BloomCast: Efficient and Effective Full-Text Retrieval in Unstructured P2P Networks

Hanhua Chen, *Member*, *IEEE*, Hai Jin, *Senior Member*, *IEEE*, Xucheng Luo, Yunhao Liu, *Senior Member*, *IEEE*, Tao Gu, *Member*, *IEEE*, Kaiji Chen, and Lionel M. Ni, *Fellow*, *IEEE*

Abstract—Efficient and effective full-text retrieval in unstructured peer-to-peer networks remains a challenge in the research community. First, it is difficult, if not impossible, for unstructured P2P systems to effectively locate items with guaranteed recall. Second, existing schemes to improve search success rate often rely on replicating a large number of item replicas across the wide area network, incurring a large amount of communication and storage costs. In this paper, we propose BloomCast, an efficient and effective full-text retrieval scheme, in unstructured P2P networks. By leveraging a hybrid P2P protocol, BloomCast replicates the items uniformly at random across the P2P networks, achieving a guaranteed recall at a communication cost of $O(\sqrt{N})$, where N is the size of the network. Furthermore, by casting Bloom Filters instead of the raw documents across the network, BloomCast significantly reduces the communication and storage costs for replication. We demonstrate the power of BloomCast design through both mathematical proof and comprehensive simulations based on the query logs from a major commercial search engine and NIST TREC WT10G data collection. Results show that BloomCast achieves an average query recall of 91 percent, which outperforms the existing WP algorithm by 18 percent, while BloomCast greatly reduces the search latency for query processing by 57 percent.

Index Terms—Peer-to-peer systems, Bloom Filter, replication.

1 Introduction

With the emergence of peer-to-peer (P2P) file sharing applications, such as Napster and Gnutella, millions of users have used P2P systems to search desired data. A P2P network has also shown great potential to become a popular network tool for sharing information on the Internet [1].

Existing P2P full-text search schemes can be divided into two types: DHT [2]-based global index and federated search engine over unstructured protocols. DHT-based searching engines are based on distributed indexes that partition a logically global inverted index in a physically distributed manner. Due to the exact match problem of DHTs, such schemes provide poor full-text search capacity. In federated search engines over unstructured P2Ps, queries are processed based on flooding. Unstructured P2Ps are commonly believed to be the best candidate for supporting full-text retrieval because the query evaluation operations can be handled at the nodes that store the relevant documents. However, search recall is not guaranteed with acceptable communication cost using a flooding-based scheme.

Replication strategies are extensively utilized to improve search performance in unstructured P2Ps. The existing replication strategies can be divided into two categories. The first type is the query popularity aware strategies [3]. Such strategies assume that the access frequencies of the items are known and the number of replicas is determined by the query's popularity. Cohen and Shenker [3] claimed that the square-root replication strategy, where the number of the replicas is proportional to the square-root of the query popularity/rate, has the optimal search performance. In query popularity aware replication strategies, the items with high query rate are highly replicated for future query searching, thus the search performance for popular items are improved. However, the strategy is inefficient for solving *insoluble queries*, the queries for rare items [3]. Moreover, in practice, the query frequency is difficult or even impossible to obtain in a distributed P2P system.

The second type of replication strategy is independent of the popularity of the query, such as the WP scheme [4]. By replicating data and query replicas randomly across a P2P network regardless of the query rate of the data, such kind of schemes improve search recall of queries no matter they are popular or not. In WP scheme, the term query replica is used to differentiate a query message transferred across the

- *H. Chen and H. Jin are with the Services Computing Technology and System Laboratory, Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: {chenhanhua, hjin}@hust.edu.cn.*
- *X. Luo is with the School of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu 610054, China. E-mail: xucheng@uestc.edu.cn.*
- *Y. Liu is with the Tsinghua National Lab for Information Science and Technology (TNLIST) and School of Software, Tsinghua University, China and the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: yunhaoliu@greenorbs.com.*
- *T. Gu and K. Chen are with the Department of Mathematics and Computer Science, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark. E-mail: {gu, chen}@imada.sdu.dk.*
- *L.M. Ni is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, and the Shanghai Key Lab of Scalable Computing and Systems at Shanghai Jiao Tong University, Shanghai, China. E-mail: ni@cse.ust.hk.*

Manuscript received 16 Oct. 2010; revised 10 Mar. 2011; accepted 18 Apr. 2011; published online 25 May 2011.
Recommended for acceptance by A. Kshemkalyani.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2010-10-0614.
Digital Object Identifier no. 10.1109/TPDS.2011.168.

network without performing and a query that evaluated in a node. A query replica will be performed by the node holding it. In [4], the WP scheme utilizes random walk technique to deploy replicas. The problem of random walk-based scheme is that it is not fault-tolerant. Another problem of the existing replication strategies is that simply replicating document reference or selected metadata cannot successfully support full text retrieval. To support full text retrieval, the existing replication strategies need to replicate the full document across the network, raising possibly unacceptable communication and storage costs.

To address the problems of the query popularity independent replication strategy, we propose a novel strategy, called BloomCast, to support efficient and effective full-text retrieval in this paper. We show mathematically that the recall can be guaranteed at a communication cost of $O(\sqrt{N})$, where N is the size of the network. Different from the WP scheme [4], BloomCast hybridizes a lightweight DHT with an unstructured P2P overlay to support random node sampling and network size estimation. Furthermore, we propose an option of using Bloom Filter encoding instead of replicating the raw data. Using such an option, BloomCast replicates Bloom Filters (BF) [5] of a document. A BF is a lossy but succinct and efficient data structure to represent a set S, which can efficiently process the membership query such as "is element x in set S." By replicating the encoded term sets using BFs instead of raw documents among peers, the communication/storage costs are greatly reduced, while the full-text multikeyword searching are supported. We show the effectiveness and efficiency of BloomCast through mathematical proof and comprehensive simulations based on NIST TREC WT10G data collection and query logs from a major commercial search engine. Results show that BloomCast can achieve guaranteed recall with largely reduced search latency, significantly outperforming existing schemes. Results also show that for multikeyword searching, Bloom Filter encoding can greatly reduce the communication cost for data replication.

