A Multi-tiered Architecture for Content Retrieval in Mobile Peer-to-Peer Networks

Neelanjana Dutta, Raghavendra Kotikalapudi, Abhinav Saxena and Sriram Chellappan
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65409, USA.
Email: {nd2n8, rkyvb, abhinav.saxena}@mail.mst.edu, chellaps@mst.edu

***Abstract*—In this paper, we address content retrieval in Mobile Peer-to-Peer (P2P) Networks. We design a multi-tiered architecture for content retrieval, where at Tier 1, we design a protocol for content similarity governed by a parameter α that trades accuracy with search overhead. At Tier 2, we introduce a novel concept called Chained Bloom Filters and design a protocol where popular search items are linked with popular content at each node in an efficient manner for subsequent retrieval. Extensive analysis and numerical simulations demonstrate the effectiveness of our techniques.**

***Keywords*-P2P Networks, Mobility, Content Management**

I. Introduction

Mobile Peer-to-Peer (P2P) networks are popular avenues for sharing information and services. In these networks, mobile users form a distributed ad hoc network and exchange information over the wireless medium. Applications include emerging participatory sensing networks [1] mobile social networks [2], [3], vehicular networks [4], [5] etc.

Our Motivations: In this paper, we address content retrieval in mobile p2p networks. For queries issued, we have two objectives: reduce system overhead in searching for accurate content, and retrieve popular content in the system related to query issued. In content retrieval, there is a trade-off between user satisfaction (i.e., accuracy of content retrieved) and overhead. Unfortunately, in existing techniques, searches for queries at each node is a best effort process, and in the worst case, the entire database is searched. From a scalability perspective, search overhead can be tremendous. We aim to reduce the search overhead founded on two observations: a) Based on past knowledge of searches, the system can derive some intelligence on expected accuracy, and use this knowledge to limit wasteful searches for similar queries; b) Also, users may not always desire perfectly matching content, and if users can specify this in their queries, searching overhead can be significantly reduced. Furthermore, multiple users will share similar interests and hence issue similar queries. Since the search process involves multiple nodes, each node can recognize popular queries, and hence popular content. For any query issued, if this query is similar to popular queries serviced earlier, then corresponding popular content can be quickly retrieved.

In mobile p2p networks, communication overhead, a network centric metric has been well addressed [4]–[8]. However, we believe that search overhead at local nodes is also important. In large scale networks minor improvements in search overhead locally will have major impacts network wide. Our techniques developed here can complement works on network-centric overhead metrics for overall improvement in system performance.

Our Contributions: In this paper, we design a Multi-Tiered architecture and a protocols for content retrieval in mobile p2p networks. Tier 1 in our architecture is designed for reducing search overhead at each node while retrieving accurate content. The premise stems from the observation that for short queries, there is a higher chance of retrieving more accurate content. When queries get longer, the chances that highly accurate content can be found is lower. In this paper, we demonstrate how the trend of accuracy vs. query length follows a logistic function. With this knowledge, each node can make intelligent choices on when to stop searching the database beyond which more accurate content is unlikely to be found. Secondly, logistic functions are governed by a parameter α that determines function growth rate. Users can set this parameter based on desired accuracy of content requested, using which search overhead can be further reduced at each node.

Tier 2 is designed for retrieving popular content. In mobile p2p networks, when similar queries are issued by multiple nodes, it allows popular queries to be disseminated to multiple nodes. In Tier 2, we define a new metric called *Rank* for each content in the database of a node, where rank is computed based on popularity of its keywords. We then introduce a new concept called *Chained Bloom Filter*, where popular key words already processed by the node are linked to popular content in a space efficient manner. For new queries, we design a protocol that efficiently determines if key words requested in the query are popular (i.e., they are in the Filter), and if they are, it quickly returns the correspondingly linked popular content.

Our analysis shows that accuracy of content retrieved follows a logistic trend that can be captured with parameter α. We show that by allowing α to be user adaptive, search overhead reduces. We also study how our proposed rank and

Table I
AN EXAMPLE OF A DATABASE AT A LOCAL NODE

File 1	Beatles	mp3	Rock	English
File 2	Elvis Presley	mp3	Summer Kisses	English
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
File F	Target	Coupon	Labor Day	

chained bloom filter techniques are effective in both retrieval of popular content and saving overhead.

