01 May 2007

ConQuer: A Peer Group-Based Incentive Model for Constraint Querying in Mobile-P2P Networks

Anirban Mondal

Sanjay Kumar Madria
Missouri University of Science and Technology, madrias@mst.edu

Masaru Kitsuregawa

Recommended Citation

A. Mondal et al., "ConQuer: A Peer Group-Based Incentive Model for Constraint Querying in Mobile-P2P Networks," *Proceedings of the 2007 International Conference on Mobile Data Management*, Institute of Electrical and Electronics Engineers (IEEE), May 2007.
The definitive version is available at https://doi.org/10.1109/MDM.2007.27

ConQuer: A Peer Group-based Incentive Model for Constraint Querying in Mobile-P2P Networks

Anirban Mondal
Institute of Industrial Science
University of Tokyo
JAPAN.
Email: anirban@tkl.iis.u-tokyo.ac.jp

Sanjay Kumar Madria
Department of Computer Science
University of Missouri-Rolla
USA.
Email: madrias@umr.edu

Masaru Kitsuregawa
Institute of Industrial Science
University of Tokyo
JAPAN.
Email:kitsure@tkl.iis.u-tokyo.ac.jp

Abstract— **In mobile ad-hoc peer-to-peer (M-P2P) networks, economic models become a necessity for enticing non-cooperative mobile peers to provide service. M-P2P users may issue queries with varying constraints on query response time, data quality of results and trustworthiness of the data source. This work proposes ConQuer, which addresses constraint queries in economy-based M-P2P networks. ConQuer proposes a broker-based incentive M-P2P model for handling user-defined constraint queries. It also provides incentives for MPs to form collaborative peer groups for maximizing data availability and revenues by mutually allocating and deallocating data items using a royalty-based revenue-sharing method. Such reallocations facilitate MPs in providing better data quality, thereby allowing them to further increase their revenues.**

Keywords: Economic model, Mobile-P2P networks, Royalty model, data allocation, free-riding.

I. Introduction

In a Mobile Ad-hoc Peer-to-Peer (M-P2P) network, mobile peers (MPs) interact with each other in a peer-to-peer (P2P) fashion. Proliferation of mobile devices (e.g., laptops, PDAs, mobile phones) coupled with the ever-increasing popularity of the P2P paradigm (e.g., Kazaa, Gnutella) strongly motivate M-P2P network applications. Incidentally, mobile devices with support for wireless device-to-device P2P communication are beginning to be deployed such as Microsoft's Zune.

Suppose while driving through a busy street, John's car encounters a mechanical problem, thereby forcing him to stop. Hence, he *urgently* looks for a car towing and repairing service, which suggests that *timeliness* of data delivery is important to him. Here, *data quality*, which relates to the quality of information concerning car towing and repairing services, is also of major concern. *Trust* associated with the peer, from whom John would receive the data, is also important e.g., he would place more trust on a traffic policeman.

While cycling, suppose Jack receives a phone call from his ailing friend Jim, who urgently needs some prescription medication from a pharmacy. Moreover, his wife may send him a message asking him to visit a plumbing company urgently. At the same time, Jack is looking for a fast food restaurant as he is hungry. Thus, constraints on timeliness, data quality and trust are also applicable when users simultaneously request multiple pieces of information. Notably, our target applications mainly concern slow-moving objects e.g., cars on busy streets, people moving in a market-place or students in a campus. Moreover, our applications only consider *read-only* data items.

Users may issue queries with varying constraints on query response time, data quality of results and trustworthiness of the data source. For example, someone helping an injured person needs information concerning hospitals quickly, hence he places more emphasis on response time. However, a mobile user looking for a song (as in a future mobile eBay market) generally wants good data quality (i.e., good audio quality) and high trust (i.e., legal copyrighted song version). Multiple copies of a data item, such as a song, may exist at different MPs with varying constraint values e.g., different data quality and trust values. Such copies are *not* replicas since their constraint values differ. We shall henceforth use the term **copies** to distinguish such copies from **replicas**.