The remainder of the paper is organized as follows: In Section 2, we discuss the related work. In Section 3, we present the model of BloomCast. Section 4 describes the system design of BloomCast in detail. We evaluate the performance of this design in Section 5. The conclusion is given in Section 6.

2 Related Work

Full-text search is an important issue in distributed P2P information sharing systems. Without centralized index servers, nodes in a decentralized P2P system have to cooperate with each other to perform a full-text search. Existing P2P content search schemes can be divided into two types: DHT-based distributed global inverted index on top of structured P2P networks, and federated search engines over unstructured P2P networks.

2.1 Full-Text Search in Structured P2P Networks

DHT-based full-text searching engines utilize distributed global indexes, which partition a logically global inverted index in a physically distributed manner. Built on existing DHTs, single-term-based distributed index can effectively support single keyword search by retrieving the list of documents for a given keyword. Because of the utilization of the exact hashing techniques, the DHT-based schemes, however, fail to support complex queries with multiple keywords. Tang and Dwarkadas [6] proposed a hybrid index scheme, where the frequent terms of a document are selected to be published on a global index. When such a keyword is published, the list of other terms in the document is replicated with the identifier of the document in the posting list. Multikeyword search is performed by first locating the position of the DHT node which is responsible for a given keyword and then performing a local search for other keywords in the posting list. Finally only the list of documents that contain all the keywords is returned as the results. Little is known about the performance of the full text search using selected keyword publishing, because a few selected frequent terms may not be representative for a document [7] and such replication strategy may incur unacceptable storage and communication cost. Another scheme performs a distributed intersection operation for multikeyword search. Based on the global single term-based inverted index built on DHT, the scheme looks up the sets for different keywords from multiple peers across the wide area network and returns the intersection. Although only a few nodes need to be contacted, each node has to send a potentially large amount of data across the wide area network, making such scheme unscalable. Reynolds and Vahdat [8] used Bloom Filters to encode the transferred lists while recursively intersecting the matching document set. Consider an example of a two-keyword (x, y) query search. The query is first routed to the DHT node which is responsible for keyword x. Then the DHT node identifies X, the list of identifiers of documents that contain x. It then generates a Bloom Filter for set X, denoted by $BF(X)$, and transmits $BF(X)$ to the DHT node responsible for keyword y, where the intersection of X and Y is estimated based on $BF(X)$. Due to the possible false positives of BFs, the result set may include elements that contain only keyword y but not x. To pick out the false positives, the scheme sends the estimated intersection, denoted by $Y \cap BF(X)$, back to the DHT node responsible for keyword x to calculate $X \cap (Y \cap BF(X))$, which is equivalent to $X \cap Y$. By transmitting the BFs of the sets instead of the raw sets among DHT nodes during the intersection with an inverse verification, the communication cost can be effectively reduced. However, the length of the set is roughly proportional to the size of the network (document collection). The Bloom Filter-based scheme achieves a substantial constant factor improvement; but it does not eliminate the linear growth in the communication cost.

2.2 Search in Unstructured P2P Networks

It is commonly believed that unstructured P2Ps are promising to provide full-text content searching in large scale distributed environments. In this kind of search networks, peers which maintain indexes of their local documents are organized in an ad hoc fashion. Without a global index, unstructured P2P networks rely on flooding-based schemes to distribute queries to the network. Thus, the queries can be handled on peers containing relevant documents. Although unstructured P2P systems can naturally support full-text query evaluation, achieving efficient and effective search over unstructured P2Ps is challenging. First, the flooding-based protocols incur exponentially growing communication cost, restricting the scalability of the system. Second, the recall cannot be guaranteed unless a query is flooded exhaustively throughout the network.

Existing unstructured federated P2P search schemes often perform the query evaluation in two levels, the peer level and document level. The scheme first detects a group of peers with potential answers to the query, and then the query is submitted to the selected peers to evaluate the query against their local indexes and return the matched answers [9]. The search performance of unstructured federated P2P search engines can be further improved using superpeer-based P2P architectures [10], which considers the inherent heterogeneity of peers [11]. Peers with more memory, processing power, and network connection capacity provide distributed directory services for resource location. Thus, the peers with limited resources won't become bottlenecks in the search network. Federated P2P search approaches can also take advantage of the enhanced properties of the network topology [12], [13] to improve search efficiency. In [12], Sripanidkulchai et al. identified a powerful principle, called interest locality, in popular P2P file sharing systems, such as Kazaa and Gnutella. The principle revealed that if a peer has a particular piece of content that a user is interested in, it is likely that the peer has other items that the user has interest in as well. Using interest locality, they linked the peers with similar interests. By forwarding the queries through the interest-based shortcuts during search, the search performance is greatly improved due to the significant amount of flooding avoided. SSW [13] dynamically clusters peers with semantically similar data closer to each other and maps these clusters in a high-dimensional semantic space into a one-dimensional small world network to improve search performance.

Another promising scheme for content search in unstructured federated P2P systems is to utilize replication strategies to improve search performance. By replicating items and queries properly across the network, such strategies can significantly improve the search success rate while avoiding exhaustively flooding the unstructured P2P networks. Existing replication strategies in unstructured P2P networks can be divided into two categories, the query popularity aware replication approach and the query popularity independent replication strategy.