The paper is organized as follows. In Section II, we present preliminaries and important metrics. The proposed multi-tier architecture is presented in detail in Section III. Performance Evaluations are then presented in Section IV, and we conclude the paper in Section V.

II. PRELIMINARIES AND METRICS

Network Model: The mobile p2p network has mobile nodes communicating wirelessly. Each node has a database of content (i.e., files) containing meta-data describing content. An example of a database at a node is in Table I. Users issue queries containing key words. For each query issued, searches are conducted in the local vicinity of the requesting node (or user) to retrieve accurate content. In this paper, we emphasize on making the retrieval process adaptive via two critical ancillary goals: minimize search overhead based on past knowledge of searches, and retrieve popular (but still relevant) content for the query.

Content Similarity Metrics: A critical issue in content retrieval is similarity between a query and content (or file). There are a number of related metrics including te *Sorensen Similarity* metric, *Jaccard Coefficient* metric, and *Cosine Similarity* Metric [5], [9]–[11]. While each metric has its own applications, we focus on *Cosine Similarity*, which borrows from Vector Space Model (VSM). For D files in the database of a node, and each file tagged with upto n keywords per file, we represent each file as a row in a $D \times n$ matrix. Each file is then projected as a binary vector in a n-dimensional vector space. Any query can be treated as a vector in the space, and the similarity between the query and a file is then computed as angle (θ) between the query and the file vector. Formally, for a query q with keywords $\overrightarrow{q} = (q_1, q_2, q_3, \ldots, q_n)$, and a file f with keywords $\overrightarrow{f} = (f_{j1}, f_{j2}, f_{j3}, \ldots, f_{jn})$, the similarity between q and f denoted as $\theta_{q,f}$ (in degrees) is

$$\begin{aligned} \theta_{q,f} &= Cos^{-1}\left(\frac{\overrightarrow{q} \odot \overrightarrow{f}}{|\overrightarrow{q}| \cdot |\overrightarrow{f}|}\right) \qquad (1) \\ &= Cos^{-1}\left(\frac{\sum_{i=1}^{n} q_i f_{ji}}{\sqrt{(\sum_{i=1}^{n} q_i^2)} \times \sqrt{(\sum_{i=1}^{n} f_{ji}^2)}}\right). \end{aligned}$$

Naturally, smaller the value of $\theta_{q,f}$, more accurate is File f for Query q, and vice versa.

III. THE MULTI-TIERED ARCHITECTURE AND PROTOCOLS

A. Overview

The proposed architecture is comprised of 2 tiers: Tier 1 for reducing search overhead during accurate content retrieval, and Tier 2 for efficiently retrieving popular content. Note that there are two situations in which a node (say Node A) receives a query to process. Either the local user of Node A issues a query, or Node A receives a query from a neighbor. In either case, a node receiving a query first processes the query at Tier 1, where its local database is searched. The novelty of Tier 1 is a technique that minimizes search overhead based on prior searches, such that at a slight cost on accuracy, a significant amount of search overhead can be saved. The query is then processed at Tier 2, where we design a Chained Bloom Filter technique to efficiently store and retrieve popular content based on processing queries for other nodes. Results from both tiers are then forwarded to the query issuing node.

B. Tier 1 - Reducing Search Overhead

Content in a mobile p2p network is identified by a set of metadata. While some content can have many descriptors, others may only have less descriptors. Since the amount and the nature of metadata varies from user to user, this negates attempts to index the database. Consequently, for any Query q arriving at a node, the worst case searching time is $O(D) \times \bar{t}$, where D is the no. of database entries, and $\bar{t}$ is the processing time to find the similarity between Query q and a File f. This imposes a tremendous overhead for a single query, which increases for multiple queries. Note that due to the ad hoc nature of p2p systems, while some users be specific about content desired, others can be general. The former case happens when the no. of keywords requested in the query is more, and the latter happens when no. of keywords is small. Naturally, when query lengths are smaller, it possible to return more accurate content, and vice versa. Formally, θ (our similarity metric) increases (i.e., accuracy decreases) for increasing query lengths due to (likely) decreased keyword matches between queries and files (from Equation 1). However, the growth in the increase of the θ metric becomes progressively slower due to the Cos^{-1} function. Based on this intuition, we conjecture that *Content Retrieval in ad hoc environments like mobile p2p networks follows the trend of a Logistic Curve in terms of Accuracy vs. Query Length.*

