**Peer-to-Peer Networking and Applications manuscript No.**
(will be inserted by the editor)

# An Economic Incentive Model for encouraging Peer Collaboration in Mobile-P2P networks with support for constraint queries

**Anirban Mondal · Sanjay Kumar Madria · Masaru Kitsuregawa**

**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. Hence, we propose **ConQuer**, which is an economic incentive model for the efficient processing of constraint queries in M-P2P networks. ConQuer also provides incentives for peer collaboration in order to improve data availability. The main contributions of **ConQuer** are three-fold. First, it uses a *broker-based economic* M-P2P model for processing constraint queries via a Vickrey auction mechanism. Second, it proposes the CR*-tree, a dynamic multidimensional R-tree-based index for constraints of data quality, trust and price of data to determine target peers efficiently. The CR*-tree is hosted by brokers, who can *sell* it to other peers, thereby encouraging the creation of multiple copies of the index for facilitating routing. Third, it provides incentives for peers to form collaborative peer groups for maximizing data availability and revenues by mutually allocating and deallocating data items using *royalty-based* revenue-sharing. Such reallocations facilitate better data quality, thereby further increasing peer revenues. Our performance study shows that ConQuer is indeed effective in answering constraint queries with improved response time, success rate and data quality, and querying hop-counts.

**Keywords** Mobile Peer-to-peer · economic model · royalty model · incentives · peer participation · constraint queries · constraint indexing

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

Sanjay Kumar Madria
Department of Computer Science
Missouri University of Science and Technology, USA
E-mail: madrias@mst.edu

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

## 1 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. Mobile devices with support for wireless device-to-device P2P communication are beginning to be deployed such as Microsoft's Zune.

The M-P2P paradigm has significant applications in social networking. Suppose John is looking for other car users with the objective of car-pooling so that he can reduce his expenses on gas. Hence, he could send a message to other nearby mobile users (e.g., people living in the same apartment complex) to locate prospective car-pooling partners, who follow similar paths and timings. Observe that he would need to find his car-pooling partners within a certain time-frame, thereby suggesting that *timeliness* of data delivery is important to him. Since John may not have any a priori information concerning his prospective partners, the *trust* associated with the peers, who reply to John's request for car-pooling, would also be important. Understandably, John would like to avoid untrustworthy partners as far as possible since such partners may not adhere to the car-pooling agreement. In a similar vein, cab-sharing among mobile users can also be facilitated by M-P2P interactions. For example, several mobile users at an airport may wish to go to the downtown area, where their respective hotels or business venues are located.

While walking around a shopping mall, Richard could issue a query for finding the cheapest available Levis jeans. Here, *query response time* is important to Richard because he needs the information while he is still somewhere near to the mall. Observe that if Richard receives a newly updated sales brochure on his mobile device, it is likely to be more useful to him than if the brochure was designed a few months ago. This is because the new brochure would contain updated prices and possibly provide a more updated view of the shop's inventory. Thus, *data quality* also carries significance to Richard. Indeed, mobile users could trade any items with each other by means of M-P2P interactions (as in a future *mobile eBay market*). For example, mobile users could share songs with each other in a P2P manner, and they could also trade PDF files and books with each other. Mobile users could also look for like-minded individuals at a conference or at a social gathering such as an art exhibition.

Observe that users may issue queries with varying constraints on query response time, data quality of results and trustworthiness of the data source. The weightage assigned to each constraint depends upon user preferences and the application under consideration. For example, when Jane is looking for a song, she could want good data quality (i.e., good audio quality) and high trust (i.e., legal copyrighted song version), but she may be willing to wait for the copy of the song. Thus, she would give higher weightage to data quality and trust as compared to response time. On the other hand, Jack may want to obtain the song quickly, and he may be willing to sacrifice data quality and trust in lieu of fast response time.

Multiple copies of a data item (e.g., 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**. Notably, higher trust in the data source may not necessarily imply better data quality e.g., a user may want a document from a trusted source, even if the document has not been recently updated. Our target applications mainly concern slow-moving objects e.g., people moving in a market-place or a conference venue, or students in a University campus.

Data availability in M-P2P networks is typically lower than in fixed networks due to frequent network partitioning arising from user movement and mobile devices switching 'off' when their generally limited energy is drained. (Data availability is less than 20% even in a wired environment [36].) *Non-incentive* replication schemes [20] for improving M-P2P data availability do not combat free-riding [18, 22]. Since a large percentage of the peers in P2P environments are typically free-riders [1] (i.e., they do not provide any data to the network), they are unlikely to collaboratively host copies of data. Furthermore, a small study [29], which was conducted on users' motivation and decision to share resources in P2P networks, revealed that 50% of the questioned users would share more, if some materialistic incentives (e.g., money) are dispensed by the application. Hence, an *economic incentive-based model* becomes necessary to entice free-riders to host data.

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. Incidentally, existing incentive schemes for mobile ad hoc networks [5, 6, 37, 8, 9] are designed for providing incentives to mobile nodes for forwarding messages. However, they do not provide any incentives for free-riders to host data and they do not address constraint queries. Moreover, existing M-P2P incentive schemes [42, 43] do not address constraint queries and they do not entice free-riders to host data. Furthermore, they address data dissemination, while we consider a query-based model (i.e., on-demand services).

If a constraint query is processed by flooding the M-P2P network, the query-issung MP may *not* obtain an answer within its desired time-frame and location due to mobility. On the other hand, if multiple queries are issued with one constraint per query, the query-issuing MP could possibly receive too many results. To obtain fewer results, it could select the first result or it could limit the TTL (Time-to-live) of the query, but this would not enable it to obtain the result satisfying its constraints at the cheapest price. (As we shall see shortly, a query issuing MP is required to pay a price for querying.)

For facilitating the efficient processing of constraint queries in M-P2P networks, we propose **ConQuer**, which is an economic incentive model for M-P2P networks. ConQuer also provides incentives for peer collaboration in order to improve data availability. In this paper, our goal is to design the *guidelines* for an economic incentive model for M-P2P networks with incentives for peer collaboration. We assume here that the peers are trusted and they do not cheat. However, trust and security issues have been handled in existing proposals. Our proposal can be used in conjunction with existing works [28, 27, 47], which handle P2P and mobile fairness issues e.g., ensuring the collection of payments, ensuring that source nodes get their desired service after they have made their payments, and the usage of tamper-proof hardware for handling payments. Furthermore, our proposal can also be used in conjunction with the works on secure virtual currency payments [10, 12, 48, 14]. Furthermore, there are existing works, which deal with P2P and mobile trust issues for controlling the deceiving behaviour of peers [23, 35, 8, 7]. In essence, we do not focus on trust and security issues in this paper, but these issues can be handled using the existing works discussed above.

The main contributions of ConQuer are three-fold:

1. It uses a *broker-based economic incentive* M-P2P model for processing user-defined constraint queries by means of a Vickrey auction mechanism.
2. It proposes the CR*-tree, a dynamic multidimensional R-tree-based index for constraints of data quality, trust and price of data to determine target peers efficiently. The CR*-tree

is hosted by brokers, who can *sell* it to other peers, thereby creating multiple copies of the index for facilitating routing.

3. It provides incentives for MPs to form collaborative peer groups for maximizing data availability and revenues by mutually allocating and deallocating data items using *royalty-based* revenue-sharing. Such reallocations facilitate better data quality, which allows MPs to further increase their revenues.

Each data item in ConQuer is associated with a *price* in *virtual currency*. Data item prices depend on data quality (e.g., image resolution, audio quality) and the amount of bandwidth that the data-providing MP makes available to the query-issuing MP for the download of the queried data item. ConQuer requires a data-requesting MP to pay the *price* of its queried item to the data-providing MP, a commission to the broker MP and a commission to the relay MPs in the successful query path. We define the **revenue** of an MP as the difference between the amount of virtual currency that it earns (by hosting data and indexes, relaying messages) and the amount that it spends (by requesting data). Thus, ConQuer provides an incentive for MPs to host data and indexes, and to relay messages so that they can earn revenue for issuing their own requests.

Notably, virtual currency is suitable for P2P environments since the transaction costs of micro-payments in real currency are generally high [40]. Similar to the motivation provided in [40], suppose MTV offers Jean 5 units of virtual currency per month (as she is a regular customer) if she agrees to store a video-clip and stream it on-demand to other peers 25 times in a market-place on a Sunday. She can use these units to buy some MTV products. The work in [12] discusses how to ensure secure payments using a virtual currency.

ConQuer does not assume a priori knowledge of the movement patterns of the mobile peers because in case of our application scenarios, mobile peers move randomly i.e., they do not follow a specific pre-defined movement pattern. However, we do assume that the brokers have limited mobility i.e., they only move within a specific radius. As an example of an application scenario, consider the case of a mobile user looking for the cheapest Levis jeans in a shopping district. In this example, the brokers are the shop-owners and they would generally stay within a threshold distance of their shops. On the other hand, the mobile user would move randomly within the shopping district, while looking for his/her queried item.

The next section provides a brief introduction to peer collaboration in ConQuer.

## 2 An overview of peer collaboration in ConQuer

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 MP, which selects a single bid and pays a *commission* to the broker MP in the selected query path. In ConQuer, brokers could be pre-designated in accordance with the application scenario under consideration, and there could be multiple pre-designated brokers. For example, recall our application scenario concerning an M-P2P user searching for the cheapest Levis jeans in a shopping district. In this application scenario, the shop-owners in the shopping area will act as brokers. Similarly, for the car-pooling application scenario, the manager of the apartment complex, where the prospective car-pooling users live, can act as the broker. If songs or movies are shared among M-P2P users in a University campus setting, some of the students (e.g., student organization leaders) can act as brokers. In the social networking setting of a conference, the conference organizers can act as brokers. Additionally, brokers can also be elected by means of an existing leader election protocol [24].

