Efficient Approximate Query Processing in P2P Network

Amol P. Bhagat
Department of Computer Science and Engg, Prof. Ram Meghe College of Engineering and Magmt, Badnera, 444701 India

P. P. Pawade
Department of Computer Science and Engg, Sipna College of Engineering and Technology, Amravati, 444601 India

V. T. Gaikwad
Department of Computer Science and Engg, Sipna College of Engineering and Technology, Amravati, 444601 India

ABSTRACT
Peer-to-peer (P2P) databases are becoming prevalent on the Internet for distribution and sharing of documents, applications, and other digital media. The problem of answering large-scale ad hoc analysis queries, for example, aggregation queries, on these databases poses unique challenges. Exact solutions can be time consuming and difficult to implement, given the distributed and dynamic nature of P2P databases. In this paper, we presented novel sampling-based techniques for approximate answering of ad hoc aggregation queries in such databases. Computing a high-quality random sample of the database efficiently in the P2P environment is complicated due to several factors: the data is distributed (usually in uneven quantities) across many peers, within each peer, the data is often highly correlated, and, moreover, even collecting a random sample of the peers is difficult to accomplish. To counter these problems, proposed approach will uses approach based on random walks of the P2P graph, as well as block-level sampling techniques.

General Terms
Databases, Query Processing.

Keywords
Peer-to-peer Network, Query Processing, Distributed Databases.

1. INTRODUCTION
Peer-to-peer (P2P) network is increasingly becoming popular because it offer opportunities for real-time communication, ad-hoc collaboration [1] and information sharing [2] in a large-scale distributed environment. Peer-to-peer computing is defined as the sharing of computer resources and information through direct exchange. The most distinct characteristic of P2P computing is that there is symmetric communication between the peers; each peer has both a client and a server role. The advantages of the P2P systems are multi-dimensional; they improve scalability by enabling direct and real-time sharing of services and information; enable knowledge sharing by aggregating information and resources from nodes that are located on geographically distributed and potentially heterogeneous platforms; and, provide high availability by eliminating the need for a single centralized component. The most compelling applications on P2P systems are file sharing & retrieval. For example, P2P systems such as Napster [3], Gnutella [4], KaZaA [5], and Freenet [6] are popular for their file sharing capabilities, for example sharing of songs, music & so on.

1.1 Structured P2P
Structured P2P network (such as Pastry [3] and Chord [4]) is organized in such a way that data items are located at specific nodes in the network, and nodes maintain some state information to enable efficient retrieval of the data. This organization maps data items to particular nodes and assumes that all nodes are equal in terms of resources, which can lead to bottlenecks and hot spots.
Structured P2P networks employ a globally consistent protocol to ensure that any node can efficiently route a search to some peer that has the desired file, even if the file is extremely rare. Such a guarantee necessitates a more structured pattern of overlay links. By far the most common type of structured P2P network is the distributed hash table (DHT), in which a variant of consistent hashing is used to assign ownership of each file to a particular peer, in a way analogous to a traditional hash table's assignment of each key to a particular array slot.

1.2 Unstructured P2P
The paper focuses on unstructured P2P network, which makes no assumption about the location of the data items in the node, and nodes are able to join the system at random times and depart without a priori notification. Several recent efforts have demonstrated that unstructured P2P network can be used efficiently for multicast distributed object location and information retrieval [5].
An unstructured P2P network is formed when the overlay links are established arbitrarily. Such networks can be easily constructed as a new peer that wants to join the network can copy existing links of another node and then form its own links over time.
In particular, three models of unstructured architecture can be distinguished:

a) In pure peer-to-peer network peers act as equals, merging the roles of clients and server. In such networks, there is no central server managing the network, neither is there a central router. Some examples of pure P2P Application Layer networks designed for peer-to-peer file sharing are gnutella (pre v0.4) and Freenet.
b) Hybrid peer-to-peer systems which distribute their clients into two groups client nodes and overlay nodes. Typically, each client is able to act according to the momentary need of the network and can become part of the respective overlay network used to coordinate the P2P structure. As examples for such networks can be named modern implementations of gnutella (after v0.4) and Gnutella2.
c) In centralized peer-to-peer systems are networks using on the one hand central server(s) or bootstrapping mechanisms, on the other hand P2P for their data transfers. These networks are in general called

'centralized networks' because of their lack of ability to work without their central server(s). An example for such a network is the eDonkey network (often also called eD2k).