We have conducted an extensive simulation study to validate this conjecture, and results are in Table II. In each case, we obtain the best θ values for varying query lengths via an exhaustive search of the database, and tried to fit a curve to best θ vs. Query Length. In Table II, D is the No. of Files in the Database; f_l is the maximum range of the No. of keywords in each file in the database; q_l is

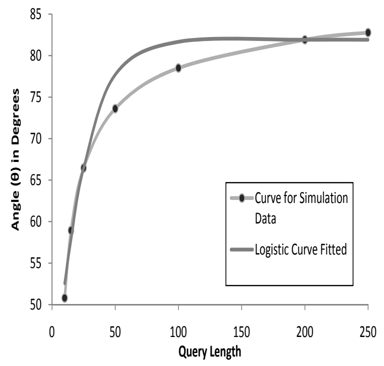

Figure 1. The Trend of Logistic Function

Table II
THE LOGISTIC FUNCTION FOR VARIOUS SYSTEM PARAMETERS

Database Parameters	Logistic Function	RMSE
$D = 10000$ f_l from 1 to 4	$86.99 \times L(0.033q_l)$	4.33
$D = 12000$ f_l from 1 to 4	$86.37 \times L(0.042q_l)$	4.16
$D = 10000$ f_l from 1 to 6	$86.37 \times L(0.029q_l)$	4.57
$D = 12000$ f_l from 1 to 6	$86.38 \times L(0.03q_l)$	4.48

query length; and RMSE is the root mean squared error between θ (derived via simulations), and θ obtained from the Logistic Function correspondingly shown in Table II, which was derived via symbolic regression techniques (a variant of Genetic Algorithms) to fit the curve. Each simulation was conducted 100 times and averaged. Note that $L(x)$ in Table II is the standard Logistic Function $L(x) = \frac{1}{1+e^{-x}}$. As, we can see the RMSE is quite low demonstrating the fidelity of the Logistic Function for Accuracy (θ) vs. Query Length. For ease of comprehension, we illustrate the trend of θ obtained via simulations for the case when $D = 10000$ and f_l is from 1 to 4 (first entry in Table II), and the Logistic function fitted for this case in Figure 1. As we can see the fidelity of the Logistic Function fit is quite high. Similar trends hold for other cases as well, not shown here.

To the best of our knowledge, this is the first work that identifies the logistic trend in content retrieval in p2p networks. At Tier 1, we exploit this trend for reducing search overhead with minimal compromise to accuracy. Each node first derives the Logistic Trend within its own database. The node can do this via prior knowledge of searches, or via periodic random sampling of the local database. We generalize Logistic Functions derived in Table II as $LF(q_l) = \beta \times L(\alpha \times q_l)$, where q_l is the length of the incoming query (See Table II). Each node first derives α and β for its local database. For any incoming query, each node first determines the expected best accuracy of search results for the length of that query q_l, and then deriving $LF(q_l)$. The node will then search the database to find a matching file, and it will stop searching once a file is found that is less than or equal to the expected best accuracy.

Discussions: A critical issue in Tier 1 design is α which decides logistic function growth, and which each node derives to return the expected level of accuracy. However, users can also manipulate α. When α is set high for a query, the growth of the Logistic Function increases, meaning users prefer a lower accurate file (higher θ), and more overhead savings, and vice versa. Thus α provides another leverage for users to adaptively choose accuracy at a cost of overhead. We study this further using Simulations in Section IV.