Incidentally, free-riding [3] is rampant in static P2P environments, thereby necessitating an *incentive-based model* to entice non-cooperative peers. For *value-added services* such as constraint querying in M-P2P networks, incentives become absolutely necessary to address limited energy, memory and bandwidth resources of MPs for ensuring efficient and timely query processing. However, existing M-P2P incentive schemes [8] are not adequate to handle constraint queries as they do not consider value-added routing services.

If a constraint query is processed by flooding the M-P2P network, the user may *not* obtain an answer within his desired timeframe and location due to mobility. On the other hand, if a constraint query is processed by issuing multiple queries with one constraint per query, the user could possibly receive too many results. To obtain fewer results, the user could select the first result or he could limit the TTL (Time-to-live) of the query, but this would not provide him the result satisfying his constraints at the cheapest price.

For facilitating constraint queries, we propose **ConQuer**, the main contributions of which are three-fold:

1) It proposes a broker-based incentive M-P2P model for handling user-defined constraint queries.
2) It provides incentives for MPs to form collaborative peer groups for maximizing data availability and revenues by mutually allocating and deallocating data items using a *royalty-based* revenue-sharing method. Such reallocations facilitate MPs in providing better data quality, thereby allowing them to further increase their revenues.
3) It discusses the CR*-tree, a dynamic multi-dimensional R-tree-based index which incorporates constraints related to the data quality, trust and price of data items for determining target peers efficiently.

1-4244-1241-2/07/$25.00 ©2007 IEEE

Each data item in ConQuer is associated with a *price* in terms of a *virtual currency*. Data item prices depend on data quality (e.g., image resolution, audio quality) and can be determined using our previous work in [5]. ConQuer requires a data-requesting MP to pay the *price* of the data item to the data-providing MP, which entices MPs to become service-providers. We define the **revenue** of an MP as the difference between the amount of virtual currency that it earns (by providing services e.g., providing data, relaying messages) and the amount that it spends (by requesting services). Virtual currency is suitable for P2P environments since transaction costs of micro-payments in real currency are generally high [7].

In ConQuer, MPs form groups to collaboratively reallocate data items using a royalty-based revenue-sharing method to maximize data availability. This also increases revenues of MPs as they make available higher quality data items by removing excess low quality copies. This also enables ConQuer to efficiently address queries involving co-related data items at a single MP or within few hops of each other. In contrast, MPs individually would store only the hot data items to maximize their own revenues, hence relatively less hot data items would become unavailable. This would cause queries with co-related data items to fail, thus resulting in lost revenues. Thus, ConQuer provides MPs with an incentive to maximize the revenue of the group as a whole, which encourages non-selfish behaviour among them. Existence of multiple peer groups ensures non-monopolistic pricing.

In ConQuer, peer collaboration is facilitated by broker MPs, which collect bids from data-providing MPs and then pass these bids to the query issuing user, who selects a single bid and pays a *commission* to the broker in the selected query path. Any MP can act as a broker to earn more revenue. Each broker dynamically creates and maintains its own CR*-tree index based on the queries that it intercepts so that it can efficiently target MPs for answering constraint queries. The CR*-tree indexes constraints in multi-dimensional space involving data quality, trust and price.

Our performance study indicates that ConQuer is indeed effective in answering constraint queries with improved response time, data availability and quality, and querying hop-counts.

II. Related Work

The incentive scheme in [3] does not consider economic models and brokerage to combat free-riding. Incentive mechanisms for static P2P networks assume peers' availability and fixed topology, which makes them too static to be deployed in mobile ad-hoc networks (MANETs). Interestingly, economic ideas for M-P2P networks have been discussed in [8], which proposes opportunistic dissemination of data in M-P2P networks, while we address on-demand data dissemination. Furthermore, brokerage is not considered in [8].

The **E-DCG+ approach** [2] for replica allocation in MANETs does not consider constraint queries, economic issues such as revenue models, incentives, prices of data items, and it does not determine the optimal number of required copies to maintain a reasonable response time.

III. Query Processing Architecture and Constraint Indexing in ConQuer

This section discusses the query processing architecture and constraint index of ConQuer.

