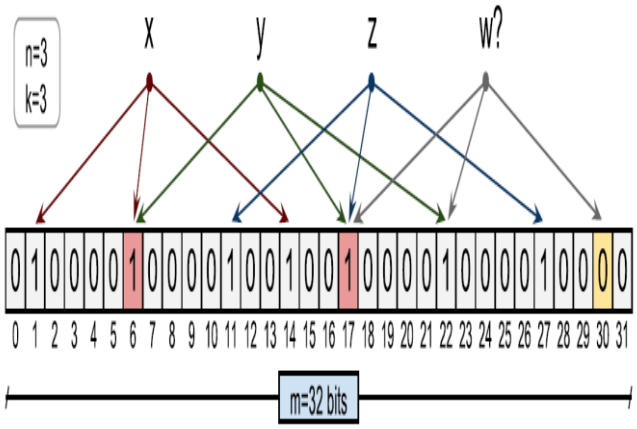

Fig. 1. Overview of Bloom Filter [15]

For each element x belonging to the set S, the bits hi(x) are set to 1 otherwise 0. Similarly for elements y, z or w. after all the n elements are hashed inserted in to the BF, the probability of specific element is set to 0 in bitvec_m is

$$p = (1 - 1/m)^{kn} \approx e^{-kn/m} \qquad (1)$$

After n elements are inserted in to the bitvec_m, the probability of false positive rate is

$$F = (1-p)^k = (1-e^{-kn/m})^k \qquad (2)$$

3.1.3 AND Query

A major solution to the multi keyword search in wide area network is to conduct distributed intersection operation. Fig1a illustrates an example of AND query based on distributed intersection. The process is as follows, the query is first routed to the DHT node which is responsible for keyword x.

Then X the set of IDs of the documents containing keyword x, is transmitted to the node which is responsible for keyword y. Now the intersection operation $X \cap Y$ for set X and set Y, which is responsible for keyword x and y are done and the results are sent to the client. Hence in DHT based P2P multi keyword search it is clearly revealed that the communication cost is extremely high while sending complete sets of document identifiers among the peers to get the small result set [1].

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

Hence it is clear that the communication cost can be minimized greatly by sending BF of sets instead of sending raw sets among peers for the distributed intersection operation [1]. Existing work [7] claimed that we can reduce the communication cost by simply minimizing the false positive rate of a BF.

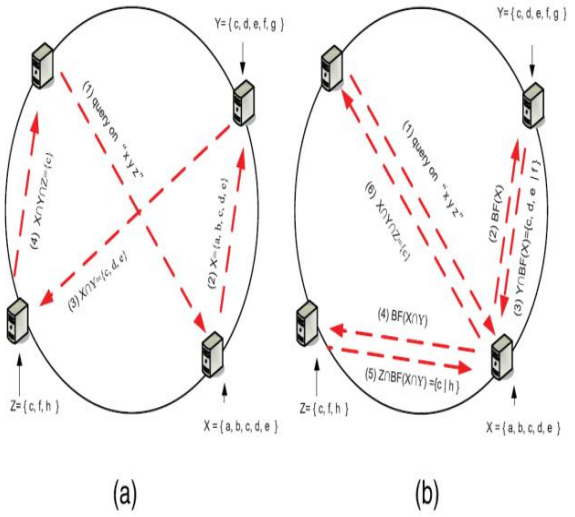

Fig.2. Actual intersection versus BF-based intersection [1].

Our design shows that communication cost can be reduced by sending the optimized BF of set X, i.e., BF(*X*) instead of sending X itself as illustrated in Fig. 1b.When BF(*X*) is sent to DHT which is responsible for keyword y, the node determines the intersection set of X and Y based on BF(X). Since there are no false negatives, the result will return all the elements of actual intersection. Due to the possibilities of false positives, the result may also contain sets that have only keyword y. Hence to achieve the exact intersection of X and Y, the result set Y ∩BF(X) is sent back to the DHT node responsible for keyword x.