C. Tier 2 - Retrieving Popular Content

While Tier 1 focused on accuracy with overhead savings, Tier 2 emphasizes on popularity. Some queries (i.e., keywords) may be very popular, and when similar queries are asked, our goal is to not only focus on accuracy, but also to quickly and efficiently retrieve popular content in the database relevant to that query. The principle behind our popularity scheme is to assign a metric called *Rank* for each file in the database, where *Rank* is a function of how popular the keywords in that file are. Each node maintains the *Rank* for every file in its database independently. As keywords are serviced by the node, they are efficiently hashed into a novel *Chained Bloom Filter* (CBF), along with popular files for these keywords. When new queries come, the corresponding keywords are compared with those in the Filter, and corresponding popular files are quickly returned.

Past access frequency of a keyword: The past access frequency ($\mu_n(k)$) captures popularity of a keyword in terms of no. of times the keyword has been queried in the past, and also how recently it was requested. For a keyword k at time t_n it is computed every $\triangle t$ time interval as

$$\begin{aligned}\mu_n(k) \quad &= \quad \lambda^{n-1}N_1(k) + \lambda^{n-2}N_2(k) + \ldots \\ &\quad +\lambda^{n-i+1}N_{i-1}(k) + N_n(k),\end{aligned} \tag{2}$$

where $N_i(k)$ is the No. of times keyword k has been queried at the i^{th} instance within the past $\triangle t$ time interval ($1 \leq i \leq \triangle t$), and λ is a damping factor λ ($0 < \lambda < 1$) denoting popularity decrease of a keyword with time.

Rank of a file: We now address now address file popularity. Our *Rank* metric captures file popularity based on frequency and time of keyword requests. A file whose keywords have been asked often recently is more desirable to users who issue queries with similar keywords. One naive technique for computing Rank could be to sum up past access frequencies of all keywords in the file. However, this technique favors files with more keywords. W compute the Rank of a file as the average of access frequencies of keywords in the file. Note that the Rank computation is dynamic and is re-computed for each file in $\triangle t$ time intervals. Formally, the Rank of a file f with z keywords $\{f_1, f_2, f_3, \ldots, f_z\}$ at time instant n is

$$Rank_n(f) = \frac{\sum_{i=1}^{z} \mu_n(f_i)}{z}, \tag{3}$$

where $\mu_n(f_i)$ is defined in Equation 2. Note that a high rank for a File f means f has at-least one keyword that has been requested often recently. Thus, for a Query q with keywords matching keywords in File f, it is ideal to return File f as a popular file for Query q.

Our Chained Bloom Filter Technique: We now present our Chained Bloom Filter approach for storing and retrieving popular (highly ranked) content [1]. There are two operations involved in Tier 2: Updating the Chained Bloom Filter, and Retrieving content from the Chained Bloom Filter. We first discuss updation. When a query comes in, then a node will hash the keywords into a regular Bloom Filter with k hash functions and m bits. The node then computes the Rank for files that have at-least one of the key words in the query. The node determines the top x files in terms of Rank, and inserts the ids of these x files in another array of m bits at the same positions in the Bloom Filter that were set to 1. The respective positions in both arrays are linked to each other, leading to the term *Chained Bloom Filter*. We consider the list of file IDs chained against each bloom filter bit as a bucket for that bit position. To summarize, our Chained Bloom Filter technique efficiently links prior keywords searched at a local node with relevant popular content in that node for quick retrieval. We next present the details of our scheme, followed by the analysis.

We now discuss how to search the Chained Bloom Filter. For a Query q arriving at Tier 2 of a node, the node first checks if at-least one keyword in the Query q is present in the Filter. If not, then the keywords were never serviced by the node, and there is nothing to retrieve at Tier 2. Otherwise, for each position in the Bloom filter where the bit is set as 1 corresponding to every keyword's hash, the node retrieves the linked files linked in the corresponding bucket. The intersection of all such buckets is then considered to be the popular file(s) corresponding to at-least one keyword in the query and is returned to the node.

One issue in the proposed scheme is memory limitations at a node. With time, more queries arrive, and more keywords are hashed. Importantly, since the bucket chained to each bit will also have memory limitations, there will be a limit of the no. of file ids stored. In our scheme, we ensure that when buckets are full, and files have to be replaced to accommodate new queries, the newer files must have equal or higher rank than current files. Otherwise, they are deferred from being added to the Chained Bloom Filter until their Rank becomes more than those currently in the Filter.