Query processing architecture

Peer groups can be formed by existing methods. In ConQuer, any MP can act as a broker to facilitate constraint queries. For every query answered through itself, a broker obtains a *commission* from the query issuing MP, which provides an incentive for MPs to become brokers. Gateway MPs select themselves as brokers because they can earn more revenue by intercepting and facilitating queries arriving at or going out of their respective groups. In ConQuer, broker's commission is 10% of the price of each data item that it retrieves. To earn commission, brokers index the constraints of a subset of the data items stored within their own groups and try to keep as much information as possible about brokers in other groups. To optimize memory space for index storage, brokers decide which data items to index in several ways e.g., based on interests or commissions gained.

Each broker has an unique identifier, designated as *broker_id*, which it periodically broadcasts to all the MPs (including broker MPs) in its group. Periodically, the broker with the lowest value of *broker_id* is automatically selected as the **Master Broker (MB)** of its group. Periodically, all brokers in the group send access frequency information and failed query statistics to MB. MB uses this information to advise MPs about the objects to be reallocated periodically. Each MP makes available only few data items to be shared based on the amount of bandwidth that it would like to share, but it has additional data items in the memory, which can be made available during reallocation. We shall henceforth refer to the data items that an MP makes available as the **shared data items**, while the additional items in the MP's memory are called **unshared data items.** If an MP is currently located at the intersection of multiple groups, it can choose to be a part of one group for the purpose of allocation and deallocation of data items, but it can serve as a broker for multiple groups.

ConQuer specifies the value DQ of data quality of an item by adopting the ideas from our previous work in [4], which relates DQ to image resolution by considering three different levels of data quality, namely *high*, *medium* and *low*. Thus, the value of DQ is assigned as follows. For *high* data quality, $DQ \geq 0.8$; for *medium* quality, $0.5 \leq DQ < 0.8$; for *low* quality, $DQ < 0.5$. ConQuer computes the trust values of data items by adopting the proposal in [6], which proposes a light-weight decentralized reputation-based trust management mechanism for ad-hoc P2P networks. ConQuer assigns the trust value $Trust$ of a data item as follows: For *high* data trust, $Trust \geq 0.8$; for *medium* trust, $0.5 \leq Trust < 0.8$; for *low* trust, $Trust < 0.5$. Thus, $0 \leq DQ, Trust \leq 1$.

User queries Q are of the form $\{Q_{id}, (k_1,k_2,...,k_n), Response, DQ, Trust, \rho_{max}\}$. Q_{id} is the unique identifier of a query, and k_i are user-specified keywords. $Response$ is the maximum tolerable query response time. DQ is the range of

user-desired data quality, $Trust$ is the range of user's desired trust value and ρ_{max} is the *maximum price* that the user is willing to pay for obtaining the query result. A user specifies constraints based on his knowledge of queries that he relays.

A user broadcasts his constraint query Q. Each query has a TTL (time-to-live) of 8 hops. Non-broker MPs simply forward Q. A broker B_i receiving Q checks if its index contains the data item(s) required by Q and if so, it puts its *broker_id* into the query message and becomes the broker for Q; otherwise, it simply forwards the query. If a broker sees that a *broker_id* has already been appended to Q, it will simply forward Q. Thus, only a single MP can act as a broker in a given query path, thereby optimizing energy consumption of brokers. However, as Q can propagate along multiple paths, multiple brokers albeit in different paths could process Q, thereby providing better fault-tolerance against unavailability of some brokers.

Each broker B_i issues a route-finding query to locate the path to the target MPs likely to satisfy Q. Then B_i collects *bid* information concerning *price*, data quality and trust values for the queried data item(s) from each target MP, and then returns the *bid* information to the user. From these bids, the user selects the lowest-priced bid satisfying his constraints and obtains his queried data item(s) via the broker. Finally, the user pays the *broker commission* to the broker.

Indexing of Constraints in ConQuer

If each broker indexes only the data items stored at different MPs, constraint queries may be unnecessarily sent to MPs containing the queried data items, but not satisfying the user-specified constraints. This motivates constraint indexing. Query response time constraint cannot be indexed since it depends upon network conditions. But the constraints of *data quality*, *trust* and *price* can be indexed since they are relatively static as they depend solely upon the data item. Thus, constraints of each data item can be specified as a point in 3D-space, the dimensions being *data quality*, *trust* and *price*. Constraint indexes are constructed by brokers based on queries that they intercept, hence indexes may differ across brokers.