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 a multi-dimensional space involving data quality, trust and price. ConQuer allows a broker to sell its index to other MPs in lieu of an payment. The index-buyer MPs are incentivized to buy indexes since they can earn broker commissions by successfully answering queries via the copy of the index. Thus, this entices both the broker and the other MPs towards the creation of multiple copies of the index, which further improves data availability and query response times.

In ConQuer, 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.**

There are three methods in ConQuer, namely CAM (Care-About-Me), CAN (Care-About-Neighbours) and CAG (Care-About-Groups). CAM is a greedy method in which there is no collaboration among the MPs. In CAM, MPs host only the hot data items to maximize their revenues. Thus, CAM suffers from the disadvantage that it 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 due to query failures (resulting in lost revenues) related to the unavailable data items.

To address the deficiencies of CAM, we propose the CAN and CAG methods, in which MPs perform collaborative data reallocations using royalty-based revenue-sharing. While CAN collaboratively reallocates data iteratively among one-hop neighbours, CAG reallocates data in peer groups, which may extend beyond one hop. In CAG, MPs form groups [46] to collaboratively reallocate data items using a royalty-based revenue-sharing method to maximize data availability, thus *conquering* the business against other groups. This increases revenues of MPs as they make available higher quality items by removing excess low quality copies. In the absence of peer groups, MPs acting individually would host only hot items to maximize their own revenues, hence relatively less hot items would become unavailable and queries on them would 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. This also helps in building trust among MPs in a group due to increasing revenue, thereby creating a strong bond among strangers. Existence of multiple peer groups ensures non-monopolistic pricing. 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 since it can only abide by the rules of one group. However, an MP can serve as a broker or as a relay MP for multiple groups.

Our performance study shows that ConQuer is indeed effective in answering constraint queries with improved query response times, query success rates and querying hop-counts. In particular, we compare the performance of the three methods in ConQuer, namely CAG, CAN and CAM with the E-DCG+ approach [20] as reference.

The results show that CAG, CAN and CAM outperform E-DCG+ due to their economic nature resulting in better MP participation and also due to the fact that they use the efficient CR*-tree for efficiently directing queries to target data-providing MPs. CAG and CAN outperform CAM due to their royalty model-based collaboration among the MPs, which results in better data allocations as compared to that of CAM, in which MPs act individually without any collaboration among themselves. CAG and CAN also facilitate improvement of

data quality in the network by collaboratively removing excess low-quality copies of data items in favour of high quality copies during reallocation. CAG outperforms CAN due to its better view of the network i.e., in CAN, data reallocations occur only across one-hop neighbours, while in CAG, such reallocations occur across a group, which may extend beyond one hop. Finally, our proposed methods exhibit good scalability as more MPs imply better opportunities for making available copies of items.

The remainder of this paper is organized as follows. Section 3 discusses existing works, while Section 4 presents the architecture of the ConQuer economic model. Section 5 proposes the CR*-tree index and details the index selling mechanism by brokers. Section 6 presents ConQuer's peer group-based royalty model for encouraging peer collaboration in efficiently handling constraint queries. Section 7 reports our performance evaluation. Section 8 discusses some additional issues, which are relevant to our proposal. Finally, we conclude in Section 9.

## 3 Related Work

This section provides an overview of existing works.

**Non-incentive-based replication in Mobile ad hoc networks (MANETs):** The proposals in [20, 19] discuss replication in MANETs. **E-DCG+** [20] creates groups of MPs that are biconnected components in a MANET, and shares replicas in larger groups of MPs to provide high stability. An RWR (read-write ratio) value in the group of each data item is calculated as a summation of RWR of those data items at each MP in that group. Each replica is allocated at an MP, whose RWR value to the item is the highest among MPs that have free memory space to create it. The work in [19] aims at classifying different replica consistency levels in a MANET based on application requirements, and proposes protocols to realize them. Consistency maintenance is performed via quorums and it is based on local conditions such as location and time. P2P replication suitable for mobile environments has also been incorporated in systems such as Clique [34] and Rumor [17]. However, the proposals in [20, 19, 34, 17] are non-economic in nature, and they do not provide any incentives for the mobile nodes to host data and to forward messages.

**Incentive schemes for combating free-riding in MANETs:** The proposals in [5, 6, 37, 8, 9] discuss incentive schemes for combating free-riding in MANETs. The work in [5] introduces a virtual currency to stimulate cooperation among the nodes in a MANET. Each node behaves autonomously and aims at maximizing its benefits from the network. The work in [5] is extended in [6], which entices nodes to forward messages by means of a simple counter-based mechanism at each node. The work in [37] proposes a distributed algorithm, which is utilized by the nodes to determine whether to accept or reject a relay request.

In [8], an auction-based incentive scheme, designated as iPass, has been proposed to entice nodes in MANETs to forward packets. The market price of the packet forwarding service is paid to the relay nodes. The work in [9] provides incentives to users for acting as transit nodes on multi-hop paths by rewarding nodes according to their ability to send messages. The work in [49] proposes incentive-compatible protocols for both routing and packet forwarding in wireless ad-hoc networks from a game-theoretic perspective. Using both incentive mechanisms and security techniques, routing protocols for deterministic link models and probabilistic link models have been designed. Furthermore, an efficient forwarding protocol, which is based on the use of hash chains in cryptography, has been proposed to deliver payments. Observe that the works in [5, 6, 37, 8, 9, 49] essentially focus on providing incentives to mobile nodes for forwarding messages. However, they do not provide any

incentives for free-riders to host data. Furthermore, they do not consider prices of data items and different prices based on data requested and queries answered.

**Incentive schemes for combating free-riding in M-P2P networks:** The work in [43] provides incentives to MPs for participation in the dissemination of reports about resources in M-P2P networks. Each disseminated report contains information concerning a spatial-temporal resource. The proposal in [42] considers opportunistic resource information dissemination in transportation application scenarios. An MP transmits its resources to the MPs that it encounters, and obtains resources from them in exchange.

Our work differs from that of [43, 42] in the following ways. First, the works in [43, 42] primarily address data dissemination, while we consider a query-based model (i.e., on-demand services). The push-based data dissemination model has limited applications since devices need to preserve their limited battery power. Hence, M-P2P environments require pull-based querying models so that peers get exactly what they are looking for. Second, the works in [43, 42] do not consider incentives for free-riders to host data, while we provide such incentives (e.g., royalty model). In querying, incentives make much more sense since an MP has to find the right data, thus, it is a more value-added service. Therefore, we need incentive schemes to entice free-riders to host data.

Third, in [43, 42], constraint queries are not addressed since their model is data dissemination as opposed to querying. In contrast, we consider the efficient processing of constraint queries by means of the CR*-tree constraint index. Fourth, in [43, 42], a royalty model is not considered, while we propose a group-based royalty model. Observe that collaboration is very explicit in a querying model, where peers may work in groups, and therefore royalty-sharing models are required. Additionally, in order to have better control, we select a broker-based model for better managment of the network operations. Furthermore, economic schemes for resource allocation in wireless ad hoc networks [26, 44, 45] also do not entice free-riders to host data.

**Schemes for static P2P networks:** Schemes for combating free-riding in static P2P networks involve formal game-theoretic models for incentive-based P2P file-sharing systems [15], utility functions to capture peer contributions [18], EigenTrust scores to capture participation criteria [22] and asymmetric incentives based on disparities between upload and download bandwidths [25]. The work in [2] proposes a barter-based economic model for P2P systems. However, the approaches in [15, 18, 22, 25, 2], are too static to be deployed in M-P2P networks since they assume peers' availability and fixed topology. As a single instance, pre-defined data access structures (e.g., distributed hash tables [38]) used in static P2P networks cannot effectively handle mobility of peers and frequent network partitioning, which are characteristic of mobile environments.

**Multi-dimensional indexing:** The R-tree [16] is a popular multidimensional index structure. Leaf nodes in the R-tree contain entries of the form *(oid, rect)* where *oid* is a pointer to the object in the database and *rect* is the Minimum Bounding Rectangle (MBR) of the object. Non-leaf nodes contain entries of the form *(ptr, rect)* where *ptr* is a pointer to a child node in the R-tree and *rect* is the MBR that covers *all* the MBRs in the child node. The R*-tree [3] is a popular and efficient variant of the R-tree.

## 4 Architecture of the ConQuer economic model

This section discusses the architecture and economic model of ConQuer, and details constraint query processing in ConQuer. The architecture of ConQuer consists of query-issuing MPs, relay MPs, broker MPs and data-providing MPs. Relay MPs forward messages (e.g.,

queries, data) in lieu of a *relay commission*. Each broker MP maintains a constraint index called the CR*-tree, and uses this index to direct queries to potential data-providing MPs in lieu of a *broker commission*. (We shall discuss the CR*-tree in Section 5.) The query-issuing MP pays the price of the queried data item to the data-providing MP (which serves the query), and a commission to the broker MP and the relay MPs in the successful query path.

### 4.1 Illustrative example for the network topology in ConQuer

Figure 1 depicts an illustrative example of the M-P2P network topology at a certain point of time. In Figure 1, the query issuing MP $M_I$, the broker MPs $B1$ to $B4$, the data-providing MPs (i.e., $M_S$) $D1$ to $D4$ and the relay MPs $R1$ to $R12$ are indicated by white, yellow, blue and green circles respectively. In this example, suppose a data item $d$ is being requested by $M_I$. Suppose $D1$ to $D4$ all contain some copy of $d$ albeit possibly with varying data quality.

Observe that there can be multiple paths from $M_I$ to the same $M_S$, and these paths may pass through different brokers. As a single instance, $D4$ may be accessed by the paths $\{M_I, R3, B2, R7, R8, R9, D4\}$ and $\{M_I, R5, R6, R7, B4, R12, D4\}$ and $\{M_I, R4, B3, R11, B4, R12, D4\}$. A given path between $M_I$ and a given $M_S$ may have multiple brokers e.g., the path $\{M_I, R4, B3, R11, B4, R12, D4\}$ contains two brokers $B3$ and $B4$. In such cases, the broker that occurs first in the traversal starting from $M_I$ (i.e., $B3$ in this example) acts as the broker MP, while the other brokers (i.e., $B4$) in the path only act as relay MPs. As we shall see shortly in Section 4.4, this is necessary to avoid conflicts among brokers.