[1]An excellent survey of Bloom Filters can be found in [12].

1) Analysis of Tier 2: We now conduct an analysis of our Tier 2 architecture and our popularity aware content retrieval protocol in terms of: retaining popular files in the filter, and probability of returning relevant files. We denote $P(X)$ as the probability that Statement X is true. We denote r_f as the Rank of file f. There are k hash functions during hashing of a keyword in the Bloom Filter. We denote N as total no. of files in the database, and c is the capacity of each bucket, which is the no. of file ids that can be stored in it.

$P_f^{f'}$ **Probability of a file f' is not present in the filter at a node when File f is present, where $Rank(f') < Rank(f)$:** It is easy to see that $P(f'$ is not in Filter when f is in the filter$)$ $=$ $P($keywords of f' is not hashed in same buckets as keywords in $f)$ $\times$ $P($keywords of f' is hashed in buckets where all files are ranked higher that $r_f{}')$.

To derive the worst case probability, we assume that all files have a minimum of l keywords. We also assume that in total there are S files in the node's database whose ranks are higher than rank(f'). We also assume that all l keywords of a file can be hashed to k' distinct buckets at the minimum. In worst case $k' = 1$. So the probability of keywords of f' is not hashed in same buckets as keywords in f is,

$$P_1 = \frac{\binom{m-k'}{k'}}{\binom{m}{k'}}. \tag{4}$$

Similarly, the probability that keywords of f' is hashed in buckets where all files are ranked higher that r' is,

$$P_2 = \frac{\binom{S}{c}^{m-k'} - \sum_{i=1}^{m-k'} \sum_{j=1}^{\frac{m-k'}{k'}} \left(\binom{S}{j} \times \binom{m-k'}{k+i}\right)^{j}}{\binom{N-1}{c}^{m-k'} - \sum_{i=1}^{m-k'} \sum_{j=1}^{\frac{m-k'}{k'}} \left(\binom{N-1}{j} \times \binom{m-k'}{k+i}\right)^{j}}. \tag{5}$$

Consequently, we have

$$P_f^{f'} = P_1 \times P_2. \tag{6}$$

Probability of returning files irrelevant files to a query If a query and a file have no keywords in common with each other, the file is called irrelevant to the query. This could happen in Bloom Filter based designs due to the inevitability of False Positives. We study this probability here.

Let i be the number of keywords in a current Query q_{cur} and ϵ $(i \geq \epsilon)$ be the maximum number of keywords matching in a cached query with the incoming query. Let us consider Query q_{arb} as the arbitrary cached query for which results were returned in response to q_{cur}, which is a False Positive. So the probability that a keyword of q_{arb} is also hashed to the same bits as query q_{cur} is,

$$\begin{aligned} &= \frac{k}{m}(m-i-1)\left(\frac{k-1}{m-1}\right)\left(\frac{k-2}{m-2}\right)\cdots\left(\frac{1}{m-k+1}\right) \\ &= (n-i-1)\frac{k!}{(m-k+1)!} \end{aligned} \tag{7}$$

Table III
SIMULATION PARAMETERS AND VALUES

Parameter	**Default Values**
Simulation Time	120 units
Simulation area	15×15 sq. units
No. of nodes	100
Communication Range	1 unit
No. of files per node	$1000 - 2000$
Total No. of Keywords	50
No. of keywords in file	$2 - 8$
No. of keywords in query	$2 - 8$
Wait time at a point	$5 - 15$ units
No. of nodes querying at an instant	$1 - 5$
No. of bits in bloom filter	20
Capacity of each bucket	10
No. of hash functions	3
Damping factor (λ)	0.8
Number of top ranked files (x)	2
Neighborhood searched	5 hops

For files associated to q_{arb} to be returned, this process has to repeat at-least ϵ times. So the probability becomes,

$$\binom{ki}{\epsilon k}\left[(n-i-1)\frac{k!}{(m-k+1)!}\right]^{\epsilon} \tag{8}$$

So the probability of returning only irrelevant files is,

$$\sum_{j=i}^{i-\epsilon}\binom{ki}{k(\epsilon+j)}\left[(n-i-1)\frac{k!}{(m-k+1)!}\right]^{\epsilon+j} \tag{9}$$

Note that in the above expression if $\epsilon = 0$, it represents the case of false positive, which means no relevant query has been cached yet, still the system returns some non-relevant files. For $\epsilon > 0$ it means that there are relevant files in the system, but non-relevant files are returned instead of them.