It has been observed that in most typical data analysis and data mining applications, timeliness and interactivity are more important considerations than accuracy. Thus, data analysts are often willing to overlook small inaccuracies in the answer, provided that the answer can be obtained fast enough. This observation has been the primary driving force behind the recent development of AQP techniques for aggregation queries in traditional databases and decision support systems.

2. AVAILABLE SYSTEMS

P2P systems are becoming very popular because they provide an efficient mechanism for building large scalable systems [6]. Most recent work has focused on Distributed Hash Tables (DHTs). Such techniques provide scalability advantages over unstructured systems (such as Gnutella); however, they are not flexible enough for some applications, especially when nodes join or leave the network frequently or change their connections.

Swarup Acharya, Phillip and Viswanath Poosala developed a system called Approximate QUary Answering (AQUA) [1] providing fast, approximate answers to aggregate queries, which are very common in OLAP applications. It has been designed to run on top of any commercial relational DBMS. Aqua precomputes synopses (special statistical summaries) of the original data and stores them in the DBMS. It provides approximate answers by rewriting the queries to run on these synopses. Finally, Aqua also incrementally keeps the synopses up-to-date as the database changes.

PIER and Piazza are two well-known data management systems running on top of peer-to-peer architectures. PIER is a general-purpose relational query processor designed to scale to millions of participating nodes on the Internet [7]. Piazza addresses many of the challenges associated with data sharing among many peering data providers [8]. Neither PIER nor Piazza describes explicit support for top-k query processing.

Methods to sample random peers in P2P network have been proposed in [8]. These techniques use Markov-chain random walks [7, 9] to select random peers from the network. Their results show that when certain structural properties of the graph are known or can be estimated (such as the second eigenvalue of the graph), the parameters of the walk can be set so that a representative sample of the stationary distribution can be collected with high probability. There are known techniques for computing approximate aggregates [8] in distributed settings (most notably, the Gossip protocol [5], [6]). The technique works generally as a preprocessing step where all peers in a network attempt to mix data among adjacent peers, eventually converging upon a single value. The inability to contact all nodes in the network makes it exceedingly difficult to Gossip in the traditional sense.

Nowadays, the two main approaches have emerged for constructing P2P networks: structured and unstructured. Several recent efforts have demonstrated that unstructured P2P networks can be used efficiently for multicast distributed object location and information retrieval [10], [11], [12]. For AQP in unstructured P2P systems, attempting to adapt the approach of precomputed samples is impractical for several reasons:

a) It involves scanning the entire P2P repository, which is difficult,
b) Since no centralized storage exists, it is not clear where the precomputed sample should reside, and
c) The very dynamic nature of P2P systems indicates that precomputed samples will quickly become stale, unless they are frequently refreshed.

Thus, the approach here is to investigate the feasibility of online sampling techniques for AQP on P2P databases. However, online sampling approaches in P2P databases pose their own set of challenges. To illustrate these challenges, consider the problem of attempting to draw a uniform random sample of n tuples from such a P2P database containing a total of N tuples. To ensure a true uniform random sample, sampling procedure should be such that each subset of n tuples out of N should be equally likely to be drawn. However, this is an extremely challenging problem due to two reasons:

a) Picking even a set of uniform random peers is a difficult problem, as the query node does not have the Internet Protocol (IP) addresses of all peers in the network. This is a well-known problem that other researchers have tackled (in different contexts) by using random-walk techniques on the P2P graph [13], [14], [15]. That is, where a Markovian random walk is initiated from the query node that picks adjacent peers to visit, with equal probability and under certain connectivity properties, the random walk is expected to rapidly reach a stationary distribution. If the graph is badly clustered with small cuts, then this affects the speed at which the walk converges. Moreover, even after convergence, the stationary distribution is not uniform; in fact, it is skewed toward giving higher probabilities to nodes with larger degrees in the P2P graph.
b) Even if we could select a peer (or a set of peers) uniformly at random, it does not make the problem of selecting a uniform random set of tuples much easier. This is because visiting a peer at random has an associated overhead; thus, it makes sense to select multiple tuples at random from this peer during the same visit. However, this may compromise the quality of the final set of tuples retrieved, as the tuples within the same peer are likely to be correlated. For example, if the P2P database contained listings of, say, movies, then the movies stored on a specific peer are likely to be of the same genre. This correlation can be reduced if we select just one tuple at random from a randomly selected peer; however, the overheads associated with such a scheme will be intolerable.

.

3. OUR APPROACH

Each peer is connected to a set of other peers in the network via uni-directional links, that is, each peer can locally select the other peers it wishes to link to. The distinguish between two types of links:

• Neighbor links connect a peer p to a set of other peers (p's neighbors) chosen at random, as in typical Gnutella-like networks.

• Acquaintance links connect a peer p to a set of other peers (p's acquaintances) chosen based on common interests.

Each peer has a bounded number of neighbor and acquaintance links. It can be called as p's friends the set of peers that have p among their acquaintances. The number of friends of a peer is its in-degree (which is unbounded, let alone by the size of the network).

A peer can make some of its local files accessible to other peers. Peers that do not share any file are called free riders. Non-free-riders or serving peers are those peers that contribute files to the community. A successful request yields a list of peers that have a file matching the original query.

Assume that, when several peers have the desired file, the peer that is closest to the requester (in number of hops) is chosen. I call that peer the answerer. Note that answering a query typically implies sending a file to the requester.

Some of the mechanisms can be introduced for shortly require peers to maintain state information about their friends. The state of a peer consists of the list of the names of its shared files. For load-balancing purposes, peers also need to know the in-degree of their friends.

To illustrate these definitions, consider the sample network depicted in Fig. 1, in which each peer has a single neighbor and acquaintance link. Peer p1 shares 99 files and its state consists of the names of all these files. It has p9 as random neighbor; p4 as acquaintance; and p2, p3, p4, p6, and p8 as friends, which corresponds to an in-degree of 5. Peers p7 and p9 are free-riders and have no friends. As it is seen here, a high in-degree generally indicates that a peer shares many files, or is well-connected to peers that share many files; in contrast, free-riders typically have a null in-degree. Pair-wise acquaintance relationships between serving peers that have similar interests (e.g., between p1 and p4) are also common in practice, and effectively yield bi-directional links.

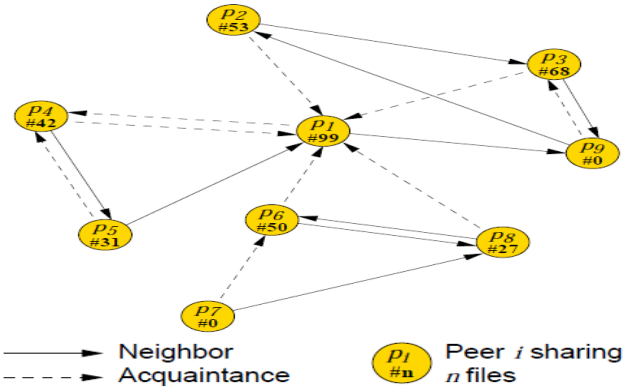

Fig 1: Sample minimal network, with a single neighbor & acquaintance link per peer