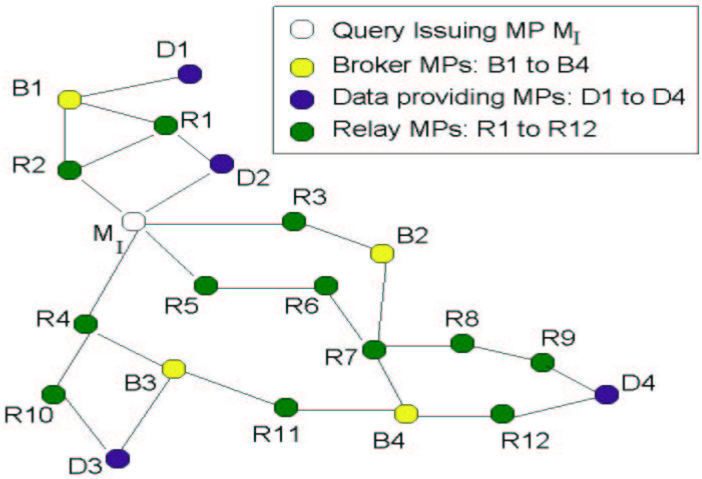



**Fig. 1** Illustrative example of an instance of network topology

The number of relay MPs between $M_I$ and a broker MP can vary e.g., the paths $\{M_I, R2, B1\}$ and $\{M_I, R5, R6, R7, B4\}$ have one and three relay MPs respectively. Furthermore, the number of relay MPs between broker MPs and a given data providing MP $M_S$ can vary e.g., the number of relay MPs in the paths $\{B4, R12, D4\}$ and $\{B2, R7, R8, R9, D4\}$ are 1 and 3 respectively. Thus, the number of hops in the path from $M_I$ to a given $M_S$

can differ. Interestingly, it is also possible for a given $M_S$ to be a one-hop neighbour of $M_I$ e.g., $M_I$ and $D2$ are one-hop neighbours. However, some other $M_S$ such as $D1$ may be able to provide better data quality and/or lower response time than $D2$ (e.g., due to low bandwidth between $D2$ and $M_I$). Hence, the role of the broker MPs would still be relevant in such cases. In essence, the broker MPs provide $M_I$ with different paths for accessing $M_I$'s requested data item $d$ or its copy. This allows $M_I$ to choose the copy of $d$, which best suits $M_I$'s requirements in terms of response time, data quality, trust and price.

### 4.2 Specifying constraint queries in ConQuer

User queries are of the form $\{Q_{id}$, ($k_1$,$k_2$,...,$k_n$), $\tau_{max}$, $DQ$, $Trust$, $\rho_{max}\}$. $Q_{id}$ is the unique identifier of a query, and $k_i$ are user-specified keywords. For example, if an M-P2P user requests for the song 'Imagine' by Beatles, $k_1$= 'Imagine' and $k_2$ = 'Beatles'. $\tau_{max}$ is the query deadline time because of the ephemeral nature of M-P2P environments, due to which queries need to be answered quickly within acceptable deadlines. $DQ$ is the range of user-desired data quality (e.g., image resolution, audio quality) provided by the data-providing MP to the query-issuing MP. We consider three discrete levels of $DQ$ i.e., *high*, *medium* and *low*, their values being 1, 0.5 and 0.25 respectively [30, 31]. Each MP maintains a table $T_{\epsilon,DQ}$, which contains the following entries: (x%, high), (y%, medium), (z%, low), where x, y, z are error-bounds, whose values are essentially application-dependent and pre-specified by the system at design time. Thus, $DQ$ is computed using the table $T_{\epsilon,DQ}$, which is replicated at each MP and is the same for each MP.

$Trust$ is the range of user's desired trust value. ConQuer computes the trust values of data items by adopting the proposal in [33], which proposes a a light-weight decentralized reputation-based trust management mechanism for ad-hoc P2P networks. In [33], the reputation information of each peer is stored in its neighbours and piggy-backed on its messages. As in the computation of $DQ$, each MP maintains a table $T_{\epsilon,Trust}$, which contains the following entries: (x%, high), (y%, medium), (z%, low), where x, y, z are error-bounds, whose values are essentially application-dependent and pre-specified at design time. ConQuer assigns the trust value $Trust$ of an 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, $0 \leq Trust \leq 1$. Thus, a query issuing MP can specify $DQ$ and $Trust$ values as 'high', 'medium' or 'low', and these values are mapped to a value between 0 and 1 e.g., $Trust$ = 'high' maps to the range (0.8,1). Based on queries which a peer relays, he can generally gain some knowledge about these constraints, which helps him to specify them. Finally, $\rho_{max}$ is the *maximum price* that the user is willing to pay for obtaining the query result. Thus, the value of $\rho_{max}$ facilitates in determining the candidate set of data-providing MPs, which can answer a given query.

### 4.3 Economic model of ConQuer

Now we shall present the economic model of ConQuer. In particular, we discuss the computation of data item prices, broker commissions and relay commissions.

**Data item price:** In ConQuer, every data item $d$ has a price $\rho$. When an MP requests $d$, it has to pay the price $\rho$ to the data-providing MP of $d$. Higher data quality $DQ$ of $d$ implies better quality of service for queries on $d$, hence $\rho$ increases as $DQ$ increases. As the bandwidth $BA_{M_S}$ provided by the data-providing MP $M_S$ for queries on $d$ increases, $\rho$ also

increases. This is because faster response times for queries on $d$ should command higher price due to better service quality, given the timeliness requirements of M-P2P applications.

The value of $BA_{M_S}$ depends upon the total bandwidth of $M_S$ and the number of concurrent access requests to $M_S$. Notably, the value of $BA_{M_S}$ is only an estimate, which we use as a guide for computing the data item price. This is because at the time of query issuing, neither the query-issuing MP nor the data-providing MP can know in advance the exact amount $BA_{M_S}$ of bandwidth that the data-providing MP would be able to provide to the query-issuing MP for the download of a given data item. Thus, the exact value of $BA_{M_S}$ will become known to both the query-issuing MP and the data-providing MP only after the data item has been already downloaded.

For determining the value of $BA_{M_S}$, the following two cases arise:

- Actual download bandwidth is greater than or equal to $BA_{M_S}$: The data item price remains the same i.e., even if the data-providing MP provides higher bandwidth for download than $BA_{M_S}$, the data item price would not increase.
- Actual download bandwidth is less than $BA_{M_S}$: The data-providing MP and the query-issuing MP would renegotiate and determine the actual data item price to be paid based on the actual download bandwidth.

In essence, the value of $BA_{M_S}$ reflects the negotiation for download bandwidth, and this negotiation is for the range of bandwidth, and we can always take the average in order to compute the price.

$M_S$ computes the price $\rho$ of a given data item $d$ as follows:

$$\rho \;=\; (\; DQ \times BA_{M_S} \;) \tag{1}$$

Observe that the value of $DQ$ will be known to the query-issuing MP $M_I$ when it obtains the queried data item $d$ from the data-providing MP $M_S$. Furthermore, $M_I$ can estimate the value of $BA_{M_S}$ based on the the download time and the queried data item size. However, this is only an estimate because the queried item may pass through multiple relay MPs before it reaches $M_I$. On the other hand, observe that if $M_S$ lies about its bandwidth to charge higher data item price to $M_I$, it may ultimately not earn any currency from $M_I$ since another data-providing MP of $d$ may be able to provide $d$ at a cheaper price to $M_I$. In essence, our data item price formula can be seen as a *guideline* for MPs to price their data items, but even if some of the MPs deviate from this guideline, the data item price would still be decided by the results of the Vickrey auction (which we shall discuss in Section 4.4) i.e., item prices would essentially depend on demand and supply.

**Broker commission:** For every query answered successfully through itself, a broker obtains a *commission* from the query issuing MP, which provides an incentive for MPs to become brokers. The broker's commission $\beta$ increases with decrease in the query response time $\tau_R$ w.r.t. the query deadline $\tau_D$ to encourage brokers to process constraint queries quickly. Thus, ConQuer provides incentives to well-connected MPs to act as brokers since they can earn higher amount of commission by providing query answers quickly. (We define connectivity of an MP as the number of its one-hop neighbours.) Hence, MPs at the articulation points of biconnected networks [20] are enticed to act as brokers, thereby improving routing services. The computation of $\beta$ follows.

$$\begin{aligned} \beta \;&=\; e^{\tau_D/\tau_R} \;\; if \;\; \tau_R \leq \tau_D \\ &=\; 0 \quad otherwise \end{aligned} \tag{2}$$

Since the values of $\tau_D$ and $\tau_R$ are known to both the query-issuing MP and the broker, it is not possible for the broker to overcharge on commissions, and it is also not feasible for the query-issuing MP to cheat the broker by paying less commission. For trusted accounting in payments between the query-issuing MP and the broker, we use a tamper-resistant security module at each MP [6].

**Relay commission:** The query-issuing MP has to pay a commission $\rho_{Relay}$ to the relay MPs in the successful query path. The value of $\rho_{Relay}$ depends upon the size of the message being relayed. As the message size $size$ (in bytes) increases, $\rho_{Relay}$ also increases due to higher energy and bandwidth consumption of the relay MP. Given that $K$ is the amount of relay commission per byte, $\rho_{Relay}$ is computed below:

$$\rho_{Relay} \quad = \quad K \times size \tag{3}$$

The value of $K$ is essentially application-dependent, and known to all the MPs in the network. Trusted accounting in payments between the query-issuing MP and the relay MPs can be handled by using a tamper-resistant security module at each MP [6]. Since the values of $K$ and $size$ are known to both the query-issuing MP and the relay MPs, it is ensured that the relay MPs cannot fraudulently charge higher relay commissions from the query-issuing MP, and the query-issuing cannot cheat relay MPs by paying lower relay commissions.