IV. PERFORMANCE EVALUATION

A. Simulation Set-up

We consider 100 nodes following Random Way Point Model in a 15×15 square unit area. Each node stores a number of files with keywords. The size of each file is considered as a single memory unit. Every node also maintains a chained bloom filter. Nodes generate queries containing keywords. Every query is processed at Tier 1, Tier 2 and forwarded to nodes within 5 hops. Results returned are consolidated from multiple node searches. Default parameters are listed in Table III.

B. Performance Evaluation of Tier 1

In Figures 2, 3, we study performance of search protocol in Tier 1. In all the Figures, the term PS stands for the *Proposed Search Technique* (in Section III), while ES stands for the baseline *Exhaustive Search Technique*, where the entire database is searched.

Figure 2 shows that with more files in the database, search overhead increases since there are more options. Figure 2 also shows that with increasing files, our protocol for Tier 1 reduces search overhead compared to exhaustive search. With more files, our proposed technique converges, since time taken to find expected similarity tends to grow very slowly beyond a point. While there is not much appreciable difference, queries with longer keywords converge slightly faster, again for the same reason that for longer queries highly accurate results are difficult to find resulting in faster convergence. In Figure 3, we study how accuracy (θ) is sacrificed when saving overhead. As we see, the worst case error between our technique and exhaustive search is around $9°$, which is acceptable when compared to significant savings in overhead, especially as error reduces for increasing files in the database at each node.

C. Performance Evaluation of Tier 2

In Figure 4, we study miss rate as a function of query rank (where rank is based on popularity of keywords in the query from Equation 5). The miss rate is the percentage of time a relevant file was not found in the Chained Bloom Filter. As we see, when queries contains more popular keywords (above 80^{th} percentile), miss rate is very low, and it increases when queries contain unpopular keywords. This demonstrates effectiveness of our technique in retaining popular files in the Chained Bloom Filter. In Figure 5, we study the issue of False Positives, where we plot the number of times an irrelevant file was returned from our Chained Bloom Filter based on no. of Bloom Filter bits. Note that by irrelevant files, we mean files that did not contain any of the keywords in the query. We see that as the number of Bloom Filter increases, the number of irrelevant files returned goes down dramatically. This trend is significant considering that no. of files in the database was 2000, and even a small addition of bits can significantly lower False Positives as database size increases.

D. Performance Evaluation of Tier 1 w.r.t. α

Finally, we study the impact of letting the user modify α in Figures 6 and 7. When α is set low, the search overhead increases, along with the accuracy, while when α increases the reverse happens. This trend has important impacts particularly from the perspective of pricing and incentive management in p2p systems. As we can see, while increased search overhead does in bring in improved accuracy of search, the relationship is not linear. Should the system decide pricing mechanisms based on accuracy or should it decide based on the overhead? How can the system resolve in tradeoff in designing optimal pricing and incentive management schemes by exploiting this trend is part of future work. This problem is more challenging when users themselves can change the parameter α.

V. CONCLUSIONS

In this paper, we address content retrieval in mobile p2p networks by first modeling the retrieval process as a Logistic

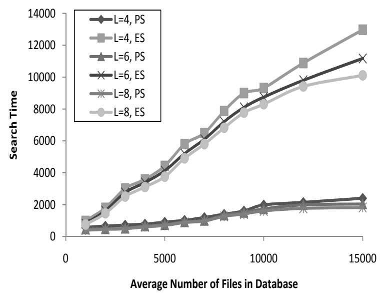

Figure 2. Search Time vs. Avg. No. of Files in Database for Different Query Lengths