Essentially, here trying to pick true uniform random samples of the tuples, as such samples are likely to be extremely impractical to obtain. In the first phase, initiate a fixed-length random walk from the query node. This random walk should be long enough to ensure that the visited peers represent a close sample from the underlying stationary distribution. Then retrieve certain information from the visited peers, such as the number of tuples, the aggregate of tuples (for example, SUM, COUNT, AVG, and so forth) that satisfy the selection condition, and send this information back to the query node. This information is then analyzed at the query node to determine the skewed nature of the data that is distributed across the network, such as the variance of the aggregates of the data at peers, the amount of correlation between tuples that exists within the same peers, the variance in the degrees of individual nodes in the P2P graph and so on. Once this data has been analyzed at the query node, an estimation is made on how much more samples are required so that the original query can be optimally answered within the desired accuracy, with high probability. The second phase is then straightforward: A random walk is reinitiated, and tuples are collected according to the recommendations made by the first phase. Effectively, the first phase is used to "sniff" the network and determine an optimal-cost "query plan," which is then implemented in the second phase.

3.1 Peer-to-Peer Model

Here, it is assumed that an unstructured P2P network represented as a graph G = (P, E) , with a vertex set P = { p_1, p_2……, p_M} and an edge set E. The vertices in P represent the peers in the network, and the edges in E represent the connections between the vertices in P. Each peer p is identified by the processor's IP address and a port number (IP_p and $port_p$). The peer p is also characterized by the capabilities of the processor on which it is located, including its CPU speed p_{cpu}, memory bandwidth p_{mem}, and disk space p_{disk}. The node also has a limited amount of bandwidth to the network, noted by p_{band}. In unstructured P2P networks, a node becomes a member of the network by establishing a connection with at least one peer currently in the network. Each node maintains a small number of connections with its peers: The number of connections is typically limited by the resources at the peer. We denote the number of connections that a peer is maintaining by p_{conn}.

3.2 Random Walks

The simulation of a random walk. or more generally a Markov chain is a fundamental algorithmic paradigm with highly sophisticated and profound impact in algorithms and complexity theory. Furthermore. it has found a wide range of applications in such diverse fields as statistics, physics, artificial intelligence, vision, population dynamics, bioinformatics, among others.

A Markov-chain random walk is a procedure that is initiated at the query node, and for each visited peer, the next peer to visit is selected with equal probability from among its neighbors (and itself and, thus, self loops are allowed). It is well known that if this walk is carried out long enough, then the eventual probability of reaching any peer p will reach a stationary distribution. To make this more precise, let P = {p_1, p_2…… p_M}be the entire set of peers, let E be the entire set of edges, and let the degree of a peer p be deg(p). Then, the probability of any peer p in the stationary distribution is

$$prob(p) = deg(p) / 2|E|$$

It is important to note that the above distribution is not uniform. The probability of each peer is proportional to its degree. Thus, even if it can be achieved this distribution efficiently, it will have to compensate for the fact that the distribution is skewed as above if used samples have to drawn from it for answering aggregation queries.

3.3 Gossip based protocols

Gossip-based (or epidemic) protocols are emerging as an important communication paradigm. In gossip-based protocols, each node contacts one or a few nodes in each round (usually chosen at random), and exchanges information with these nodes. The dynamics of information spread bear a resemblance to the spread of an epidemic [16, 17], and lead to high fault tolerance and "self-stabilization" [18, 17, 19]. Gossip-based protocols usually do not require error recovery mechanisms, and thus enjoy a large advantage in simplicity, while often incurring only moderate overhead compared to optimal deterministic protocols, such as the construction of data dissemination trees. The guarantees obtained from gossip are usually probabilistic in nature; they achieve high stability under stress and disruptions, and scale gracefully to a huge number of nodes. In comparison, traditional techniques have absolute guarantees, but are unstable or fail to make progress during periods of even modest disruption.