By removing the false positives from the set Y∩BF(X), we achieve the exact intersection X ∩Y by calculating the X∩(Y ∩BF(X)) [1].And for large inverted lists , we may need to transfer multiple rounds of bloom filter block by block as explained in [16].

In the distributed algorithm for AND queries , sending the result (Y ∩BF(X) directly to the client instead of sending it again the node which is responsible for first keyword x. it will also reduces the communication cost further more . But this method suffer from loss in result precision due to the false positive rate of BF [1].The list of notations used in our algorithms are listed in the following table, Table 1.

Table 1
Notations used in Algorithms 1 and 2 [1].

Notation	Description
n	Number of elements inserted into a BF
m	Size of the bit vector used as a BF
k	Number of hash functions used for a BF
f	False positive rate of a BF
f_{min}	Minimized false positive rate of a BF
X	The set of the IDs of the documents that containing keyword x
$BF(X)$	The BF for set X
$Y \cap BF(X)$	The estimated intersection of sets X and Y based on BF(X) and Y
r	Number of bits each item in the posting list takes

First, the query q which contains keywords x and y is the received input. Now ensure that the system contains list of documents with those keywords using inverted list. Now the client chooses node S_x to send query q(x,y). Node S_x finds set X which contains documents containing keyword x . Using optimal BF settings, BF hashes the X using set of independent hash functions hj where j= 1 to k to transmit BF(X) to node S_y which is responsible for keyword y. now S_y identifies set Y which contains documents containing keyword y. intersection operation for set Y and BF(X) is performed to achieve the set which contains both keywords x and y. To eliminate the false positive rate gained from set X, the resultant set is again directed to node S_x to achieve exact intersection of documents and is directed to the client [1]. Algorithm 1 shows detailed process of distributed intersection for "AND" queries [1].

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

Algorithm 1 Distributed algorithm for "AND" query

Require: Query q (x and y);
Lists of postings separately containing x and y and accommodated by DHT nodes S_x and S_y in a distributed manner.

Ensure: List of documents containing both x and y

Step (1): issue query

1: Client chooses S_x to send query $q(x, y)$.

Step (2): compute BF(X)

1: S_x identifies the set X which contains all the documents containing x.
2: S_x generates a BF with optimal settings for X using hash functions $\{h_j(.), 1 \le j \le k\}$
3: S_x transmits BF(X) to S_y.

Step (3): compute $Y \cap$BF(X)

1: S_y identifies the set Y which contains all the documents containing y.
2: $Y \cap BF(X) \leftarrow \emptyset$
3: **for** $i = 1$ to $|Y|$ **do**
4: S_y checks b_i, an item of Y against BF(X) with hash functions $\{h_j(.), 1 \le j \le k\}$.
5: **if** $\forall(j)(1 \le j \le k)$,s.t.$h_j(b_i) = 1$ **then**
6: $Y \cap BF(X) \leftarrow (Y \cap BF(X)) \cup \{b_i\}$.
7: S_y transmits $Y \cap$BF(X) to S_x.

Step (4): reverse verification

1: S_x picks out the false positive.
2: $X \cap Y \leftarrow X \cap (Y \cap BF(X))$.
3: S_x sends the results $X \cap Y$ to the client.

3.1.4 OR Query

We may also require "OR" queries which results in containing any of the multi keywords in the query for some applications. Such queries are critical than AND queries because those keywords may rarely present in the system. Hence the search engine results both the "AND" and "OR" results for multi keyword query to the users [1].

In our design for "OR" queries as shown in fig 3, the query is directed to the DHT node responsible for keyword x. now BF(X) is forwarded from node responsible for keyword x to the DHT node responsible for keyword y. Here the documents which are not present in X will be picked by checking against set Y using BF(X) i.e., the set Y- BF(X) is returned to the client for consequent union operations. Our algorithm 2 shows this process of union operation for "OR" queries in detail [1].