### 4.4 Broker-facilitated Constraint Query processing in ConQuer via Vickrey Auctions

Constraint query processing in ConQuer uses a *reverse Vickrey auction* [41] mechanism. Hence, let us understand how the Vickrey auction works for a single item $d$. For simplicity, suppose there is only one seller $S_d$ of $d$, and prospective buyers of $d$ send their *sealed* bids to $S_d$. (Since bids are sealed, buyers are not aware of each others' bids.) $S_d$ sells $d$ to the highest-bidding buyer $B_{High}$. Interestingly, instead of paying $S_d$ its own bid price for $d$, $B_{High}$ pays the price of the second-highest bid (i.e., the highest losing bid) to $S_d$.

A variant of the Vickrey auction is the *reverse* Vickrey auction, whose explanation follows. For simplicity, assume a single item $d$ being auctioned, and there are multiple sellers, but only one buyer $B_d$. Sellers send their bids to $B_d$, and $B_d$ selects the seller $S_{Low}$, whose bid value is lowest. Now, instead of paying $S_{Low}$ the bid price of $S_{Low}$, $B_d$ has to pay $S_{Low}$ the price of the second-lowest bid. The reverse Vickrey auction mechanism is applicable to ConQuer because we can view the data-providing MPs as the sellers, and the query-issuing MP as the single buyer of the queried item under consideration. As noted in [26], the advantage of this auction mechanism is that it provides incentives to sellers to bid according to their perception of the true value of an item without considering the bids of other sellers. Thus, this auction mechanism gives data-providing MPs higher incentives to provide items to the query-issuing MP.

Query processing in ConQuer proceeds as follows. A query-issuing MP $M_I$ broadcasts its constraint query $Q$. Each query has a pre-specified TTL (time-to-live). Non-broker MPs simply forward $Q$. A broker MP $B_i$ receiving $Q$ checks if its index contains the data item (satisfying queried constraints) required by $Q$ and if so, it puts its *broker_id* (unique identifier for a broker) into the query message with a timestamp and becomes a designated broker for $Q$; otherwise, it simply forwards $Q$. (We shall discuss the details of the index at the broker in Section 5.) Notably, once a broker has become a designated broker for $Q$, it uses its index to forward $Q$ to potential data-providing MPs as opposed to broadcasting $Q$.

ConQuer stipulates that in a given query path, only a single MP can act as a broker, thereby resolving conflicts among brokers. Competition among brokers in the same query

path is undesirable because it would be likely to result in duplicate messages, which would increase energy and bandwidth consumption of the relay MPs due to possibly sending and receiving the same query messages more than once. Thus, if a broker $B$ sees that a *broker_id* has already been appended to $Q$, it will not attempt to become a designated broker for $Q$. Even if $B$ fraudulently tries to become the second designated broker in that query path, it cannot earn any currency by facilitating the answer to $Q$ because the query-issuing MP would detect that the timestamp (in the query message) of the designated broker of that path was earlier than $B$'s timestamp. Thus, $B$ would receive no commission from the query-issuing MP, hence it has no incentive to illegitimately try to become the second designated broker in that query path.

Observe that the designated broker for a given query path has no incentive to accept being a broker for a given query if it does not intend to find the answer to the query since it cannot earn any currency by this kind of cheating behaviour. This is because only the broker in the successful query path gets the commission from the query-issuing MP. Furthermore, observe that $Q$ can propagate along multiple paths, hence multiple brokers albeit in different paths would process $Q$, thereby providing better fault-tolerance against the unavailability of some of the brokers. This also guards against the probability of a few brokers accepting queries, but not intending to answer them.

Each broker $B_i$ issues a route-finding query to locate the path to the target data-providing MPs likely to satisfy $Q$. Then $B_i$ sends the query constraints to these target MPs, and collects *bid* information concerning *price*, data quality and trust values for the queried data item(s) from each target data-providing MP, and then returns the *bid* information to the query-issuing MP $M_I$. Notably, the data-providing MPs are allowed to send only one sealed bid per query to a given broker in order to optimize communication overheads e.g., time delays and energy consumption. Since bids are sealed, the target data-providing MPs are not aware of each others' bids. Such sealed bids can be achieved by using traditional encryption schemes such as symmetric keys or public-private key pairs.

From these bids, $M_I$ selects the lowest-priced bid $bid_{low}$ satisfying its constraints and requests the broker $Broker_{Sel}$, which passed it $bid_{low}$, to obtain the queried item from the data-providing MP $M_{bid_{low}}$ corresponding to $bid_{low}$. $Broker_{Sel}$ obtains the data item from $M_{bid_{low}}$ and passes the item to $M_I$. Finally, in accordance with the rules of the reverse Vickrey auction mechanism, $M_I$ pays the price of the second lowest-bid to $M_{bid_{low}}$, the *broker commission* to $Broker_{Sel}$ and the *relay commissions* to the relay MPs in the successful query path. Since $bid_{low}$ is the lowest bid by definition, the second-lowest bid is definitely higher than $bid_{low}$, thereby providing added incentive to MPs to host data items. Notably, as discussed earlier, trusted accounting in payments among the MPs is handled by using a tamper-resistant security module at each MP [6].

## 5 CR*-tree: A dynamic Multidimensional Index for Constraint Queries

This section discusses the CR*-tree constraint index, which is used by the brokers to efficiently direct queries to potential data-providing MPs.

To earn commission, brokers index the constraints of a subset of data items in the network by means of the CR*-tree. The CR*-tree indexes may differ across brokers since they are constructed by brokers based on the queries intercepted by the brokers. Brokers can decide which items to index in several ways e.g., based on interests or commissions gained. For example, a broker may index only movies, while another broker could index a specific genre of songs and so on. Since brokers collect access frequencies and failed query statistics

concerning queries that they intercept, over a period of time, they can estimate their commission from indexing a given item. In essence, in consonance with P2P autonomy, ConQuer allows brokers to autonomously decide the data items, which they want to index.

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. Hence, the constraints need to be indexed. The query response time constraint cannot be indexed since it depends upon network conditions, and delays between the query issuing MP and the data-providing MP. But the constraints of *data quality*, *trust* and *price* can be indexed since they are static as they depend solely upon the data item. Thus, the constraints of each data item can be specified as a point in 3D-space, the dimensions being *data quality*, *trust* and *price*.

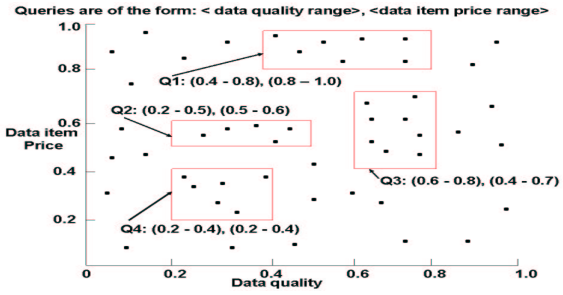



**Fig. 2** Constraints in 2D-space

For clarity, Figure 2 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 2 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 constraint ranges.

### 5.1 The CR*-tree constraint index

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 constraint index stored at each broker. (We use the R*-tree [3] as an example, although it can be substituted by any other multi-dimensional index.) 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. Incidentally, the root node is a special case of the non-leaf nodes for which $LL$ is simply a null list. This facilitates

in avoiding the execution of a sequential scan of all keywords indexed by the CR*-tree for each incoming constraint query.

A leaf node of the CR*-tree is an array of 3D-MBRs. Each MBR contains entries of the form (*mbr*, $LL_I$), where the form of $mbr$ is same as that of the non-leaf nodes. $LL_I$ 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*. $LL_I$ is sorted in dictionary order of the keywords. *freq* is the number of times *keyword* occurs within *mbr*. (*freq* facilitates deletion of data items from the CR*-tree). *arr_MP* is an array of the MPs that store the data item. (A data item could occur at multiple MPs with different values of data quality, trust and price albeit within the same $mbr$.) Thus, brokers process constraint queries by issuing 3D-window queries on the CR*-tree to short-list queried data items satisfying the constraints, and then examining them to determine items satisfying the response time constraint (based on its access statistics for similar queries targetted to the same set of peers). In case a user specifies multiple keywords to specify a given data item, the keywords need to occur *together* in at least one MP within the 3D query window for a query match to occur. For example, suppose the user requests the song 'Imagine' by Beatles. If the keywords 'Imagine' and 'Beatles' occur in different MPs, it does not imply a query match since both these keywords relate to one data item, hence they should occur together in at least one MP for a match to occur.

CR*-tree creation uses the R-tree insertion algorithm [16]. Insertion/deletion of points from the CR*-tree also follow standard R-tree algorithms. The linked lists in the CR*-tree enable efficient insertion/deletion of data items, which is important for ephemeral M-P2P environments. In particular, the CR*-tree uses lazy deletions to conserve MP energy. Brokers maintain access statistics concerning average prices of data items for different values of data quality, trust and response time constraints, hence they have some knowledge concerning reasonable prices, thereby enabling them to reject unrealistic user queries.

Now let us examine the memory space consumption of the CR*-tree. Suppose there are $n$ data items indexed by a given CR*-tree, which has a fanout $f$. Hence, the number $h$ of levels in the CR*-tree would be $\lceil log_f n \rceil$. Thus, the number of nodes in level $i$ of the CR*-tree would be $f^{i-1}$. (The root is considered to be level 1 i.e., $i$ =1 for the root level, and the level just below the root is level 2 and so on.) Thus, the maximum number of nodes in a CR*-tree with $h$ levels (including the root) would ( $1 + f + f^2 + f^3 + \ldots + f^{h-1}$ ), which is essentially the sum of a geometric progession, that reduces to $((1-f^h)/(1-f))$. Since each node can have at most $f$ MBRs, the maximum number of MBRs is $(f(1-f^h)/(1-f))$. Each 3D-MBR can be represented by 12 bytes (restricting to 2 bytes for each of the six constraint values), thus the memory space consumed by the MBRs would be $\mathbf{12 \times (f(1-f^h)/(1-f))}$.