3.4 Aggregation Queries

Aggregation queries have the potential of finding applications in decision support, data analysis, and data mining. Decision support applications such as On Line Analytical Processing

(OLAP) and data mining tools for analyzing large databases are gaining popularity. Executing such applications on large volumes of data can be resource intensive. Fortunately, small samples of the data can be used by data mining and statistical techniques effectively without significantly compromising the accuracy of their analysis. Likewise, OLAP servers that answer queries involving aggregation can potentially benefit from the ability to use sampling.

Aggregation query :

SELECT Agg-Op(Col) FROM T WHERE selection-condition

In the above query, the Agg-Op may be any aggregation operator such as SUM, COUNT, AVG, and so on, Col may be any numeric measure column of T or even an expression involving multiple columns, and the selection condition decides which tuples should be involved in the aggregation.

COUNT:

Here, the first phase is broken up into the following main components. First, a random walk is performed on the P2P network, attempting to avoid skewing due to graph clustering and vertices of high degree. The walk skips j nodes between each selection to reduce the dependency between consecutive selected peers. As the jump size increases, our method increases overall bandwidth requirements within the database, but for most cases, small jump sizes suffice for obtaining random samples.

Second, aggregates of the data is computed at the peers and send these back to the query node. Here somewhat a simpler approach is taken in which take fix a constant t such that if a peer has at most t tuples, then its database is aggregated in its entirety, whereas if the peer has more than t tuples, then t tuples are randomly selected and aggregated. Subsampling can be more efficient than scanning the entire local database, for example, by block-level sampling, in which only a small number of disk blocks are retrieved. If the data in the disk blocks are highly correlated, then it will simply mean that the number of peers to be visited will increase, as determined by our cross validation approach at query time.

Third, the CVError of the collected sample is estimated and use that to estimate the additional number of peers that need to be visited in the second phase.

SUM and AVERAGE:

Although the algorithm has been presented for COUNT queries, it can be easily extended to other aggregates such as the SUM and AVERAGE by modifying the y(Curr) value specified on line 8, phase 1 of the algorithm. For the SUM, no changes are required, and for the AVERAGE, (#tuples/#processTuples) is removed from y(Curr), since no scaling is required.

MEDIAN:

For more complex aggregates such as estimation of medians, quantiles, and distinct values, more sophisticated algorithms are required. In addition to computing COUNT, SUM, and AVERAGE aggregates, I can also efficiently estimate more difficult aggregates such as the MEDIAN. I propose an algorithm for computing the MEDIAN in a distributed fashion based upon comparing the rank distances of medians of individual peers. Our algorithm for computing the MEDIAN is given as follows:

1. Select m peers at random by using random walk.
2. Each peer sj computes its median medj and sends it to the query node, along with prob (s_j).
3. The query node randomly partitions the m medians into two groups of m=2 medians: Group1 and Group2.
4. Let medg1 be the weighted median of Group1, that is, such that the following is minimized:

$$abs\left(\sum_{\substack{med_j \in Group1,\\ med_j < med_{g1}}} 1/prob(s_j) - \sum_{\substack{med_j \in Group1,\\ med_j > med_{g1}}} 1/prob(s_j)\right).$$

5. Find the error between the median of Group2 (say, medg2) and the weighted rank of medg1 in Group2. That is, let

$$c = abs\left(\sum_{\substack{med_j \in Group2,\\ med_j < med_{g1}}} 1/prob(s_j) - \sum_{\substack{med_j \in Group2,\\ med_j < med_{g2}}} 1/prob(s_j)\right)/(m/2)$$

6. Select additional $c2/\Delta^2_{req}$ peers by using random walk.
7. Find and return the weighted median of the medians of the additional peers.

3.5 Hybrid Algorithm

In order to further improve the quality of our random sampling process, we have employed a hybrid sampling technique by allowing individually selected peers to perform additional sampling in parallel with the random sampling phase.