Algorithm 2 Distributed algorithm for "OR" queries

Require: Query q (x or y);
Lists of postings separately containing x and y and accommodated by DHT nodes S_x and S_y in a distributed manner.

Ensure: The estimated set of documents containing x or y

Step (1): issue query

1: Client chooses S_x to send query q (x, y).

Step (2): return result X

1: S_x identifies the list of postings X which contains the documents containing x.
2: S_y sends X to the client.

Step (3): computing BF(X).

1: S_x creates an empty m-bits bit vector for BF(X), the bloom filter with minimized false positive, for X.
2: S_x generates a BF with optimal settings for X using hash functions $\{h_j(.), 1 \le j \le k\}$.
3: S_x transmits BF(X) to S_y.

Step (4): estimate union

1: S_y identifies the list of postings, Y, which contains the documents containing y.
2: $Y - BF(X) \leftarrow \emptyset$.
3: **for** $i = 1$ to $|Y|$ **do**
4: S_y checks item b_i against BF(X) by using hash functions $\{h_j(.), 1 \le j \le k\}$.
5: **if** $\exists(j)(1 \le j \le k)$, s.t. $h_j(b_i) = 0$ **then**
6: $Y - BF(X) \leftarrow (Y - BF(X)) \cup \{b_i\}$.
7: S_y transmits $Y -$BF(X) to the client.

Algorithm 2 shows detailed process of distributed union for "OR" queries [1].

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

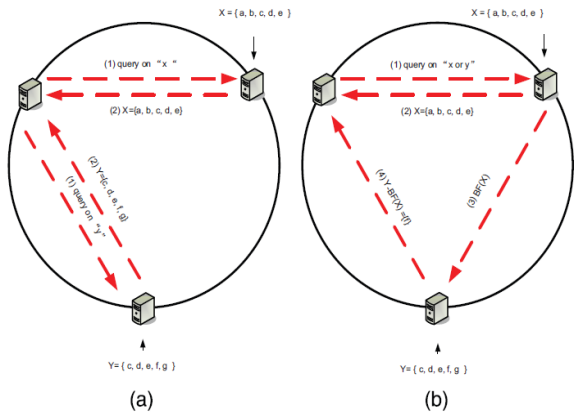

Fig.3. Straightforward distributed union versus BF-based union operation [1].

IV. Optimization Strategy For And Operation – Size Estimation

For queries with more than two keywords, by performing intersection operation for keywords which have smaller size of intersection we can able to achieve reduction in communication cost in high rate. But the complexity is, it is very difficult to estimate the size for each intersection before we perform the operation. In our design, we propose to use BF to estimate the intersection size for any pairs of keywords. The technique of estimating the size of intersection is explained as follows.

4.1 Intersection Size Estimation

Consider two BFs representing set X and set Y separately with same number of m bits and same set of hash functions. Now we deal with the inner product of the two BFs to calculate their similarities. Using the two BFs of X and Y it is very easy to obtain the inner product. If the ith bit in both BFs are set to 1 means, it may be set by some element which is present in X ∩Y, or by some element present in the set X - (X ∩Y) and by another different element in the set X - (X ∩Y). Hence in total, the probability that ith bit is set to "1" in both the BFs is mathematically represented in equation (3) [1].

$$(1 - (1- 1/m)^{k|X \cap Y|}) + (1- 1/m)(1 - (1- 1/m)^{k|X-(X \cap Y)|}) * (1 - (1- 1/m)^{k|X-(X \cap Y)|}) \quad (3)$$

By simplifying further, using the equation in [17] the inner product of two BFs is quantified as

$$P = m(1-(1- 1/m)^{k|X|}) - (1- 1/m)^{k|Y|}) + ((1- 1/m)^{k(|X|+|Y|-|X \cap Y|}) \quad (4)$$

Given |X|, |Y|, k, m and p, now we can calculate the intersection size using the following equation (5).

$$|X \cap Y| = \frac{- \log_{1-1/m}(p/m+(1-1/m)^{k|X|})+(1-1/m)^{k|Y|} - 1)}{K} + (|X| + |Y|) \quad (5)$$

4.2 Estimation from Search History

Due to the fact that, the above technique is infeasible to apply for all pairs of keywords and also impossible to predict for all the combinations of keyword pairs, which involves high communication cost. To solve this issue we make use of the query history to find out frequent optimal pairs. More specifically in fig 2b BF(X) is cached in the node which is responsible for keyword y to calculate X∩Y. more those keywords are searched by users more they are correlated and its estimated size will be stored on both the nodes for x and y for future queries [1].