At level $i$ of the CR*-tree, there will be at most $f^i$ MBRs i.e., a maximum of $f^i$ linked lists of keywords. Suppose the linked list corresponding to the $j^{th}$ MBR at level $i$ contains $n_{i,j}$ keywords, hence the number of pointers in this linked list would be $(n_{i,j} - 1)$ *plus* one pointer connecting the linked list to an MBR. Restricting each keyword to be encoded within 8 bytes and representing each pointer by 1 byte, each linked list will consume memory space of $(8 \times n_{i,j}) + (n_{i,j} - 1 + 1) = 9 \times n_{i,j}$. Thus, all the linked lists together consume memory space of $\mathbf{\Sigma_{i=1}^{h} \Sigma_{j=1}^{l_i} (9 \times n_{i,j})}$. Here, $l_i$ is the total number of MBRs in level $i$. For simplicity, we neglect the small differences in memory consumption between linked lists $LL$ of keywords (for non-leaf nodes) and linked lists $LL_I$ (for leaf-nodes) as the predominant storage cost is due to keywords, while the storage costs for keyword frequencies and arr_MP are negligible w.r.t. keyword storage cost.

The number of pointers emanating from the the $i^{th}$ level of the CR*-tree is $f^i$. Hence, the total number of pointers emanating from all the levels is $\Sigma_{i=1}^{h-1} f^i$. Representing each pointer by one byte, the total amount of memory space consumed by the pointers emanating from all levels of the CR*-tree would be $\boldsymbol{\Sigma_{i=1}^{h-1} f^i}$.

Thus, the total memory space consumed by a given CR*-tree indexing $n$ items and having $h$ levels is ( $\boldsymbol{12 \times f(1-f^h)/(1-f) + \Sigma_{i=1}^{h} \Sigma_{j=1}^{l_i} (9 \times n_{i,j}) + \Sigma_{i=1}^{h-1} f^i}$ ). To put things into perspective, suppose $n$ = 10000 and $f$ = 32. Thus, the CR*-tree will have 3 levels, including the root level. (Incidentally, 32768 items can be indexed by a CR*-tree with $f$ = 32.) For simplicity, assuming no overlaps and one keyword per item, the second level and the leaf-level will each have at most a total of 10000 keywords in the corresponding linked lists. (Recall that the root level does not contain a linked list to keywords in order to avoid the potential cost of sequentially scanning through all the items in the index.) In this case, the total number of bytes for storing the CR*-tree (including the linked list of keywords) is less than 600 Kbytes.

Given that the total number of items in M-P2P networks is generally far below 10000 (and a single broker would typically not index all the items), the average size of an MP3 song is usually at least 1 Megabyte and 600 Kbytes is negligible w.r.t. the memory space of most mobile devices in circulation nowadays, we believe that the memory space consumed by the CR*-tree even in the most extreme cases is a small price to pay for the benefit of the value-added routing service provided by the index.

### 5.2 Selling of the CR*-tree index by a broker

ConQuer allows a broker to autonomously sell its CR*-tree index anytime to other MPs in lieu of a payment $\rho_{index}$. This provides an added incentive for brokers to index data and encourages free-riders to host indexes to earn revenue. Furthermore, this also effectively entices the creation of multiple copies of the index, which facilitates more efficient query redirections. This is due to two reasons. First, the broker is enticed to make multiple copies of its index because it is able to earn currency by selling the index. Second, there is an incentive for MPs to buy the index since they can earn currency due to broker commissions on queries successfully directed via the index. In essence, given that there is generally a significant number of queries directed at the index for hot items within a certain time-frame, the index selling mechanism in ConQuer provides incentives to both the broker and the MPs to create multiple copies of the index, thereby leading to improved data availability, query hop-counts and query response times. Trusted accounting in payments between the broker and the MPs can be handled by using a tamper-resistant security module at each MP [6].

The price $\rho_{index}$ of a CR*-tree index hosted at a broker is computed below.

$$\rho_{index} \;=\; \Sigma_{i=1}^{N_{items}} \;\; ( \; ( \; \eta_{succ_i} \; / \; \eta_{total} \; ) \times \tau_i \;\; ) \;\; / \;\; N_{Cp} \qquad (4)$$

where $N_{items}$ denotes the number of items indexed, $\eta_{succ_i}$ is the number of queries on item $i$ successfully answered by using the index and $\eta_{total}$ is the total number of queries (on $i$) that were directed at the index. Thus, the term ( $\eta_{succ_i}/\eta_{total}$ ) quantifies the quality of the index by incorporating the success rate of queries on $i$ due to using the index. In Equation 4, $\tau_i$ is the time-to-expiry of item $i$. $\rho_{index}$ increases with increase in $\tau_i$ because the revenue-earning potential of the index increases when the index comprises higher time-to-expiry items. Finally, $\rho_{index}$ decreases with the number $N_{Cp}$ of existing copies of the index. This is because if more MPs host copies of the index, the future revenue-earning potential of each MP from the index decreases due to the query direction work being divided among the MPs.

The value of $N_{Cp}$ is only an estimate by the broker $B_i$ (which hosts the index) because $B_i$ computes the value of $N_{Cp}$ based on the queries that it intercepts.

Observe that it is possible for $B_i$ to fraudulently increase the price of its index so that it can charge higher selling price for its index since other MPs do not know the values of the variables in Equation 4. However, such a possibility is significantly reduced due to several reasons. First, if $B_i$ charges too high a selling price for its index, fewer MPs would be willing to buy the index from $B_i$, thereby decreasing the total amount of currency that $B_i$ would be able to earn from its index. Second, ConQuer does not restrict the number of copies of a given index because more copies of the index leads to better data availability, lower number of query hop-counts and consequently, faster response times. Moreover, as we discussed earlier, the memory space consumed by the index is negligible w.r.t. the memory space of the mobile devices in circulation nowadays. Thus, $B_i$ has more incentive to keep the index selling price lower so that it can earn more currency by selling its index to more MPs albeit possibly at a lower price.

Third, prospective MPs, which intend to buy the index from $B_i$, would have some idea concerning the actual value of the index due to the queries that pass through themselves, hence they would not be willing to buy an index, whose selling price is estimated by them to be too high. Fourth, since several brokers would try to sell their indexes to earn currency, there would be competition among brokers in terms of index selling. Such competition implies that any given broker would not have much of an incentive to lie about their index selling price since over-priced indexes would be more difficult to sell due to the competition. Our formula for the index selling price is essentially a *guideline* to the brokers. However, even if some of the brokers deviate from this guideline, the index selling mechanism would still be based on the market forces of demand and supply since the prospective index-buyer MPs autonomously decide whether to buy the index.

When a broker $B_i$ wishes to sell its index, it broadcasts the list of items in the index and the price $\rho_{index}$ of the index. MPs, which wish to buy the index from the broker and are willing to pay the price of the index, reply to $B_i$. $B_i$ sends a copy of the index to each of these MPs and receives a payment of $\rho_{index}$ from each of them. MPs decide whether to buy the index based on their estimate of the future revenue-earning potential of the index. This estimate is based on the queries relayed by the MPs and the relative position of the MP in the network w.r.t. the network topology.

ConQuer does not allow indexes to be re-sold i.e., once an MP buys an index from a broker, it cannot re-sell that index. This is because if an index-buyer MP is allowed to re-sell the index, there would be a high possibility of the index selling prices being driven down due to too much supply of indexes w.r.t. demand for the indexes. Such an over-supply of indexes would be highly likely to occur because index-buyer MPs would have a definite incentive to re-sell indexes to earn more currency. However, significant decrease in index selling prices would provide less incentive for brokers to index data. On the other hand, observe that the index selling mechanism in ConQuer from brokers to index-buyer MPs ensures that index prices are not significantly decreased because of the fewer sources of supply (since only brokers can sell indexes), thereby providing the brokers with incentive to index data. The enforcement of this policy of ConQuer concerning not allowing indexes to be re-sold essentially relates to digital rights management and copyright issues, which are beyond the scope of this paper. However, existing digital rights management techniques can be used in conjunction with ConQuer to enforce this policy.

## 6 Peer Group-based Incentive M-P2P Model for Constraint Querying

This section first discusses a greedy method in which all MPs try to maximize their own revenues. Next, to address its drawbacks, we propose a *royalty-based* revenue sharing mechanism among a set of peers. Finally, we devise two *royalty-based* peer-group methods for improving data availability, data quality and MP revenues.

### 6.1 Greedy method: Care-About-Me (CAM)

Under the CAM method, each MP $M$ *greedily* tries to make available (i.e., host) only those data items that will maximize its own revenue. Let the data items stored at $M$ constitute a list $D$. $M$ sorts $D$ in descending order of a parameter $\gamma$, 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 $\gamma$) 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 $D$ at $M$ be $acc_i$ and $\rho_i$ respectively. $\gamma$ is computed based on one of the following approaches:

1. **Energy-constrained approach:** $M$ makes available data items that facilitate it in earning the maximum revenue per unit of energy spent since $M$'s energy resource may be limited. $M$ computes $\gamma$ as $\Sigma_i(acc_i \times \rho_i / E_i)$, where $E_i$ is total energy spent by $M$ for processing $acc_i$ accesses to data item $i$.
2. **Efficiency-constrained approach:** $M$ makes available data items that facilitate it in earning the maximum revenue per unit of time served since $M$ may become unavailable after a short period of time (e.g., due to its schedule or because it may soon relocate outside the group). $M$ computes $\gamma$ as $\Sigma_i(acc_i \times \rho_i / T_i)$, where $T_i$ is the total time spent by $M$ for processing $acc_i$ accesses to item $i$.
3. **Memory-space-constrained approach:** $M$ makes available data items that facilitate it in earning the maximum revenue per unit of its memory space since $M$'s available memory space may be limited. $M$ computes $\gamma$ as $\Sigma_i(acc_i \times \rho_i / size_i)$, where $size_i$ is the size of data item $i$.