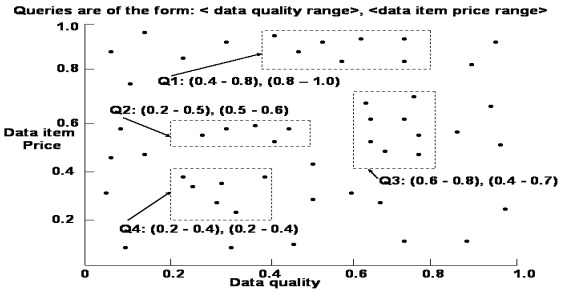


Fig. 1. Constraints in 2D-space

For clarity, Figure 1 provides the intuition concerning how to index constraints in 2D-space with *data item price* and *data quality* as the dimensions. Each point in Figure 1 indicates the constraints of a given data item, while $Q1$ to $Q4$ represent constraint queries with the queried ranges in parentheses. Since multiple data items may satisfy the range of user-specified constraints, such constraint indexing has the drawback of unnecessarily retrieving *non-queried* data items, which satisfy the respective ranges of user-specified constraints.

To satisfy our objective of retrieving only the *queried* data items that satisfy user-specified constraints, we propose the **CR*-tree (Constraint R*-tree)**, which is a multi-dimensional R*-tree-based [1] constraint index stored at each broker. The constraints of each data item are represented by a point in 3D-space with *data quality*, *trust* and *price* as the dimensions. The CR*-tree indexes this 3D-space. Non-leaf nodes of the CR*-tree contain entries of the form *(ptr, mbr*, LL) where *ptr* is a pointer to a child node in the CR*-tree and *mbr* is 3D-MBR (minimum bounding rectangle), which covers all the MBRs in the child node. *mbr* is of the form $\{(x_{min}, y_{min}, z_{min}), (x_{max}, y_{max}, z_{max})\}$, the first and second terms denoting the minimum and maximum values of the constraint parameters in 3D-space. We specify a given data item using one or more keywords. We shall henceforth use the terms *keywords* and *data items* interchangeably. LL is a linked list of such keywords within *mbr*. Entries in LL are sorted in dictionary order to facilitate efficient retrieval.

A leaf node of the CR*-tree is an array of 3D-MBRs. Each MBR contains entries of the form (*mbr*, LL_l), where the form of mbr is same as that of the non-leaf nodes. LL_l is a linked list containing entries of the form (*keyword, freq, arr_MP*), where *keyword* indicates the keyword of a given data item within *mbr*. LL_l is sorted in dictionary order of the keywords. *freq* is the number of times *keyword* occurs within *mbr*. *arr_MP* is an array of the MPs that store the data item. Thus, brokers process constraint queries by issuing 3D-window queries on the CR*-tree. Brokers can reject unrealistic user queries based on their knowledge.

IV. ConQuer: A Peer Group-based Incentive Model for Constraint Querying in M-P2P networks

This section first discusses a greedy method in which all MPs try to maximize their own revenues. To address the deficiencies in the greedy method, we propose the peer group-based method deployed by ConQuer for improving the overall data availability, data quality and MP revenues.

Greedy method: Care-About-Me (CAM)

Under the CAM method, each MP M tries to make available only those data items that will maximize its revenue. Let the data items stored at M constitute a list D. M sorts D in descending order of a parameter γ, which quantifies the revenue-earning potential of a given data item. M greedily fills up its available memory space with data items from D (starting from the item with highest value of γ) until it has no available memory space. While traversing D, if M encounters a data item, whose size is larger than its available memory space, it skips to the next item in D. Let access frequency and price of data item i in D be acc_i and ρ_i respectively. γ is computed as ($acc_i \times \rho_i / size_i$), where $size_i$ is the size of i.

Observe that CAM may lead to the duplication of the hot data items across several neighbouring MPs, while other items would become unavailable due to memory space constraints of the MPs, thereby decreasing overall data availability. This would also reduce the revenues of individual MPs due to the total revenue for the hot data items being divided among neighbouring MPs and/or due to query failures related to the unavailable data items. Existing approaches [2], which allocate replicas based on access frequencies of data items, cannot reconcile such redundancy due to greedy behaviour of MPs.