V. Global Keyword Popularity

In order to minimize the communication cost, we proposed a method of gathering popularity of the keywords globally. In our design of hybrid P2P network we use a push synopsis gossip algorithm to collect the keyword popularity globally in every participating peer. It is not a difficult task to collect the keyword popularity from the inverted index list of the corresponding DHT node. But in our design we are not gathering information from DHT nodes due to the extra search latency. Because this process requires to travel O (log (N)) hops across the wide area network to find the frequency of keywords [1].

In contrast, we tend to use gossip algorithm, to gather global keyword popularity. Using this algorithm we can find the global frequency of all the keywords directly from the local synopsis i.e., index list with constant latency O(1), greatly reducing the processing time and so reduces the search latency [1].

The main concept of this gossip algorithm technique is as follows: for example consider |X|, the global statistics of keyword x. first, to find the popularity of keyword x in the documents among peers we follow this method. This method first let all the peers to check in their own local index for keyword x. when the keyword is found for the first time in the document on a peer it performs the following operation called Flip Coin operation. It flips a coin for t times and counts the number of heads appeared before the first tail. Then it saves this count as FC(x). Now the FC(x) value of the local peer is gossiped among randomly chosen neighbor peer.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

After n rounds of gossiping and receiving the FC(x) values from neighbor peers, a peer computes the maximum of FC(x) values i.e., *max*FC(x) [1]. This technique leads the computation of aggregate information to compute |X|, i.e. the documents with highest popularity of keyword x as per the results shown in [17].

There are three main operations involved in the push synopsis gossip algorithm to compute the global statistics of the keyword. They are synopsis generation, synopsis dissemination, synopsis merging.

i) Synopsis generation: synopsis generation is the first phase. When a peer joins the network, it browses its local index information to build from a local synopsis using the duplicated-insensitive counting technique. Design of synopsis is (x, bitvec_x) where bitvec_x is no of frequency count of the keyword x.

ii) Synopsis dissemination: In the second phase, synopsis is distributed among the peers. It is done by using push synopsis gossip algorithm proposed in [17]. Each peer chooses a neighbor peer randomly and sends its local synopsis. Fig 4 illustrates a sample of one round of gossip among nodes [1].

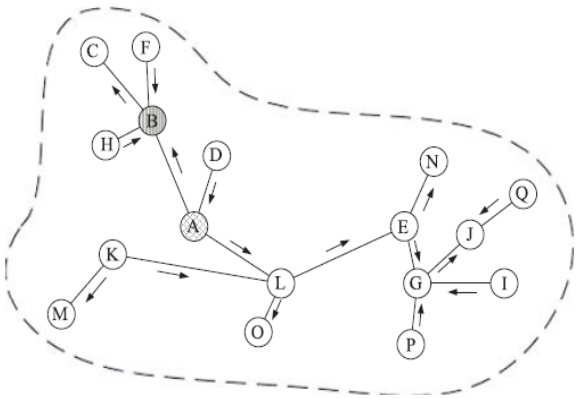

Fig 4. One round of gossip [1].

iii) Synopsis merging: when a peer receives synopsis from its neighbor peer it checks the received synopsis against its own synopsis and performs merging operation based on the following conditions. If the both the received and own synopsis contains the keyword x it performs the bitwise-or operation to merge the results. If the received synopsis contains the keyword but not the local synopsis, then it merges the particular bit vector into its own synopsis. Fig 5 illustrates the process of merging by peer B with Peer A synopsis which gossips to it [1].