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 due to query failures (resulting in lost revenues) related to the unavailable data items. Notably, the approaches in [20], which allocate replicas based on access frequencies of data items, cannot be used to reconcile such redundancy in our incentive-based approach because MPs are autonomous and can therefore exhibit greedy behaviour. Thus, neighbouring MPs would not want to remove duplicate data items among themselves in the absence of incentives.

### 6.2 Royalty-based revenue-sharing approach

To address the deficiencies of the greedy CAM method above, we propose a royalty-based revenue-sharing mechanism among a set of MPs, the aim being to provide incentives to

the MPs to host different data items, thereby improving the overall data availability and data quality. Even though *some* duplicate copies of an item need to be removed, we do not want to completely eliminate all duplicates. This is because we need to maintain acceptable response time so that MPs do not lose revenues due to failure of queries with response time constraints.

In order to effectively coordinate the working of the set of MPs in royalty-based revenue-sharing, a **Master Broker** (**MB**) is needed for the set of MPs. MB is also needed for better network management to facilitate the efficient handling of constrained resources. As we shall see shortly, MB is also required for determining in a coordinated manner how many copies of each item is needed. Recall our application scenario concerning an M-P2P user searching for the cheapest Levis jeans in a shopping district. In this application scenario, the shop-owners in the shopping area will act as brokers. During each time-period, a different shop-owner (broker) will become the MB i.e., the brokers become MB on a round-robin basis. Thus, ultimately, every broker puts in almost equal effort for the coordination of the set of MPs.

Observe that MB is essentially a broker albeit with some additional responsibility. The incentive for a broker to act as the MB is that MB gains additional information concerning the data items in the network from the other brokers. Thus, MB can incorporate this added information into its index, thereby improving the quality of its index. The quality of the MB's index is further improved because when MB is assigning copies of data items to peers under the royalty-based revenue-sharing model, it can also include this information in its index. This enables it to earn more currency in subsequent time-periods due to more amount of broker commissions arising from its better index quality. Furthermore, taking on the responsibilities of being the MB on a round-robin basis ensures smooth and coordinated functioning of the set of MPs, which ultimately results in better revenues as well as better data availability and query response times for the set of MPs as a whole.

Let us now discuss how the MB determines the number of copies for a given item. All brokers in the set send item access frequency information for each query intercepted by them to the designated **MB** of the set. If all queries on a data item $i$ had been successful in satisfying the response time constraint, MB takes no action. Otherwise, MB computes total access frequency $acc_i$ of $i$ by summing up 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.

$$K_i' \;=\; (\; K_i(1 + \lceil (T_{avg} - T_{min})/T_{avg} \rceil)\;) \tag{5}$$

Thus, $(K_i' - K_i)$ additional copies of $i$ need to be made available. Hence, MB estimates the number of copies of data items based on response time constraints posed by queries on them. Notably, MB does not decrease the number of copies of $i$ since such decrease occurs during data reallocation, as we shall see now.

Suppose $n$ MPs in a group are currently making available the data item $i$. In practice, $n$ would be much higher than $K_i'$ since initially, most of the MPs would make available copies of the same hot items to maximize their own revenues as in the CAM method. (If $n$ equals $K_i'$, no action needs to be taken.) If $n > K_i'$, 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 ($acc_{i,j} \times \rho_{i,j}$).

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** 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 items, hence access frequencies for high quality items are much higher than for low quality items. Thus, an MP hosting 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, selection of the top-$K'_i$ revenue-earners for $i$ essentially implies that during data reallocation, excess low quality copies are replaced, while the high quality copies are hosted, thereby implying improvement of 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 hosting the 'hot' 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 also earn some revenue by making available their previously unshared data items to fill up the available memory space arising from replacing $i$. Hence, 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 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. (The reallocation period is the period of time after which MB periodically checks whether any duplicate removal is necessary, and it is application-dependent.) 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:

$$P'_{i,k} \;=\; (C/(t - t_i)) + (1 - C) \times P_{i,k} \tag{6}$$

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 applications, hence we 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 access frequency of $i$ at MP $k$ during the previous period.

Given that $\rho_{i,j,k}$ is the price of the $j^{th}$ access to $i$ at MP $k$, the lost revenue of $k$ (for the next period) due to replacing $i$ for each (future) access is $\rho_{i,j,k}$. Hence, for future $S'_{i,k}$ accesses, the lost revenue of $k$ (for the next period) due to replacing $i$ is ( $\Sigma_{j=1}^{S'_{i,k}} \rho_{i,j,k}$ ). For the next $\tau$ periods, MP $k$'s total lost revenue $LR$ due to replacing $i$ is computed as follows:

$$LR \;=\; \Sigma_{t=1}^{\tau} \;\; ( \;\; \Sigma_{j=1}^{S'_{i,k}} \; (\rho_{i,j,k}) \;\; ) \;\; (1 + \lambda)^{-t} \tag{7}$$

where $\lambda$ 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. $\lambda$ adjusts the royalty payment based on the predicted increase or decrease in access frequency in the next $\tau$ reallocation periods. Preliminary experiments showed that $\tau$ = 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:** 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. 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 $L_f$.

Given that $\rho_{i,j,k}$ is the price of the $j^{th}$ (failed) access to an unshared data item $i$ at MP $k$, the lost revenue of the replace-MP $k$ due to a failed query on $i$ is $\rho_{i,j,k}$. For each (previously unshared) data item $i$ now made available by MP $k$, $k$ computes its future access frequency $F'_{i,k}$ during the next reallocation period based on Equation 6, which can be used in this case because $i$ was missed, which implies that $i$ was accessed. Hence, for $F'_{i,k}$ accesses to $i$, the revenue gained by $k$ would be ( $\Sigma_{j=1}^{F'_{i,k}}\ \rho_{i,j,k}$ ). Observe that now replace-MP $k$ would gain this revenue for accesses to $i$ due to making $i$ available. For the next $\tau$ periods, MP $k$'s total predicted gain in revenue $GR$ due to making available $p$ new data items (i.e., those items which were previously unshared) is computed as follows:

$$GR \ = \ \Sigma_{i=1}^{p} \ ( \ \Sigma_{t=1}^{\tau} \ ( \ \Sigma_{j=1}^{F'_{i,k}} \ (\rho_{i,j,k}) \ ) \ \times \ (1+\lambda)^{-t} \ ) \tag{8}$$

where the significance of $\tau$ and $\lambda$ are same as in Equation 7.

**Computation of the royalty to be paid to replace-MP $k$ by the store-MPs:** Using Equations 7 and 8, 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. $\boldsymbol{RY_i = (LR - GR)}$. 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 $k$.

### 6.3 Royalty-based CAN (Care-About-Neighbours) and CAG (Care-About-Group) methods

Based on the royalty-based revenue sharing method discussed above, we devise two methods, namely CAN (Care-About-Neighbours) and CAG (Care-About-Groups). Both CAN and CAG use the royalty-based revenue sharing method for improving data availability and MP revenues.

Under the CAN method, MPs perform royalty-based revenue sharing only with their one-hop neighbours for improving data availability. For example, suppose MP $A$ is a one-hop neighbour of MP $B$, which in turn, is a one-hop neighbour of MP $C$. In this case, $A$ and $B$ will collaboratively reallocate data using royalties, and $B$ and $C$ will also collaboratively reallocate data. Thus, one-hop neighbours *iteratively* reallocate data collaboratively among each other using the royalty-based method. Observe that CAN has a 'myopic' view in that it only takes its one-hop neighbours into account for collaboratively improving data availability, hence it cannot effectively remove duplicates existing beyond one-hop neighbours.

To remedy the 'myopic' view of CAN, we propose the CAG method, which goes beyond collaboration with one-hop neighbours to collaboration with MPs in a *group*. A group may be formed by MPs with similar interests such as movies and songs. In contrast with CAN, the groups in CAG comprise MPs, which can be beyond one-hop neighbours. Thus, in CAG, MPs use the royalty-based method for revenue sharing with other MPs in their group, which may comprise MPs that are physically further apart in space. Hence, CAG has a broader view of the data availability situation in the network. For forming peer groups, we adopt the proposal in [46], which performs peer clustering in M-P2P networks by modelling the clustered layout based on the *Pareto distribution*. We use the *Bounded Pareto distribution* to bound the minimum and maximum number of nodes in each sub-area to produce a connected network.

For effectively facilitating MPs in the group to make available the required number of copies of hot data items, CAG deploys a super-peer architecture in the group with the Master broker (MB) as the leader of the group to facilitate effective collaboration within the group. (We have discussed the selection of MB in Section 6.2.) 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, while also making available less hot data items, thereby minimizing query failures and improving MP revenues. CAG also ensures better data quality since MPs remove excess low quality copies in favour of higher quality copies.

## 7 Performance Evaluation

This section reports our performance evaluation using simulation. For the simulation, we have used our own implementation.

The parameter values used in our performance evaluation have been selected carefully based on other works such as [20], where a similar environment has been considered for the performance. Additionally, some of the parameter values have been selected according to our environment and our application scenarios. As a single instance, observe that our application scenarios typically concern situations, where the number of MPs is not large. For example, recall our application scenario of car-pooling among car-users, who follow similar paths and timings. In such applications, generally not more than 100 users would be a part of the network. Similarly, for the cab-sharing application scenario, the number of mobile users would typically be less than 100. In a similar vein, for the application scenario involving an M-P2P user looking for like-minded individuals at a conference or social gathering, the number of mobile users would usually be less than 100. Hence, in our experiments, we have set the number of MPs to 100.

The bandwidth range of 28-100 Kbps is in consonance with the available bandwidth ranges, which could be reasonably expected in our application scenarios. Note that our application scenarios involve the transfer of small-sized data items such as text messages and thumbnails of images. Furthermore, we have set the size of a data item in the range of 50-750 Kb because our data items are generally in that range e.g., data about names of songs, data about cheapest available Levis jeans and car-pooling information. In particular, we do not consider big-sized data such as movies. Additionally, we have set relocation period to 200 seconds since it is a reasonable value for our application scenarios. In general, relocation period is tied to replication (analogous to allocation and deallocation of data items in our case), and it is essentially application-dependent, hence nobody knows what is a good relocation period.