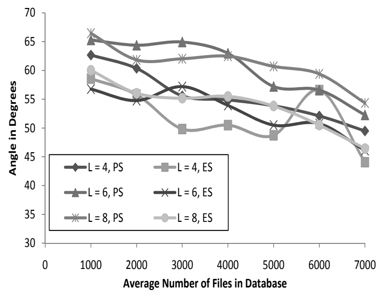

Figure 3. θ vs. Avg. No. of Files in Database for Different Query Lengths

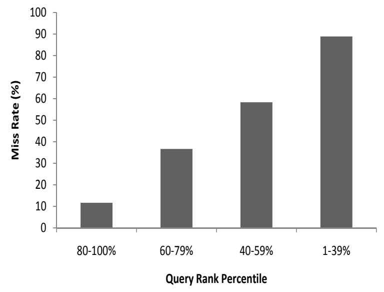

Figure 4. Miss Rate vs. Query Rank Percentile

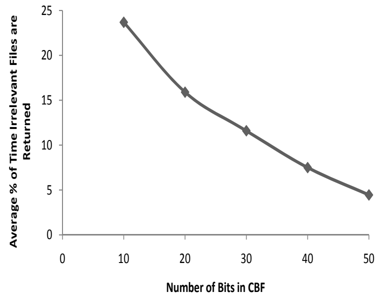

Figure 5. Percentage of Irrelevant Files vs. No. of Bloom Filter Bits

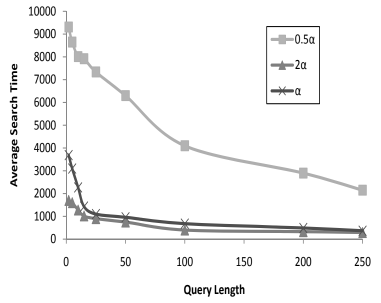

Figure 6. Search Time vs. Query Length for Different α

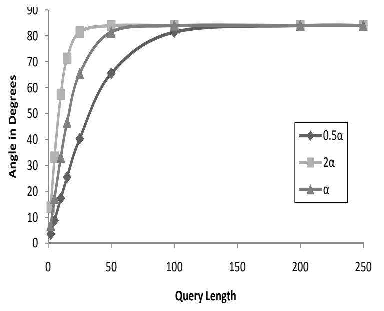

Figure 7. θ vs. Query Length for Different α

Function for overhead minimization, and designing a novel Chained Bloom Filter technique for popularity management. On-going work considers incentive and pricing schemes.

ACKNOWLEDGMENTS

The authors would like to thank University of Missouri Research Board, and University Transportation Center at Missouri S&T for funding part of this research.

REFERENCES

[1] J. Burke, D. Estrin, M. Hansen, and et. al., "Participatory sensing," in *Proceedings of Workshop on World-Sensor-Web (WSW06)*, Colorado, USA, 2006.

[2] E. Miluzzo, N. Lane, K. Fodor, R. Peterson, and et. al., "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *Conference on Embedded network sensor systems*. ACM, 2008.

[3] L. Humphreys, "Mobile social networks and social practice: A case study of Dodgeball," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 341–360, 2008.

[4] J. Zhao and G. Cao, "VADD: Vehicle-assisted data delivery in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 3, pp. 1910–1922, 2008.

[5] Y. Zhang, J. Zhao, and G. Cao, "Roadcast: A popularity aware content sharing scheme in vanets," in *Proceedings of IEEE ICDCS*, Canada, June 2009.

[6] L. Chen, B. Cui, H. Shen, W. Lu, and X. Zhou, "Efficient information retrieval in mobile peer-to-peer networks," in *Proceedings of ACM CIKM*, 2009, pp. 967–976.

[7] M. Fiore, C. Casetti, and C. Chiasserini, "Efficient retrieval of user contents in MANETs," in *IEEE Infocom*, 2007.

[8] T. Repantis and V. Kalogeraki, "Data dissemination in mobile peer-to-peer networks," in *Proceedings of ACM MDM*, 2005.

[9] G. Erkan and D. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, no. 1, pp. 457–479, 2004.

[10] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, no. 1, 2006, p. 775.

[11] T. Hasegawa, S. Sekine, and R. Grishman, "Discovering relations among named entities from large corpora," in *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2004, p. 415.

[12] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.