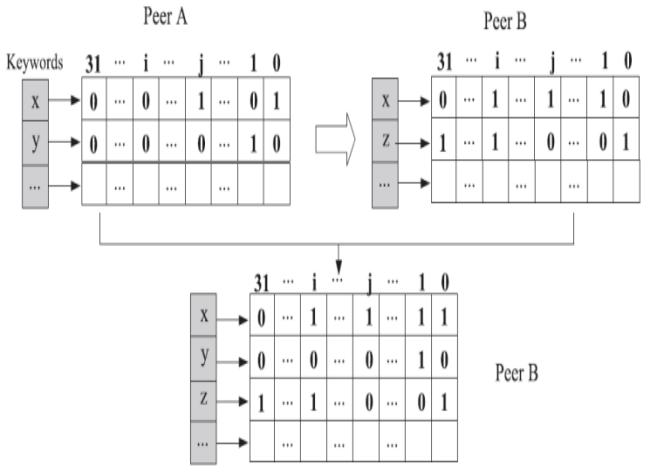

Fig 5. Synopsis Merging [1].

Here we may also deal with the issue of change over time. Because when a node leaves the network it has no way to delete the information it contributed to the synopsis. To solve this issue, we set a maximum value to check. When a node receives a counter value larger than the maximum value say V, it drops both the incoming and local synopsis and re-computes the synopsis based on a current document set. Then it performs the gossip protocol based on new synopsis. By this technique the stale data will leave the system eventually [1].

VI. Virtual Host Technique

To address the load balancing issue in decentralized structure P2P networks due to the variation in node capacity, we utilize virtual host technique. By the use of virtual servers each node can able to maintain more than one logical host. Existing load balancing issue can be solved by transferring virtual servers from heavy loaded nodes to the lightly loaded nodes. So that nodes capacity can be maintained in a distributed manner. To deal with the issue caused by skewed distribution of queries, we can use splitting and merging strategies proposed by Rao et al in [18]. When the virtual server load is higher than the threshold value which is predefined, we split virtual server based on its ID length. So that load of the virtual server can be bounded by predefined threshold value. In the same way light loaded virtual servers can be merged in to a new virtual server with the value below the threshold value [1].

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)
International Conference on Information Systems and Computing (ICISC-2013), INDIA.

VII. SCALABILITY

In P2P search engines, the number of results for a given query is directly proportional to the number of documents in the network. Hence the communication cost of retrieving all the results to the client will grow linearly. Bloom filter improves the constant factor but not the linear growth in cost. We can solve this scalability problem effectively by combining top-k pruning techniques with the bloom filter techniques [16].

VIII. APPLICATION: YACY

Related to our work, a real time application which demonstrates our whole work is YaCy, a P2P search engine. YaCy is an open source distributed search engine, built on the principles of P2P networks. System architecture of YaCy search engine is built based on four elements.

First, a Crawler, a search robot which travels across web pages to analyze the contents. Second, an indexer, a reverse word index which enlists the list of relevant URLs and its ranking information. Index list contains only the hashed values of the keywords using set of hash functions and utilizes optimal bloom filter settings, a data structure to check whether the keyword is present in the list or not. Third, a search and administrative interface, which is a web interface provided by a local HTTP servlet with servlet engine. Fourth, data storage, to store the index information database of the peers by utilizing the Distributed Hash Tables (DHT).A demo of YaCy search portal is present in http://search.yacy.net [19].

IX. CONCLUSIONS

This paper presents an effective way to achieve the optimal bloom filter settings in terms of traffic cost. Our work makes two main contributions to this paper. First, we proposed an optimization strategy for "AND" and "OR" queries by pre-estimating the size of intersection sets using bloom filter to reduce the communication cost. Second, we use a variant of push-synopsis gossip algorithm to show how the gossip based computation of global keyword popularity improves the search efficiency, reduces the search latency and also the bandwidth costs. We also described a virtual host technique to address the load balancing issues. Here we also suggested the effective solution to scalability problem with the use of bloom filter techniques. Results demonstrate that an optimal setting of BF is determined by the global keyword popularity of the keywords and the intersection order.