We ran each experiment 10 times, hence the experiment result values are the average of 10 runs of each experiment. The respective confidence intervals for our experiments ranged from 92% to 95%, depending upon the experiment. As we present each experiment, we will also point out the confidence interval for that experiment.

MPs move according to the *Random Waypoint Model* [4] within a region of area 1000 metres $\times$1000 metres. The *Random Waypoint Model* is appropriate for our application scenarios, which involve random movement of users. Each MP owns 4 shared data items and 4 unshared data items. (Unshared data items play a role only for CAG and CAN during reallocation.) Among the total of 400 shared data items, the number of unique items is 40. (Also, for the unshared data items, the number of unique items is 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 of *high*, *medium* and *low* by using the Zipf distribution with zipf factor of 0.7 over 3 buckets with $y$ as the input number. The number generated for the first, second and third buckets are for *low*,*medium* and *high quality* respectively. We also generate the data trust mix in the same manner using zipf factor of 0.7 over 3 buckets, corresponding to *high*, *medium* and *low*. Then these constraint values of data quality and trust are assigned randomly to the 10 copies of the same data item. Thus, the proportion of low quality and low trust data items is initially higher, which is in consonance with practice.

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 (zipf factor (ZF) = 0.7). Data item(s) to be queried are selected randomly from the entire set of items in the M-P2P network, the Zipf distribution being used for determining the number of queries corresponding to each of the data items (ZF = 0.7). 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. Response time constraint for each query was determined by a random number generator to be between 40 seconds and 150 seconds. 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. Table 1 summarizes performance study parameters.

| Parameter | Default value | Variations |
|---|---|---|
| No. of MPs ($N_{MP}$) | 100 | 20, 40, 60, 80 |
| Zipf factor (ZF) | 0.7 | |
| Bandwidth between MPs | 28 Kbps to 100 Kbps | |
| Probability of MP availability | 50% to 85% | |
| Size of a data item | 50 Kb to 750 Kb | |
| Memory space of each MP | 2 MB to 5 MB | |
| Speed of an MP | 1 metre/s to 10 metres/s | |
| Size of message headers | 220 bytes | |

**Table 1** Performance Study Parameters

In our simulation model, we have also considered the probability of interruption during data transfer as a parameter during data transfer with the chances of interruption/failure (during data transfer) being 10% to 20%. However, our simulation model does not consider

complete disconnections. In particular, the occurrence of complete interruptions during data transfer is in the realm of networking, which is not our primary focus in this paper. However, we do acknowledge the importance of this issue and leave this issue open to further research. Furthermore, the overhead introduced by the Master Broker is negligible in comparison to the traffic for allocation and reallocation of data items. Notably, the overhead introduced by the Master Broker and the replicas reallocation is part of the communication delay.

Performance metrics are **average response time** (**ART**) of queries, **query success rate** (**SR**) and **query hop-count** (**HC**). ART = $(1/N_Q)\sum_{i=1}^{N_Q}(T_f - T_i)$, where $T_i$ is the time of query issuing, $T_f$ is time of the query result reaching the query-issuing MP, and $N_Q$ is the total number of queries. ART includes the download time, and is computed only for the successful queries. (Unsuccessful queries are not considered for ART computations.) $SR = (N_S/N_Q) \times 100$, where $N_S$ is the number of queries that were answered successfully and $N_Q$ is the total number of queries. In ConQuer, queries can fail due to their constraints not being satisfied or due to MPs being unavailable or due to network partitioning, or due to exceeding the TTL. (Each query has a TTL of 8 hops.) We define the query hop-count **HC** as the hop-count incurred by the query in the successful query path, hence HC is measured only for successful queries.

**Performance comparison w.r.t. existing approaches:** Incidentally, none of the existing proposals for M-P2P networks addresses peer group-based incentive models. Our work cannot be directly compared with incentive schemes for M-P2P networks [43, 42] due to the differences that we have highlighted in Section 3. In particular, the works in [43, 42] consider a data dissemination model, while we consider a querying-based model. The "push-based" data dissemination model in [43, 42] will increase traffic significantly as compared to our "pull-based" querying model, thereby resulting in significantly higher battery power consumption of MPs, and consequently, making it unsuitable for our M-P2P application scenarios, where the battery power of devices is generally limited. Hence, there is no point comparing our work with that of [43, 42].

Our work also cannot be meaningfully compared with incentive schemes for static P2P networks [15, 18, 22, 25, 2] because these incentive schemes do not consider peer mobility and limited battery power resources of the peers, which are characteristic of M-P2P environments. Moreover, schemes for static P2P networks are too static to be deployed in M-P2P environments because they assume peers' availability and fixed topology.

Thus, for purposes of meaningful performance comparison, we adapt the **E-DCG+** approach [20] to our scenario and compare our work against the E-DCG+ approach since it is the closest to our proposed **CAG** and **CAN** methods. E-DCG+ is executed at every reallocation period. Notably, E-DCG+ allocates replicas based on data item access frequencies, and as such, it is not an economy-based incentive scheme. We also compare CAG and CAN with the **CAM** method to indicate the performance gain due to royalty model-based peer collaboration among MPs in a group. For CAM, the three approaches of computing the revenue-earning potential $\gamma$ of a given data item (see Section 6.1) exhibited similar trends, hence here we present the results corresponding to CAM's memory-space-constrained approach. Note that CAG, CAN and CAM use CR*-tree indexes to facilitate efficient direction of queries to target data-providing MPs.

### 7.1 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. For this

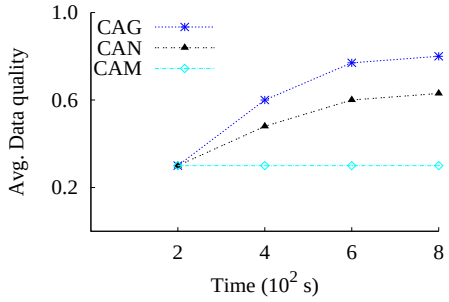
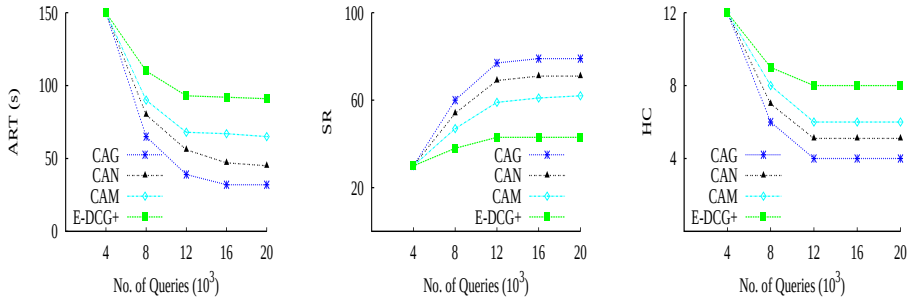



**Fig. 3** Improvement of Data Quality

experiment, we randomly selected a data item whose average data quality was initially low i.e., 0.3. This experiment was run a total of 10 times, each time a data item being randomly selected from the set of items, whose average value of data quality was 0.3. The confidence interval for the results of this experiment is 95%.

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 3. Similarly, for CAN, the average data quality improves due to royalty-based revenue-sharing, but the improvement is less than that of CAG since in CAN, only one-hop neighbours collaborate for data reallocations. Since CAM does not perform such collaborative reallocation of data items, its average data quality remains relatively constant over time. This experiment does not consider E-DCG+, which does not address data quality issues.



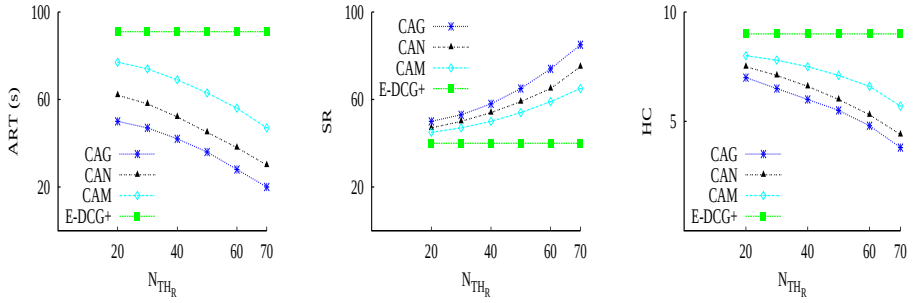

(a) ART (b) SR (c) HC

**Fig. 4** Performance of ConQuer

## 7.2 Performance of ConQuer

Figure 4 depicts the results of our experiment using default parameter values of Table 1. The confidence interval for the results of this experiment is 93%.

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 due to each ap-

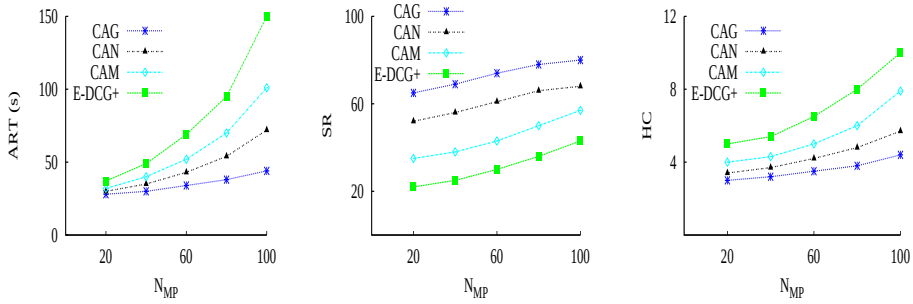



(a) ART (b) SR (c) HC

**Fig. 5** Effect of incentive-based revenue model

proach creating its own optimal number of copies of items in response to the given access pattern. CAG, CAN and CAM outperform E-DCG+ as their incentive-based models encourage MP participation, hence their total available memory space and total bandwidth are higher. The CR*-tree index enables CAG, CAN and CAM to find target MPs efficiently, so they incur lower HC, and hence lower ART than E-DCG+. Furthermore, the index selling mechanism in CAG, CAN and CAM incentivizes MPs to create multiple copies of indexes, thereby further improving HC and ART.