Peer group-based method: Care-about-Groups (CAG)

To address the deficiencies of the CAM method above, we propose CAG, which provides incentives for enticing MPs in a group to store different (possibly co-related) data items as well as improve data quality of existing items.

Estimating the number of copies for a data item: CAG needs to determine the number of copies for a given data item because eliminating all duplicates could cause queries with response time constraints to fail. Recall that all brokers in the group send data item access frequency information for each query intercepted by them to the designated **Master Broker** (**MB**) of the group. MB computes the total access frequency acc_i of i by summing up the individual access frequencies of i at each broker within the group. Given that $size_i$ is the size of i, MB initially computes the number K_i of copies for i as $(\sqrt{(size_i \times acc_i)})$. Then MB checks the response time constraints on the failed queries on i to determine the failed query (on i), which had the lowest response time constraint T_{min}. MB also determines the current average response time T_{avg} of queries on i. Then MB computes the optimal number K'_i of copies of i as follows:

$$K'_i = K_i(1 + \lceil (T_{avg} - T_{min})/T_{avg} \rceil) \quad (1)$$

where K'_i is the required number of copies of i, hence $(K'_i - K_i)$ additional copies of i need to be made available.

Royalty-based Revenue-sharing method for peer groups: Suppose n MPs in a group are currently making available the data item i. In practice, n would be typically much higher than K'_i (evaluated from Equation 1) since initially, most of the MPs would make available copies of the same hot data items to maximize their own revenues as in the CAM method. (If n equals K'_i, no action needs to be taken.) If n exceeds K'_i, we proceed as follows. Given that the access frequency and price of i at the j^{th} MP j are $acc_{i,j}$ and $\rho_{i,j}$ respectively, the revenue $Rev_{i,j}$ generated due to i at MP j equals $(acc_{i,j} \times \rho_{i,j})$.

The master broker MB sorts the MPs in descending order of their values of $Rev_{i,j}$, and selects the top-K'_i MPs from the sorted list. These top-K'_i MPs would make available i, while the remaining (n - K'_i) MPs would replace their copy of i and fill in the resulting available memory space with some of their unshared data item(s). (Recall that each MP has **shared** and **unshared** data items.) However, this may result in increasing the revenues of the MPs that make i available, while decreasing the revenues of the MPs which replaced i. We shall henceforth refer to the MPs that make i available and the MPs which replaced i as **store-MPs** and **replace-MPs** respectively. Users generally want high quality data items, hence access frequencies for high quality items is much higher than for low quality items. Thus, an MP storing a high quality copy of an item i generally earns higher revenues due to i than an MP that stores a low quality copy of i. Hence, CAG's selection of the top-K'_i revenue-earners for i essentially implies that during data reallocation, CAG replaces excess low-quality copies, while keeping the high quality copies, thereby implying that CAG improves the average data quality.

If the store-MPs pay a percentage of the revenues that they earned from i to the replace-MPs, it would not be economically viable. This is because replace-MPs would not agree to give up their 'hot' data items just for receiving only a small percentage of the revenues (as royalty) since they would want to earn as much revenue as they were earning earlier by storing the 'hot' data items. On the other hand, if the replace-MPs demand the amount of revenue that they were earning earlier from i, there would be no benefit for the store-MPs. Replace-MPs would earn some revenue by making available their previously unshared data items to fill up the available memory space arising from replacing i. To evaluate the royalty payment that must be paid by the store-MPs to a given replace-MP k, we compute the *difference* between the lost revenues of MP k (due to replacing i) and the revenues gained by MP k due to making available new data items.