The query popularity aware replication strategies assume that access frequencies of the items are known and the number of replicas is determined by the query's popularity. Let r_i denote the number of replicas of item i [3]. The sum of the replica amounts is $R = \sum_{i=1}^{m} r_i$. Let q_i denote the query rate of item i, which is the fraction of all the queries that are issued for item i. The number of replicas in this strategy is $r_i = f(q_i)$. Two natural strategies among existing schemes are uniform and proportional strategies, while in the uniform replication strategy, all items are equally replicated, that is $r_i = R/m$. In the proportional replication strategy, the number of replicas is proportional to the query rates, that is $r_i = R \times q_i$. Cohen and Shenker [3] studied the two query-rate based strategies. Their results showed that both the strategies are not better than square-root strategy, where the number of replicas is proportional to the square-root of the query rates. It is also proved that query popularity aware replication strategies are only efficient for "soluble queries", while for the rare items, the search performance is low. Furthermore, how to identify popular and unpopular queries is difficult, if not impossible.

Recently, the query popularity independent replication strategy has attracted much attention. In such schemes, items are equally replicated regardless of the popularity of the related queries. For full-text search, documents and queries are both replicated to some randomly selected nodes. The WP scheme [4] employs random walk to sample nodes at random. A peer p publishes its content by performing a random walk of length of $c \log n + \gamma\sqrt{n}$ hops, where c and γ are constants. During the first $c \log n$ hops, peers along the walk just forward the message to a random node without replicating, while the peers along the remaining $\gamma\sqrt{n}$ hops of the random walk install the references to p's content. Random walks, however, are not fault-tolerant. A failed node along the path could make the replication strategy fail. Thus, it is difficult to guarantee the search success rate. Another problem of existing replication strategies is that simply replicating document reference or selected metadata cannot successfully support full text retrieval. To support full text retrieval, existing replication strategies need to replicate the full document across the network, raising possibly unacceptable communication and storage costs.

To solve the above problems, in this paper we propose BloomCast, a novel replication strategy to support efficient and effective full-text retrieval. Different from the WP scheme, BloomCast leverages a lightweight DHT for random node sampling. We mathematically show that the optimal number of replicas is bounded by $O(\sqrt{N})$, where N is the network size. By further replicating the optimal number of Bloom Filters instead of the raw documents, BloomCast can achieve guaranteed recall rate while significantly reducing the communication cost for replicating. Based on the Bloom Filter membership verification we design a query evaluation language to support full-text multikeyword search.

3 Model

Before introducing the detailed design of BloomCast, we specify the system model of BloomCast. In this section, we will introduce the BloomCast replication model. Theorem 1 shows that given a number of r data replicas and a number of g query replicas, if BloomCast deploys all the replicas uniformly at random across the network, the lower bound of the search success rate is $1 - e^{-\frac{rg}{N}}$ [4]. Given a network size N, the lower bound of search success rate is determined by the product of numbers of data replicas and query replicas. Specifically, the lower bound is monotonically increasing as the product $r \cdot g$ increases. We give the proof of Theorem 1 in Section 1 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

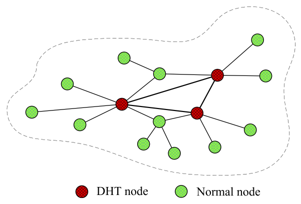


Fig. 1. Hybrid architecture of BloomCast.

Theorem 1. *Given a P2P network with N nodes, if we replicate r copies of a data and g copies of related queries, both uniformly at random, the probability that at least one peer has both a data replica and a query replica is not less than $1 - e^{-\frac{rg}{N}}$.*

Given a lower bound of success rate, it is important to calculate the number of replicas to deploy. In Theorem 2, we show how we can achieve a formal bound of the number of replicas. Theorem 2 shows that the optimal replica number is determined by $\sqrt{N}$. The minimal cost can be achieved when $r = c\sqrt{N \cdot \frac{S_q}{S_d}}$ and $g = c\sqrt{N \cdot \frac{S_d}{S_q}}$. Since the parameters c, S_q, and S_d are constants, the formal bound of the replication cost of BloomCast model is $O(\sqrt{N})$. We put the proof of Theorem 2 in Section 1 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

Theorem 2. *Given the size of the data replica is S_d and the size of the query replica S_q. To achieve a specified search success rate, the minimal numbers of data replicas and query replicas are in proportion to $\sqrt{N}$, where N is the size of the network.*

4 System Design

In this section, we describe the design of BloomCast in detail. We first give an overview of BloomCast scheme in Section 4.1. We then address the issues of random node sampling and network size estimation in Sections 4.2 and 4.3, respectively. Finally, we describe the Bloom Filter-based replicating protocol and query evaluation protocol of BloomCast in Sections 4.4 and 4.5.

4.1 Overview

It is clear that the BloomCast model works only when the two constraints are met: 1) The query replicas and document replicas are randomly and uniformly distributed across the P2P network; and 2) Every peer knows N, the size of the network. To support random node sampling and network size estimation, BloomCast combines a lightweight DHT into the unstructured P2P network. To further reduce the replication cost, BloomCast utilizes Bloom Filters to encode the full documents.