Since each peer is limited to knowledge of adjacent peers, computing aggregates based upon a single start location (query node) limits the total number of peers available for processing. Each peer accessed is aware of the peers that make up the path from the query node to itself. Most queries are unable to reach all peers in a network either due to high per-message cost or query execution time requirements. Techniques exist for generating expander P2P topologies [20]: A fully decentralized approach for computing random samples is impractical. We address the possibility of highly connected networks by randomly selecting adjacent peers as opposed to flooding. Since we cannot fully exploit a decentralized approach for query processing, we propose a hybrid solution for random sampling, focusing on extending our technique with a hybrid in-network decentralized approach.

Upon selection of a peer p_i by the random-walk phase, p_i contains a period p_i where further processing may be performed to improve the quality of a peer's local data. The period p_i is defined as the number of hops remaining in the random-walk phase before the final peer p_m is selected for sampling. In order to exploit these periods, we propose an incremental decentralized sampling technique building upon the Gossip protocol [21]. The number of messages sent over the network due to gossiping may be varied based upon the user-defined parameters r_a & r_r. Parameter r_a is the number of edges that a peer may randomly select for gossiping, and r_r is the maximum number of hops from p_i that gossiping is permitted. Regardless of the value of r_a or r_r, the number of messages sent to the query node remains constant. These two parameters combined allow the user to leverage in-network computation, without affecting the number of messages sent back to the query node, avoiding possible bottlenecks.

Mixing between peers increases the diffusion of values through the network. For our purpose, there is no specific constraint on the number of iterations required before exiting. Under our hybrid approach, the algorithm attempts to maximize the amount of mixing per peer p_i by exploiting the period before a peer must send a sample back to the query node. As stated in [21], the diffusion speed of the network can be represented as $T(n,\varepsilon) = O(\log n + \log 1/\varepsilon)$ for expander-type networks. In addition, for very long walks, convergence may occur before the final peer has been selected, but we can continue to perform gossiping, without loss of benefit, since P2P networks such as Gnutella [22] are, by nature, transient. Where peers are continually entering and leaving the network,

gossiping can continue to diffuse new values as peers enter the network.

Simply, since we know how many peers remain to be selected by the random-walk phase, the lower bound for the period p_{i_period} is the remaining number of hops required to obtain the required sample, given the specified jump size and sample size. For example, suppose a query is executed with the following parameters: jump size of 10, tuples per peer 100, and a sample size of 400. After selection of the first peer, at least 30 hops are required by the random walk before completion. For the first peer selected p1, the period p1 period is equal to 30 hops. This determines that for the next 30 hops, further processing can be utilized to improve the sample quality for the selected peer p1. Thus, for each consecutive peer selected, the period p_i period is bounded as the (jump size _ the number of remaining peers to be selected).

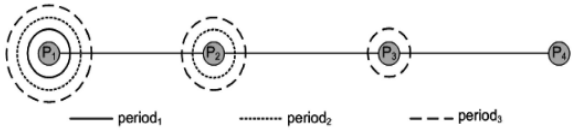

Fig 2: Each ring represents the increase in gossiping per period for each peer.

As shown in Fig. 1, the earlier that a peer is selected for sampling, the larger is the period available for gossiping. As additional hops are taken to reach the next peer for sampling, already selected peers can continue to gossip (this is represented by the rings around peers for each period). By combining our knowledge of Gossip and the p_{i_period} for selected peers, we can maximize the quality of the sample obtained from individual peers. Our hybrid sampling algorithm executes as follows:

1. Given a random start-location peer p_0, the local group is $\{ p_0 \}$.
2. Initialize a group for each selected peer p_i : $group_i \in \{ p_i \}$.
3. For each peer in $group_i$, randomly select r_a adjacent peers.
4. Extend the local group to include adjacent peers if and only if (path from $p_i \leq r_r$ $group_i \in group_i$, $^{\cup}$ { for each peer in $group_i$ add p_{i1} …..r_a}.
5. Perform Gossip on current $group_i$.
6. Continue steps 2-5 for each peer in $group_i$ until p_{i_period} has been reached.
7. All peers selected by the random sampling phase, excluding peers selected by the local groups, send their current mixed values back to the query node.
8. Compute remaining algorithm normally.