Computation of lost revenue of a replace-MP k due to deallocating i: First, we estimate the future access frequency of i during the next reallocation period. Suppose $P_{i,k}$ is the running probability of accesses to data item i at MP k during the previous r reallocation periods (r = 4 was found to be a reasonable value for our applications), t is the time of latest access to i, and t_i was the time when i was accessed during the previous set of time periods. The running probability $P'_{i,k}$ of the data item i being accessed at a replace-MP k during the next periodic interval is computed as follows:

$$P'_{i,k} \; = \; (C/(t - t_i)) + (1 - C) \times P_{i,k} \quad (2)$$

where C is a constant quantifying how much emphasis is given to the previous probability of accesses to i when computing $P'_{i,k}$. Preliminary experiments revealed that C=0.5 is a reasonable value for our M-P2P application scenarios, hence we shall use C = 0.5 for this work. Thus, we compute the predicted access frequency $S'_{i,k}$ of i at MP k as $(P'_{i,k} \times prev_{acc_{i,k}})$, where $prev_{acc_{i,k}}$ is the previous access frequency of i at MP k during the previous period. Given that $\rho_{i,k}$ is the price of i at MP k, the lost revenue of k (for the next period) due to replacing i is $(S'_{i,k} \times \rho_{i,k})$. However, since the query issuing user would have to pay 10% broker commission for each query on i, MP k would have received only 90% of the price of i. Thus MP k's *actual* lost revenue due to replacing i would be $0.9(S'_{i,k} \times \rho_{i,k})$. For the next τ periods, MP k's total lost revenue LR due to replacing i is computed as follows:

$$LR \; = \; \Sigma_{t=1}^{\tau} \; 0.9 \times \; S'_{i,k} \times \rho_{i,k} \; (1 + \lambda)^{-t} \quad (3)$$

where λ is the percentage increase or decrease in the access frequency of the data item i (in the most recent period) as compared to the moving average of the previous set of reallocation periods. λ adjusts the royalty payment based on the predicted increase or decrease in access frequency in the next τ reallocation periods. Preliminary experiments showed that τ = 4 is a reasonable value for our application scenarios.

Computation of revenue gained by a replace-MP k due to making available some data items from its set of unshared data items: Recall that MPs have **shared** and **unshared** data items (see Section III). A given replace-MP k makes available some of its unshared data items to fill up the available memory space due to replacing i. MP k selects data items, which it wants to make available, by examining past access statistics to determine items on which queries had failed. Given that $\rho_{i,k}$ is the price of an unshared data item i at MP k and $acc_{i,k}$ is the number of times a query failed to obtain i during the last reallocation period, the lost revenue of the replace-MP k due to a failed query on i is $(\rho_{i,k} \times acc_{i,k})$. First, MP k sorts all its unshared data items in descending order of the revenues lost (due to failed queries) into a list L_f. Then it fills up its available memory space with data items from L_f (starting from the item with the highest value of lost revenue) until it has no available memory space. While traversing L_f, if MP k encounters a data item, whose size is larger than its available memory space, it simply skips to the next item in L_f.

For each (previously unshared) data item now made available by MP k, k computes its future access frequency during the next reallocation period using Equation 2, which can be used in this case because i was missed, which implies that i was accessed. Now, MP k computes its predicted revenue α gained due to making available new data items as follows:

$$\alpha = \Sigma_{i=1}^{p} \ (0.9 \times \rho_{i,k} \times acc_{i,k}) \tag{4}$$

where $\rho_{i,k}$ and $acc_{i,k}$ are the price and predicted access frequency of the i^{th} newly shared item for the next reallocation period, while p is the number of such newly shared items. The factor of 0.9 arises due to the 10% broker commission as explained for Equation 3. For the next τ periods, MP k's total gained revenue GR due to making available new data items is computed as follows:

$$GR \ = \ \Sigma_{t=1}^{\tau} \ \alpha \ (1+\lambda)^{-t} \tag{5}$$

where the significance of τ and λ are same as in Equation 3.

Computation of the royalty to be paid to replace-MP k by the store-MPs: Using Equations 3 and 5, the royalty revenue RY_i that must be paid by the store-MPs making i available to the replace-MP k, which replaced i, is computed as follows:

$$RY_i \ = \ LR \ - \ GR \tag{6}$$

Since access frequency increases in i at each store-MP cannot be predicted in advance, RY_i is equally divided among the store-MPs. Thus, if K_i' store-MPs make i available, each of these store MPs will pay (RY_i/K_i') to a given replace-MP k.