Fig. 1 shows the architecture of the BloomCast hybrid P2P network. A BloomCast hybrid P2P network has three types of nodes: normal peers, structured peers, and bootstrap peers. A BloomCast peer stores a collection of documents and maintains a local repository. A bootstrap node maintains a partial list of BloomCast nodes it believes are currently in the system. There are different ways to implement the bootstrap mechanism in previous P2P designs [14]. A new BloomCast node first contacts a bootstrap node to join. A small fraction peers with good connectivity and long uptimes are promoted to structured peers by the bootstrap peers to form a global DHT. Initially, a normal peer browses it local repository. For each document, it generates a Bloom Filter to represent the document in a compressed form. For each term appearing in the document for the first time, the peer inserts the term into the Bloom Filter using the set of hash functions. The functionality of the DHT in BloomCast is different from other existing Hybrid P2P networks. In previous hybrid P2P systems, DHT is used for indexing and querying rare items. In contrast the DHT in BloomCast stores the state information of normal peers to provide services of random node sampling and network size estimation. We will describe how the DHT of BloomCast supports network size estimation and random nodes sampling in detail in Sections 4.2 and 4.3, respectively. For replication, a normal peer first utilizes the connected DHT nodes to sample an optimal number of peers in the network, and then deploys the replica in the form of a Bloom Filter attached with the *url* of the document into the selected random nodes. During searching, a query replica (a set of keywords) will be deployed using the similar way. Thus, the multikeyword query can be evaluated by using the membership verifying function of Bloom Filters. The *urls* corresponding to the Bloom Filters that contain all the keywords in the query will be returned to the client as the results for downloading.

4.2 Network Size Estimation

BloomCast estimates the network size by exploiting the node information stored in the DHT subsystem. Since each DHT node charges an ID section, intuitively, the network size can be inferred by the density of node entries in an ID section. If the DHT nodes are distributed in the ID space uniformly, each DHT node charges an equal length of ID partition. This length is denoted as α. If all nodes in the P2P system are also mapped onto the ID space randomly, each DHT node obtains the same number of normal P2P nodes, which is denoted as ω. Thus, the density of node information charged by a DHT node is $d = \frac{\omega}{\alpha}$. The size of ID space is known as L, for example, $L = 2^{160}$. Therefore, the network size estimation is $N = Ld = L\frac{\omega}{\alpha}$. However, the density of node entries in each DHT node may not be the same in practice. The longest partition is $\Theta(\frac{\log L}{L})$ [15] and the shortest partition is $\Theta(1/L^2)$ [16]. This large deviation may incur inaccurate estimations on the density.

To obtain good estimation of the network size, BloomCast estimates the network size through the information aggregated from several DHT nodes. For a node i in DHT, it has local observation $<l_i, x_i>$, where l_i is the length of its ID partition and x_i is the number of data items stored on node i. Therefore, the density in node i is $d_i = \frac{x_i}{l_i}$. If we collect a number of DHT nodes and aggregate their $<l_i, x_i>$, we have $l = \sum l_i$ and $x = \sum x_i$. The network size thereby can be estimated as $\hat{N} = L\frac{x}{l}$. The parameter L is specified during design of systems, for example, $L = 2^{16}$. By way of the above aggregation, we can obtain parameters x and l. The key point here is how to choose a reasonable l for the design. To determine l, we propose Theorem 3. We put the proof of Theorem 3 in Section 2.1 of the supplementary file, which can be found on the Computer Society Digital

Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

Theorem 3. *If $l = \frac{4\varepsilon \ln N}{\delta^2 N} L$, the probability of $(1-\delta)N \le \hat{N} \le (1+\delta)N$ is $1 - 2N^{-\varepsilon}$, where $\varepsilon > 0$ and $0 \le \delta \le 2e - 1$ are two constants.*

4.3 Node Subset Sampling

Node subset sampling can be implemented by the query function of the DHT subsystem. Each node has an entry in the DHT subsystem. If a node wants to get a random peer, it first generates a random ID of the ID space. The node then queries the DHT subsystem for this ID. If this ID has been occupied by some other nodes in the system, the DHT subsystem returns the corresponding IP address to the querying node. Otherwise, the IP address of the first peer following the queried ID is returned. To sample a set of s nodes, s random IDs should be generated. Since $\frac{s}{N}$ is very small, the probability of selecting one node more than once is negligible. In the DHT subsystem, the message overhead of each query is $O(\log n)$, where n is the number of nodes in the DHT subsystem. Correspondingly, to select s random nodes, the message overhead is $O(s \log n)$ and the latency is $O(s \log n)$.

By exploiting the neighbor list of DHT nodes, the message overhead and latency can be further reduced. First, the requested DHT node generates a random ID set. Then, this node sorts these IDs to get an ordered ID list. Since the IDs are generated randomly, these IDs are distributed on the DHT uniformly. The requested DHT node divides the ordered ID list into $\log n$ sections according to the neighbor list. Each neighbor takes charge of one section for query. By this way, each message can at least solve one random node selection. In short, to select a number of s random nodes collectively, a number of s messages are sufficient. The parallel queries by utilizing the neighbor list can further reduce the query latency.

4.4 Replication Protocol

Traditional replication strategies cannot provide real full-text search capacity unless the full documents are replicated. The problem is that replicating raw documents across the P2P network may raise a large amount of communication cost, making the schemes infeasible in practice.

To address this problem, we propose to use Bloom Filter [17], [5] to reduce the cost. We give a brief review of Bloom Filter in Section 2.2 in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

By distributing the document replicas in the form of Bloom Filters instead of raw data, BloomCast is able to achieve significant cost savings while supporting accurate full text retrieval with high probability. The basic idea of the scheme is described as follows:

When a peer joins the network, it browses its local index. For each document it contains, it generates a Bloom Filter to represent it. It browses the document from the beginning to the end. When it meets a term in the document the first time, it inserts the term into the Bloom Filter by using the set of k hash functions, $\{h_j(\cdot), 1 \le j \le k\}$. When it meets the end of the document, it attaches the *url* to the Bloom Filter and deploys the replicas of the Bloom Filter into an optimal number of randomly sampled nodes in the network using the algorithm presented in Section 4.3. We describe the replication strategy in more detail in Algorithm 1 in Section 2.3 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

4.5 Query Evaluation

When a user issues a query, BloomCast replicates it to an optimal number of $g = c\sqrt{N \cdot \frac{S_d}{S_q}}$ random peers sampled using the algorithm described in Section 4.3. Every receiving peer checks all the keywords of the query against the BFs replicated locally. It is not difficult to see that the member verification mechanism provided by the Bloom Filter can effectively support multikeyword full text retrieval. If all the keywords in the query are tested to be contained in a Bloom Filter, the document corresponding to the replicated Bloom Filter should be the desired result with high probability. The *url* of the document is then returned as the results for the client to download the document. If any keyword in the query is tested to be not an member of the Bloom Filter, the document corresponding to the Bloom Filter is clearly not what the user wants. Due to the false positives of Bloom Filters, the returned results may contain undesired documents with very low probability. This may lead to a slight decrease of the precision of the final results while keeping the recall guaranteed. It is easy to see that the false positives can be removed by forwarding the query to the peers which contain the unwanted *urls*. Algorithm 2 in Section 2.4 in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168, describes the query evaluation process in detail.