Peers near the beginning of the random walk have a longer period to gossip, whereas peers closer to the end of the walk contain an incrementally smaller period for gossiping. This creates an uneven level of mixing among the local groups of peers, but since all peers obey mass conservation as previously defined, the number of rounds performed by each group does not affect the overall results between the different gossiping groups.

4. CONCLUSIONS

In this paper, adaptive sampling-based techniques are presented for the approximate answering of ad hoc aggregation queries in P2P databases. This approach requires a minimal number of messages sent over the network and provides tunable parameters to maximize performance for various network topologies. The used approach provides a powerful technique for approximating aggregates of various topologies and data clustering but comes with limitations based upon given topologies structure and connectivity. For topologies with very distinct clusters of peers (small cut size), it becomes increasingly difficult to accurately obtain random samples due to the inability of random-walk process to quickly reach all clusters. This can be resolved by increasing the jump size, allowing a larger number of peers to be considered and increasing the allowed mixing by hybrid approach. By varying a few parameters, the given algorithm successfully computes aggregates within a given required accuracy.

5. REFERENCES

[1] S.Acharya, P.B.Gibbons, and V.Poosala, “Aqua: A Fast Decision Support System Using Approximate Query Answers,” Proc. 25th Int’l Conf. Very Large Data Bases (VLDB ’99), 1999.

[2] Adamic, R. Lukose, A. Puniyani, and B. Huberman, “Search in Power-Law Networks,” Physical Rev. E, 2001.

[3] B. Babcock, S. Chaudhuri, and G. Das, “Dynamic Sample Selection for Approximate Query Processing,” Proc. 22nd ACM SIGMOD Int’l Conf. Management of Data (SIGMOD ’03), pp. 539-550, 2003.

[4] A.R. Bharambe, M. Agrawal, and S. Seshan, “Mercury: Supporting Scalable Multi-Attribute Range Queries,” Proc. ACM Ann. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM ’04), 2004.

[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Analysis and Optimization of Randomized Gossip Algorithms,” Proc. 43rd IEEE Conf. Decision and Control (CDC ’04), 2004.

[6] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Gossip and Mixing Times of Random Walks on Random Graphs,” Proc. IEEE INFOCOM ’05, 2005.

[7] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya, “Towards Estimation Error Guarantees for Distinct Values,” Proc. 19th ACM Symp. Principles of Database Systems (PODS ’00), 2000.

[8] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and V. Narasayya,“Overcoming Limitations of Sampling for Aggregation Queries,” Proc. 17th IEEE Int’l Conf. Data Eng. (ICDE ’01), pp. 534-542, 2001.

[9] S. Chaudhuri, R. Motwani, and V. Narasayya, “Random Sampling for Histogram Construction: How Much Is Enough,” Proc. ACM SIGMOD Int’l Conf. Management of Data (SIGMOD ’98), pp. 436-447, 1998.

[10] Y. Chu, S. Rao, and H. Zhang, “A Case for End System Multicast,” Proc. ACM Int’l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS ’00), 2000.

[11] X. Li, Y.J. Kim, R. Govindan, and W. Hong, “Multi-Dimensional Range Queries in Sensor Networks,” Proc. First ACM Int’l Conf. Embedded Networked Sensor Systems (SENSYS ’03), 2003

[12] D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos, “Exploiting Locality for Scalable Information Retrieval in Peer-to-Peer Networks,” Information System, vol. 30, no. 4, pp. 277-298, 2005.

[13] Gkantsidis, M. Mihail, and A. Saberi, "Random Walks in Peerto- Peer Networks," Proc. IEEE INFOCOM '04, 2004.

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

[16]

[15] C. Faloutsos, P. Faloutsos, and M. Faloutsos, "On Power-Law Relationships of the Internet Topology," Proc. ACM Ann. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '99), 1999.