V. Performance Evaluation

In our experiments, MPs move according to the *Random Waypoint Model* within a region of area 1000 metres $\times$1000 metres. In all our experiments, 20 queries/second are issued in the network, the number of queries directed to each MP being determined by the Zipf distribution with Zipf factor (ZF) of 0.7. Bandwidth between MPs varies from 28 Kbps to 100 Kbps, while probability of MP availability varies from 50% to 85%. Data item sizes range from 50 Kb to 750 Kb, while memory space of each MP varies from 2 MB to 5 MB. Speed of an MP varies from 1 metre/s to 10 metres/s.

Our experiments consider 100 MPs. Each MP owns 4 shared data items and 4 unshared data items. (Unshared data items play a role only for CAG during reallocation.) Among the total of 400 shared data items, the number of unique data items is 40. (The number of unique unshared data items is also 40.) Hence, there are 10 copies per data item and these copies are assigned different constraint values of data quality and trust as follows. Given y different copies of the same data item, we generate the data quality mix by using the Zipf distribution with zipf factor of 0.7 over 3 buckets. The numbers generated for the first, second and third buckets are for *low*, *medium* and *high quality* respectively. The data trust mix is also generated similarly. These constraint values of data quality and trust are assigned randomly to the 10 copies of the same data item.

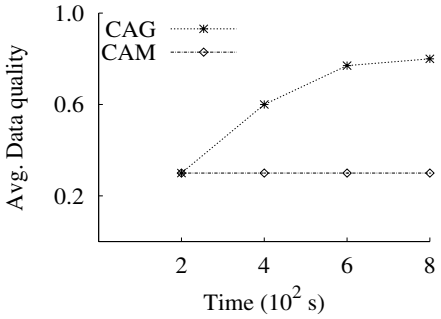
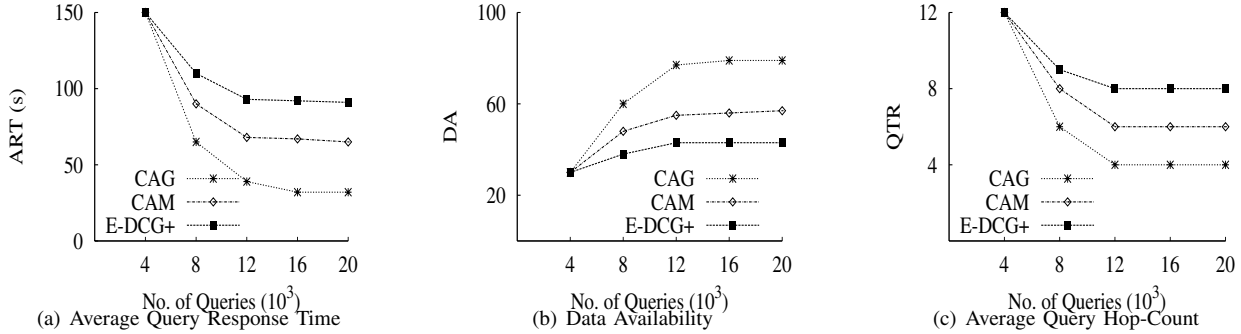


Fig. 2. Improvement of Data Quality

Data item(s) to be queried are selected randomly from the entire set of data items in the M-P2P network, the Zipf distribution being used for determining the number of queries corresponding to each of the data items. 60% of our queries involved single data items, while the other 40% concerned between 2 to 5 data items, as determined by a random number. Recall that data quality and trust are in the range (0,1). We normalized price values to be in the range of (0,1) by dividing all prices with the price of the highest item in the network. For each query, the constraints were generated by selecting a random number between 0 and 1 corresponding to each constraint. The response time constraint for each query was between 40 seconds and 120 seconds, the exact value being based on a random number. Communication range of all MPs is a circle of 100 metre radius. *Periodically*, every 200 seconds, the master broker MB decides whether to perform reallocation.