5 Methodology

We evaluate the performance of BloomCast design using trace-driven simulations. In this section, we describe the simulation setup. First, we introduce the Gnutella traces we collected. We then describe the data used for evaluation including the WT10G data collection from NIST and the query logs. Finally, we present the metrics used for performance evaluation.

In order to well represent real world systems, we consider both the underlying physical topology and the P2P overlay.

The physical topology should represent the real topology with Internet characteristics. Previous studies [18] have shown that a large scale Internet physical topology follows the small world and power law properties. The topology of a small-world network has the properties of sparseness, short global separation, and high-local clustering of nodes while power law denotes the property of the node degree distribution. The study from Tangmunarunkit et al. [18] found that the topologies generated using the AS model have the properties of the small world and power law. BRITE [19] is a topology generation tool that provides the option of generating topologies based on the AS model.

Using BRITE, we generate a physical topology with 100,000 nodes. Using the physical topology generated by BRITE, we can simulate the underlying Internet with rich configuration information, including bandwidth configuration, latency, and so forth.

We have developed a crawler in Java based on the limewire open source client to collect topology information of Gnutella network [20]. We then use the traces to simulate a real P2P network. We have described how the crawler works in detail in Section 3.1 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

Using BRITE, we configure the upload bandwidth of a peer according to the measurement study on MSN from Microsoft [21] in 2007. The study has shown that 97.2 percent MSN video users have upstream bandwidth higher than 128 Kbps (16 KBps). This corresponds to a DSL1 line quality. In the experiment, we set the upload bandwidth of a peer to 128 Kbps (16 KBps) and set the download bandwidth to 768 Kbps (96 KBps). On one hand, this conservative configuration about peer bandwidth capacity indeed pushes the system performance examination close to the system limits. On the other hand, in practice a real-world peer-assisted text retrieval system may not want to fully exploit the available bandwidth of a high capacity peer, as doing so might deter their participation.

All P2P nodes in the trace are mapped into the underlying physical topology. The communication cost between two logical neighbors is calculated based on the physical shortest path between the pair of nodes. The uptime of peers follows the distribution of Gnutella P2P systems reported in [22]. About 10 percent ultra peers have an average uptime longer than 80 minutes, among them 5 percent nodes are selected as the DHT nodes for node number estimating and node sampling. The Chord protocol [2] is used to connect the DHT nodes.

There has been no standard data set established for evaluating the performance of P2P content search [23]. We built one based on TREC WT10G collection [24], a large test set widely used for performance evaluation in information retrieval research area. We introduce the WT10G collection in more detail in Section 3.2 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

The number of queries provided by US National Institute of Standards and Technology (NIST) for the TREC WT10G web test collection is far from enough to be used in studies on P2P content search. We evaluate our design using the query logs of a major web search engine.

We have used several standard metrics for evaluating the performance of BloomCast. The evaluation considers both search quality and system efficiency. Quality focuses on user-perceived qualities such as recall, precision, F-Measure, and latency; while efficiency focuses on resource utilization such as traffic and efficiency. Since all of them are widely used standard metrics, we introduce the formal definitions of the metrics in Section 3.3 in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

6 Results

Since different documents in the network may have different numbers of unique terms, the settings of Bloom Filter should vary according to the size of documents to improve search performance. In the simulation we vary the setting of parameter k from 2 to 24 and compute the optimal value of parameter m by: $m = \frac{nk}{lg2}$ accordingly, which achieves the slightest false positive of Bloom Filters [25]. To choose the number of replicas, the parameter λ is set to 4. We find that when the setting of k varies, the recall of the BloomCast changes very slightly. In practice, the system administrator can further improve the recall by adjusting the parameter of λ. Such an adjustment will also increase the communication cost of the system. We also find that the precision increases greatly with the increase of parameter k. When k is increased to the value of 16, the precision reaches the maximum value. When the parameter k changes, the average traffic of the queries changes very slightly. When the value of parameter k increases, the F-Measure increases and reaches the maximum value of 0.95 at the point $k = 16$. The efficiency is quite stable when k changes. The results reveal that the quality of the search results and the efficiency of the system are quite acceptable if we set the parameter k of the Bloom Filter to 16. According to the results, we set k to 16 and compute the optimal value of m for each document in the following experiments. We show the figures for optimizing the parameters of a Bloom Filter to Section 4 of the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.168.

To evaluate the performance of BloomCast, in the simulation we implement three baseline schemes: the WP algorithm presented in [4], the flooding strategy, and the DHT-based scheme proposed by Reynolds and Vahdat [8]. For the WP algorithm, we set the parameter of c to one and set the parameter λ to two. The TTL in the flooding algorithm is set to seven [26]. When simulating the DHT-based multikeyword search algorithm, we set the size of Bloom Filter by $m = |A| log_{0.6185}(2.081 \frac{|A|}{|B|j})$ to achieve the minimized the false positive, where $|A|$ and $|B|$ are the sizes of the posting list in both sides during the intersection, respectively. When simulating the DHT-based scheme, we set j, the average URL length, to 250 bits based on the research results conducted on Google search engine, which shows that the average URL length measured in character is 31.2 characters [27]. We compare our BloomCast strategy with these two schemes.