Simulation results shown in [1] shows that our proposed approach significantly reduces the search traffic cost and search latency of the existing approaches [1].

REFERENCES

[1] Hanhua Chen , Lei Chen, Yunhao Liu, " Optimizing Bloom Filter Settings in Peer to Peer Multikeyword searching", in proceedings of IEEE Transactions on knowledge and data engineering, April 2012.

[2] Chahine, M.K. and Mazzini, G., "P2P application for file sharing", in Proceedings of Telecommunications (ICT), 19th International Conference April 2012.

[3] Fujitha.S, "Proximity-Aware DHT for Efficient Lookup Service in Peer-to-Peer Applications", in Proceedings of IEEE transactions on computer science and engineering, 2011.

[4] B.Bhatacharjee, S.chawathe, V.Gopala krishnan, et al., "Efficient Peer-to-Peer Searches using Result Caching", in Proceedings of IPTPS 2003.

[5] Zhu,Y. And Jiang, H., "False Rate Analysis of Bloom Filter Repoicas in Distributed Systems", in Proceedings of ICPP, 2006.

[6] Li,J.,Loo,B.T., Hellerstein,J.M., Huuebsch,R., Shankar,S., and Stoica,I., "Enhancing P2P File Sharing with an Internet Scale Query processor", in Proceedings of VLDB ,2004.

[7] P. Reynolds and A. Vahdat., "Efficient Peer-to-Peer Keyword Searching," in Proceedings of. Int'l Conf. Distributed Systems Platforms and Open Distributed Processing (Middleware), 2003.

[8] D. Hawking., "Overview of the TREC-9 Web Track," in Proceedings of Text Retrieval Conf. (TREC-9), 2000.

[9] Lu, J. and challan, J.P., "User Modeling for Full Text Federated Search in Peer to Peer Networks", in Proceedings of SIGIR, 2006.

[10] Cucnca-Acuna, F.M., Peery, C., Martin, R.P., and Nguyen, T.D., "PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities", in Proceedings of HPDC, 2003.

[11] Tang, C. and Dwarkadas, S., "Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval", in Proceedings of NSDI, 2004.

[12] Hanhua Chen, Jun Yan, Hai Jin, Yunhao Liu, Lionel M. Ni, "TSS: Efficient Term Set Search in Large Peer-to-Peer Textual Collections," in Proceedings of IEEE Transactions On Computers, Vol. 59, NO. 7, July 2010.

[13] I.podnar, M.Rajman, T.Luu, F.Kleman and K.Aberar., "Scalable Peer to Peer Networks with Highly Discriminative Keys," in Proceedings of IEEE Int'l Conf Data Eng.(ICDE),2007.

[14] M.Bender, S.Michel. "P2P Content Search: Give the Web Back to the People," in Proceedings of Int'l Workshop peer to peer System (IPTPS),2006.

[15] Sasu Tarkoma, Christian Esteve Rothenberg, Eemil Lagerspetz., "Theory and Practice of Bloom Filters for Distributed Systems," in proceedings of IEEE Communications Surveys and Tutorials, Vol. 14, NO. 1, Pg. 131-155, 2012.

[16] J. Zhang and T.Seul , "Efficient Query Evaluation on Large Textual Collections in a Peer-to-Peer Environment ",,in Proceedings of IEEE Int'l conf. Peer to Peer computing (P2P), 2005.

[17] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregation Information," in Proceedings of Ann. IEEE Symposium Foundations of Computer Science (FOCS), 2003.

[18] A. Rao, K. Lakshminarayanan, S. Surana, R.M. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems," in Proceedings of Int'l Workshop Peer-to-Peer System (IPTPS), 2003.

[19] YaCy, Yet Another CYberspace, http://yacy.net, (Accessed on November 27, 2012).