(a) Average Query Response Time (b) Data Availability (c) Average Query Hop-Count

Fig. 3. Performance of ConQuer

Performance metrics are **average response time** (**ART**) of a query, **data availability** (**DA**) and **average querying traffic** (**QTR**). Suppose T_i is the time of query issuing, T_f is the time of completion of download of the queried data item(s) at the query issuing MP, and N_Q is the total number of queries. **ART** equals $((1/N_Q)\sum_{i=1}^{N_Q}(T_f - T_i))$. Since our computation of ART includes the time required for downloading the queried data item(s), we consider ART only for *successful* queries. **DA** equals $((N_S/N_Q) \times 100)$, where N_S is the number of queries that were answered successfully, and N_Q is the total number of queries. **QTR** is the average number of hops per query.

Incidentally, none of the existing proposals for M-P2P networks address peer group-based incentive models. As reference, we adapt the **E-DCG+** approach [2] to our scenario since it is the closest to our proposed **CAG** method. E-DCG+ is executed at every reallocation period. We also compare **CAG** with **CAM**.

Improvement in data quality

We define average data quality as $((\Sigma D_i)/n_i)$, where D_i is the value of data quality for the i^{th} copy of a data item i, and n_i is the total number of copies of i in the network. We randomly selected a data item whose average data quality was low i.e., 0.3. CAG's peer group collaboration using the royalty-based revenue-sharing method facilitates MPs in improving their revenues by collaboratively removing excess low-quality copies during the reallocation of data items. Hence, the average data quality for CAG improves over time as indicated by the results in Figure 2. Since CAM does not perform such collaborative reallocation, its average data quality remains constant over time. This experiment does not consider E-DCG+, which does not address data quality issues.

Performance of ConQuer

Figure 3 depicts the performance of ConQuer. The first 4000 queries were used to obtain access statistics, and then the three approaches were deployed. ART decreases for each approach and eventually plateaus after some time due to each approach creating its own optimal number of copies of data items in response to the given access pattern. Both CAG and CAM outperform E-DCG+ as their incentive-based models encourage MP participation, hence their total available memory space and total bandwidth are higher, thereby implying higher DA. The CR*-tree used by CAG and CAM finds target MPs efficiently, so CAG and CAM incur lower QTR and ART than E-DCG+.

CAG outperforms CAM as its peer group collaboration via the royalty-based revenue-sharing method ensures better data allocation, which leads to higher DA. In contrast, CAM encourages MPs to make available only the hot data items, hence less hot data items become unavailable, thus constraint queries with co-related data items fail due to unavailability of some of the data items. CAG's allocation enables queries on co-related data items to be answered at a single MP or at multiple MPs that are within few hops of each other, which reduces its ART and QTR. Unlike CAM, CAG decides the number of copies for a given data item based on response time constraints posed by the queries, which optimizes its ART.

VI. Conclusion

We have proposed ConQuer, which addresses constraint queries in economy-based M-P2P networks. ConQuer proposes a broker-based incentive M-P2P model for handling user-defined constraint queries. It also provides incentives for MPs to form collaborative peer groups for maximizing data availability, data quality and revenues by mutually reallocating data items using a royalty-based revenue-sharing method.

References

[1] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. *Proc. ACM SIGMOD*, 1990.

[2] T. Hara and S.K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing*, 2006.

[3] S. Kamvar, M. Schlosser, and H. Garcia-Molina. Incentives for combatting free-riding on P2P networks. *Proc. Euro-Par*, 2003.

[4] A. Mondal, S.K. Madria, and M. Kitsuregawa. CADRE: A collaborative replica allocation and deallocation approach for Mobile-P2P networks. *Proc. IDEAS*, 2006.

[5] A. Mondal, S.K. Madria, and M. Kitsuregawa. EcoRep: An economic model for efficient dynamic replication in Mobile-P2P networks. *Proc. COMAD*, 2006.

[6] T. Repantis and V. Kalogeraki. Decentralized trust management for ad-hoc peer-to-peer networks. *Proc. MPAC*, 2006.

[7] D.A. Turner and K.W. Ross. A lightweight currency paradigm for the P2P resource market. *Proc. Electronic Commerce Research*, 2004.

[8] O. Wolfson, B. Xu, and A.P. Sistla. An economic model for resource exchange in mobile Peer-to-Peer networks. *Proc. SSDBM*, 2004.