Fig. 2 plots the recall of all the queries, where 91 percent BloomCast queries have a recall greater than 90 percent, while only 64 percent queries using the WP algorithm and none queries with the flooding scheme can achieve such a high recall. The average recall of the queries using BloomCast is 91 percent, which outperforms the WP algorithm by 18 percent. Such an improvement is 67 percent of the theoretical potential 27 percent improvement. BloomCast significantly outperforms the flooding strategy by 152 percent.

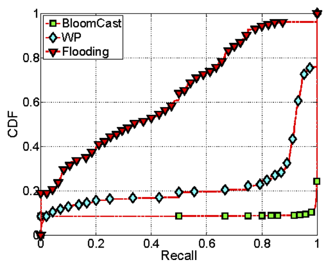


Fig. 2. Recall.

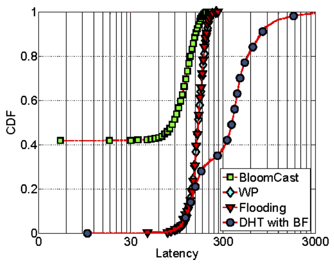


Fig. 3. Latency.

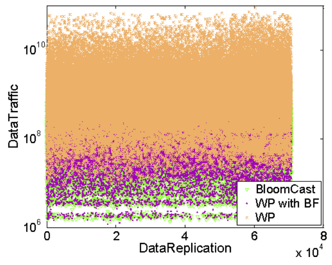


Fig. 4. Traffic for data replication.

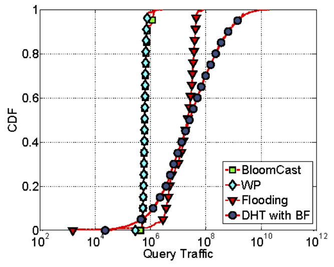


Fig. 5. Query traffic.

In Fig. 3, we evaluate the query processing latency. The result shows that the BloomCast scheme greatly reduces the search latency comparing to the WP algorithm and the flooding scheme. The average latency of the queries using BloomCast is 69 ms, which is only 43.7 percent of that of the flooding scheme and 43.6 percent of that of the WP algorithm. The average query latency of DHT-based scheme is 441 ms, which is 6.4 times longer than that of BloomCast.

Fig. 4 plots the traffic for data replication. The results show that the mean traffic for data replication using WP algorithm is 2.28×10^9, while the average traffic for BloomCast data replication is 2.31×10^8. We also extended the WP algorithm by using Bloom Filter encoding. The simulation results show that the average traffic for WP algorithm with Bloom Filter is 2.36×10^9. The result shows that the BloomCast scheme significantly reduces the traffic for data replication by 89 percent comparing to the basic WP strategy; while by using Bloom Filters in WP algorithm, the difference between WP and BloomCast is very slight.

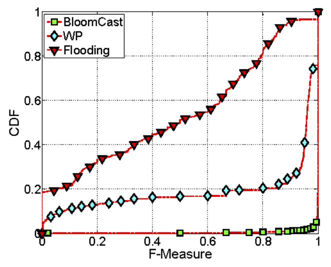


Fig. 6. F-Measure.

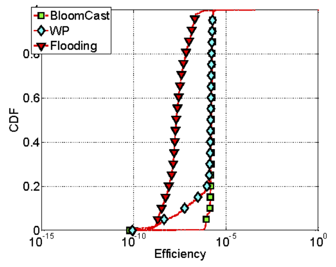


Fig. 7. Efficiency.

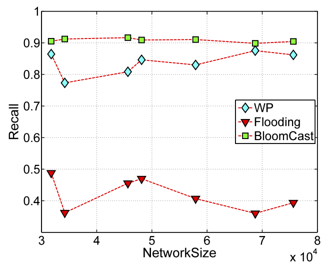


Fig. 8. Recall changes with network size.

The result in Fig. 5 shows that the average query traffic of BloomCast is 6.5×10^5, very similar with that of the WP algorithm. The average traffic of BloomCast is much less than that of flooding. The results show that the average traffic of DHT-based scheme is 2.9×10^8, two orders of magnitude larger than that of BloomCast.

Fig. 6 plots the F-Measure. The result shows that the BloomCast scheme greatly improves the overall quality of the search results comparing with the WP algorithm and the flooding scheme. The average F-Measure of BloomCast outperforms that of the flooding strategy by 119 percent, while improves that of the WP scheme by 23 percent.

Fig. 7 shows that the BloomCast scheme greatly improves the search efficiency comparing with both the WP algorithm and the flooding scheme. The average search efficiency of BloomCast outperforms that of the flooding strategy by 67 percent, as well as outperforming that of the WP scheme by 24 percent.

We vary the size of the network by using different Gnutella traces and evaluate the performance of BloomCast in different network scales. The result in Fig. 8 shows that when the size of network increases from 30K to 75K the recall of flooding decreases by 17 percent, from 48.8 to 40.67 percent. While the recalls of the BloomCast and the WP algorithms will not decrease when the network size increases. Overall, the recalls of BloomCast in different network scales are better than those of the WP algorithm. The result in Fig. 9 shows that in different network scales the precision of BloomCast changes very slightly, approximating to 100 percent.

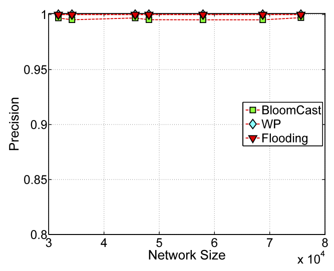


Fig. 9. Precision changes with network size.

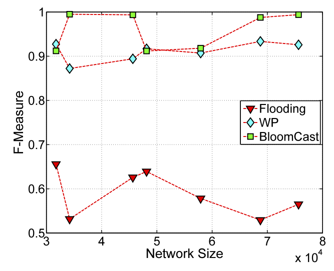


Fig. 10. F-Measure changes with network size.

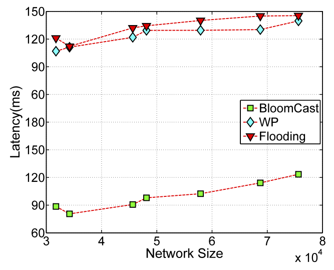


Fig. 11. Latency changes with network size.

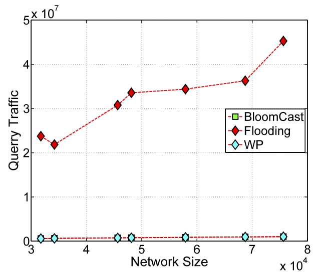


Fig. 12. Recall.

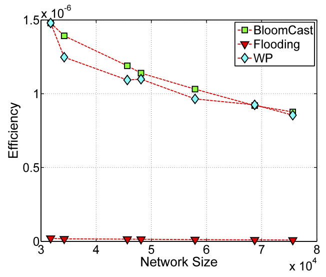


Fig. 13. Search efficiency changes with network size.

The result in Fig. 10 shows that when the network size increases from 30K to 75K the F-Measure of flooding decreases from 60 to 54 percent. The F-Measure of BloomCast increases from 91 to 99 percent, while the F-Measure of the WP algorithm decreases from 88 to 86 percent.

Fig. 11 shows that in different network scales search latency of BloomCast is much shorter than those of flooding and the WP algorithm, while the latencies of flooding and the WP algorithm are very similar.

The result in Fig. 12 shows that when the size of network changes from 30K to 75K the average traffic of a query increases significantly, while the query traffic of BloomCast and the WP algorithm are quite stable.

Fig. 13 shows the efficiency of BloomCast in different network scales. When the network size increases from 30K to 75K, the improvement of efficiency increases from 72× to 93×. In most cases, the efficiency of BloomCast is better than that of the WP algorithm.

7 Conclusion

In this paper, we propose BloomCast, an efficient and effective full-text retrieval scheme, in unstructured P2P networks. BloomCast is effective because it guarantees the recall with high probability. It is efficient because the overall communication cost of full-text search is reduced below a formal bound. Furthermore, by replicating Bloom Filters instead of the raw documents across the network, BloomCast significantly reduces the communication cost for replication. We demonstrate the power of BloomCast design through both mathematical proof and comprehensive simulations based on the TREC WT10G data collection and query logs from a real world search engine. Results show that BloomCast outperforms existing schemes in terms of both search results quality and system efficiency.

Acknowledgments

This paper is supported by National Science Foundation of China and RGC Joint research fund under grant No.60731160630 and NSFC fund under grant No.60933011.

References

[1] D. Li, J. Cao, X. Lu, and K. Chen, "Efficient Range Query Processing in Peer-to-Peer Systems," *IEEE Trans. Knowledge and Data Eng.*, vol. 21, no. 1, pp. 78-91, Jan. 2008.
[2] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM '01*, pp. 149-160, 2001.
[3] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," *Proc. ACM SIGCOMM '02*. pp. 177-190, 2002.
[4] R.A. Ferreira, M.K. Ramanathan, A. Awan, A. Grama, and S. Jagannathan, "Search with Probabilistic Guarantees in Unstructured Peer-to-Peer Networks," *Proc. IEEE Fifth Int'l Conf. Peer to Peer Computing (P2P '05)*, pp. 165-172, 2005.
[5] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, "Fast Hash Table Lookup Using Extended Bloom Filter: An Aid to Network Processing," *Proc. ACM SIGCOMM*, 2005.
[6] C. Tang and S. Dwarkadas, "Hybrid Global-Local Indexing for Effcient Peer-to-Peer Information Retrieval," *Proc. First Conf. Symp. Networked Systems Design and Implementation (NSDI '04)*, p. 16, 2004.
[7] S. Robertson, "Understanding Inverse Document Frequency: On Theoretical Arguments for IDF," *J. Documentation*, vol. 60, pp. 503-520, 2004.
[8] P. Reynolds and A. Vahdat, "Efficient Peer-to-Peer Keyword Searching," *Proc. ACM/IFIP/USENIX 2003 Int'l Conf. Middleware (Middleware '03)*, pp. 21-40, 2003.
[9] F.M. Cuenca-Acuna, C. Peery, R.P. Martin, and T.D. Nguyen, "Planetp: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities," *Proc. 12th IEEE Int'l Symp. High Performance Distributed Computing (HPDC '03)*, pp. 236-246, 2003.

[10] H. Shen, Y. Shu, and B. Yu, "Efficient Semantic-Based Content Search in P2P Network," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 7, pp. 813-826, July 2004.

[11] H. Shen and C. Xu, "Locality-Aware and Churn-Resilient Load-Balancing Algorithms in Structured Peer-to-Peer Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 6, pp. 849-862, June 2007.

[12] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," *Proc. IEEE INFOCOM '03,* 2003.

[13] M. Li, W.-C. Lee, A. Sivasubramaniam, and J. Zhao, "SSW: A Small-World-Based Overlay for Peer-to-Peer Search," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 6, pp. 735-749, June 2008.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM '01,* pp. 161-172, 2001.

[15] G.S. Manku, "Routing Networks for Distributed Hash Tables," *Proc. ACM 22nd Ann. Symp. Principles of Distributed Computing (PODC '03),* pp. 133-142, 2003.

[16] V. King and J. Saia, "Choosing a Random Peer," *Proc. ACM 23rd Ann. Symp. Principles of Distributed Computing (PODC '04),* pp. 125-130, 2004.

[17] B.H. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *Comm. the ACM,* vol. 13, no. 7, pp. 422-426, 1971.

[18] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based vs. Structural," *Proc. ACM SIGCOMM '02,* pp. 147-159, 2002.

[19] A. Medina, A. Lakhina, I. Matta, and J.W. Byers, "BRITE: An Approach to Universal Topology Generation," *Proc. Ninth Int'l Symp. Modeling, Analysis and Simulation of Computer and Telecomm. Systems, (MASCOTS '01),* 2001.

[20] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-Like P2P Systems Scalable," *Proc. ACM SIGCOMM '03,* pp. 407-418, 2003.

[21] C. Huang, J. Li, and W. Ross, "Can Internet Video-on-Demand Be Profitable?," *Proc. ACM SIGCOMM,* 2007.

[22] S. Saroiu, P.K. Gummadi, and S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking (MMCN '02),* pp. 156-170, 2002.

[23] J.P.C. Jie Lu, "Content-Based Retrieval in Hybrid Peer-to-Peer Networks," *Proc. 12th Int'l Conf. Information and Knowledge Management (CIKM),* pp. 199-206, 2003.

[24] E.M. Voorhees, "Overview of Trec-2009," *Proc. 16th Text REtrieval Conf. (TREC-11),* 2009.

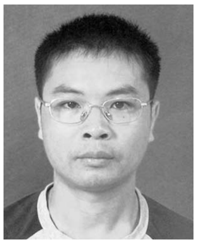
[25] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Math.,* vol. 1, no. 4, pp. 485-509, 2004.

[26] X. Tang, J. Xu, and W. Lee, "Analysis of TTL-Based Consistency in Unstructured Peer-to-Peer Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 12, pp. 1683-1694, Dec. 2008.

[27] N.F. Huang, R. Liu, C.H. Chen, Y.T. Chen, and L.W. Huang, "A Fast Url Lookup Engine for Content-Aware Multi-Gigabit Switches," *Proc. 19th Int'l Conf. Advanced Information Networking and Applications (AINA),* 2005.

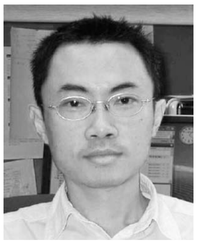
Hanhua Chen received the PhD degree in computer science and engineering from Huazhong University of Science and Technology (HUST), in 2010. He is now a faculty member at the School of Computer Science and Technology in HUST. He worked at the Hong Kong University of Science and Technology as a postdoctoral research associate between 2009 and 2010, and as a visiting scholar between 2007 and 2009. His research interests include peer-to-peer computing and wireless sensor networks. He is a member of the IEEE.

Hai Jin received the PhD degree in computer engineering from HUST in 1994. He is a Cheung Kung Scholars chair professor of computer science and engineering at the Huazhong University of Science and Technology (HUST) in China. He is now a dean of the School of Computer Science and Technology at HUST. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. He is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientist of National 973 Basic Research Program Project of Virtualization Technology of Computing System. He is the member of Grid Forum Steering Group (GFSG). He has co-authored 15 books and published more than 400 research papers. His research interests include computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security. He is the steering committee chair of International Conference on Grid and Pervasive Computing (GPC), Asia-Pacific Services Computing Conference (APSCC), International Conference on Frontier of Computer Science and Technology (FCST), and Annual ChinaGrid Conference. He is a member of the steering committee of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), the IFIP International Conference on Network and Parallel Computing (NPC), and the International Conference on Grid and Cooperative Computing (GCC), International Conference on Autonomic and Trusted Computing (ATC), International Conference on Ubiquitous Intelligence and Computing (UIC). He is a senior member of the IEEE and a member of the ACM.



Xucheng Luo received the PhD Degree in computer science from the University of Electronic Science and Technology of China, in 2008. He worked at the Hong Kong University of Science and Technology as a visiting scholar in 2007. He is currently an assistant professor in the School of Computer Science and Technology at the University of Electronic Science and Technology of China. His research interests include peer-to-peer computing and mobile social networks.

Yunhao Liu (SM'06) received the BS degree in automation from Tsinghua University, Beijing, China, in 1995, and the MS and PhD degrees in computer science and engineering from Michigan State University, in 2003 and 2004, respectively. Being a member of Tsinghua National Lab for Information Science and Technology, he also holds Tsinghua EMC chair professorship. He is the director of Key Laboratory for Information System Security, Ministry of Education, and professor at School of Software, Tsinghua University. He is also a faculty member at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include peer-to-peer computing, pervasive computing, and sensor networks. He serves the vice chair for Association for Computing Machinery (ACM) China Council, and he is currently the ACM distinguished speaker. He is a senior member of the IEEE.



Tao Gu received the PhD degree in computer science from National University of Singapore. He is currently an assistant professor in the Department of Mathematics and Computer Science at the University of Southern Denmark. His research interests include pervasive computing and wireless sensor networks. He is a member of the IEEE and ACM.

Kaiji Chen received the BS degree in computer science from Huazhong University of Science and Technology, in 2010. He is currently working toward the PhD degree in computer science at the University of Southern Denmark.

Lionel M. Ni is a chair professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology (HKUST). He also serves as the special assistant to the President of HKUST, dean of HKUST Fok Ying Tung Graduate School, and visiting chair professor of Shanghai Key Lab of Scalable Computing and Systems at Shanghai Jiao Tong University. He has chaired more than 30 professional conferences and has received six awards for authoring outstanding papers. He is a fellow of the IEEE.