CAG and CAN outperform CAM as royalty-based revenue-sharing ensures better data allocation, which leads to higher SR. In contrast, CAM encourages MPs to make available only hot data items, hence less hot items become unavailable, thereby reducing SR. Unlike CAM, CAG and CAN decide the number of copies for a given item based on response time constraints posed by queries, which further optimizes their ART. CAG outperforms CAN because CAG considers data allocation across peer groups, hence it has better view of the network situation than CAN, which performs allocation only among one-hop neighbours.




(a) ART (b) SR (c) HC

**Fig. 6** Effect of variations in the number of MPs

### 7.3 Effect of incentive-based revenue model

We define threshold revenue $TH_R$ as the ratio of the total revenue of the system to the total number of MPs. The results in Figure 5 indicate that when the number $N_{TH_R}$ of MPs above $TH_R$ increases, ART, SR and HC improve for CAG, CAN and CAM. This is due to more MPs providing service as their revenues increase, which implies more memory space, higher bandwidth and multiple paths for answering constraint queries. E-DCG+ shows constant performance as it is independent of revenue. CAG outperforms both CAN and CAM due to the reasons explained for Figure 4. The confidence interval for the results of this experiment is 92%.

### 7.4 Effect of variations in the number of MPs

Figure 6 depicts the results for varying the number of MPs. The confidence interval for the results of this experiment is 93%.

With increasing number of MPs, HC increases due to larger network size, hence ART increases for all approaches. However, the increase in ART for CAG and CAN is less than that of CAM since CAG and CAN determine the number of copies of any item based on response time constraints posed by queries. SR increases for all the approaches since more MPs imply more opportunities for making available copies of data items. For CAG and CAN, SR eventually plateaus due to network partitioning and unavailability of some MPs. CAG outperforms the other three approaches due to the reasons explained for Figure 4.

## 8 Discussion

This section discusses a few other important issues, which are relevant to our proposal.

Issues concerning fairness are central to the implementation of incentive mechanisms in M-P2P environments. The work in [28] addresses fairness issues concerning scenarios such as how to ensure that intermediate nodes get their commission and how to guarantee that source nodes get their desired service after they have made their payments. Some other works like [27] address issues, where nodes cannot lie about the cost and stability information. In [47], a security scheme has been presented to provide tamper-proof payment in Mobile Ad hoc networks.

In the context of virtual currency payments, there are coupon-based systems such as adPASS [39]. The works in [10, 12, 48] discuss how to ensure secure payments using a virtual currency. Another way proposed in [14] describes Coupons, an incentive scheme that is inspired by the eNcentive framework [32], which allows mobile agents to spread digital advertisements with embedded coupons among mobile users in a P2P manner.

Several non-repudiation [23, 35] systems, which can be incorporated to control the deceiving behaviour of peers, have been developed. In many applications such as content distribution, the price can also be controlled by the service-providers [13]. MoB [7] is an open market collaborative wide-area wireless data services architecture, which can be used by mobile users for opportunistically trading services with each other. MoB also handles incentive management, user reputation management and accounting services. A bootstrap kind of mechanism can also be used in many applications [11]. Symella is a Gnutella file-sharing client for Symbian smartphones. It expects that illegal acts occur, such as interpolation or destruction of the distribution history to get incentives. Therefore, the distribution history

attached to the e-coupon [8] is enciphered with a public-key cryptographic system so that users cannot peruse the distribution history. Furthermore, a message digest (MD) of the distribution history is embedded by digital-watermarking technology to check the validity of the history. Tribler [21] is a first attempt towards turning bandwidth into a global currency. Notably, the schemes discussed above are complementary to our proposal, but they can be used in conjunction with our proposal.

In this paper, we only consider cost parameters which we discussed. Some other costs that could be included in the future are the cost of maintaining virtual currency, encryption cost and storage cost. Note that the bandwidth cost is same for all the participating peers as they put aside some fixed bandwidth for collaboration and for their own benefit, and therefore we do not consider that cost explicitly.

## 9 Conclusion

In M-P2P networks, peers may issue queries with varying constraints on query response time, data quality of results and trustworthiness of the data source. For facilitating the efficient processing of such constraint queries in M-P2P networks, we have proposed **ConQuer**, which is an economic incentive model for M-P2P networks. ConQuer also provides incentives for peer collaboration in order to improve data availability. The main contributions of ConQuer can be summarized as follows. First, it uses a *broker-based economic incentive* M-P2P model for processing constraint queries via a Vickrey auction mechanism. The incentive model effectively combats free-riding and encourages peer participation. Second, it proposes the CR*-tree, a dynamic multidimensional R-tree-based index for constraints of data quality, trust and price of data to determine target peers efficiently. The CR*-tree is hosted by brokers, who can *sell* it to other peers, thereby encouraging the creation of multiple copies of the index for facilitating routing. Third, it provides incentives for peers to form collaborative peer groups for maximizing data availability and revenues by mutually allocating and deallocating data items using *royalty-based* revenue-sharing. Such reallocations facilitate better data quality, thereby further increasing peer revenues. Our performance study demonstrates that ConQuer is indeed effective in answering constraint queries with improved response time, success rate and data quality, and querying hop-counts.

## References


1. E. Adar and B. A. Huberman. Free riding on Gnutella. *Proc. First Monday*, 5(10), 2000.
2. K.G. Anagnostakis and M.B. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. *Proc. ICDCS*, 2004.
3. 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.
4. J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocol. *Proc. MOBICOM*, 1998.
5. L. Buttyan and J. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. *Technical Report DSC/2001/001, Swiss Federal Institute of Technology,Lausanne*, 2001.
6. L. Buttyan and J.P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), 2003.
7. R. Chakravorty, S. Agarwal, S. Banerjee, and I. Pratt. MoB: a mobile bazaar for wide-area wireless services. *Proc. MobiCom*, 2005.
8. K. Chen and K. Nahrstedt. iPass: an incentive compatible auction scheme to enable packet forwarding service in MANET. *Proc. ICDCS*, 2004.

9. J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modelling incentives for collaboration in mobile ad hoc networks. *Proc. WiOpt*, 2003.
10. P. Daras, D. Palaka, V. Giagourta, and D. Bechtsis. A novel peer-to-peer payment protocol. *Proc. IEEE EUROCON*, 1, 2003.
11. A. Datta, M. Hauswirth, and K. Aberer. Beyond web of trust: Enabling P2P e-commerce. *Proc. ICEC*, 2003.
12. E. Elrufaie and D. Turner. Bidding in P2P content distribution networks using the lightweight currency paradigm. *Proc. ITCC*, 2004.
13. D.R. Figueiredo, J. Shapiro, and D. Towsley. Payment-based incentives for anonymous peer-to-peer systems. *UMass CMPSCI Technical Report 04-62*, 2004.
14. A. Garyfalos and K.C. Almeroth. Coupon based incentive systems and the implications of equilibrium theory. *Proc. IEEE International Conference on E-Commerce Technology proceedings*, 2004.
15. P. Golle, K.L. Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. *Proc. Electronic Commerce*, 2001.
16. A. Guttman. R-trees: A dynamic index structure for spatial searching. *Proc. ACM SIGMOD*, pages 47–57, 1984.
17. R. Guy, P. Reiher, D. Ratner, M. Gunter, W. Ma, and G. Popek. Rumor: Mobile data access through optimistic peer-to-peer replication. *Proc. ER Workshops*, 1998.
18. M. Ham and G. Agha. ARA: A robust audit to prevent free-riding in P2P networks. *Proc. P2P*, pages 125–132, 2005.
19. T. Hara and S.K. Madria. Consistency management among replicas in peer-to-peer mobile ad hoc networks. *Proc. IEEE SRDS*, 2005.
20. T. Hara and S.K. Madria. Data replication for improving data accessibility in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11), 2006.
21. http://tv.seas.harvard.edu/research.php.
22. S. Kamvar, M. Schlosser, and H. Garcia-Molina. Incentives for combatting free-riding on P2P networks. *Proc. Euro-Par*, 2003.
23. S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of non-repudiation protocols. *Technical Report 473, ULB*, 2002.
24. S. Lee, R. M. Muhammad, and C. Kim. A leader election algorithm within candidates on ad hoc mobile networks. *LNCS Embedded Software and Systems*, 2007.
25. N. Liebau, V. Darlagiannis, O. Heckmann, and R. Steinmetz. Asymmetric incentives in peer-to-peer systems. *Proc. AMCIS*, 2005.
26. Jinshan Liu and Valerie Issarny. Service allocation in selfish mobile ad hoc networks using vickrey auction. *Proc. Current Trends in Database Technology - EDBT Workshops, LNCS 3268*, 2004.
27. M. Lu, F. Li, and Jie Wu. Incentive compatible cost and stability-based routing in ad hoc networks. *Proc. ICPADS*, 2006.
28. R. Lu and X. Lin et al. A novel fair incentive protocol for mobile ad hoc networks. *Proc. WCNC*, 2008.
29. R. Mannak, H. de Ridder, and D.V. Keyson. The human side of sharing in peer-to-peer networks. *Proc. European Union symposium on Ambient intelligence*, 2004.
30. A. Mondal, S.K. Madria, and M. Kitsuregawa. CADRE: A collaborative replica allocation and deallocation approach for Mobile-P2P networks. *Proc. IDEAS*, 2006.
31. A. Mondal, S.K. Madria, and M. Kitsuregawa. CLEAR: An efficient context and location-based dynamic replication scheme for Mobile-P2P networks. *Proc. DEXA*, pages 399–408, 2006.
32. O. Ratsimor, T. Finin, A. Joshi, and Y. Yesha. eNcentive: A framework for intelligent marketing in mobile Peer-to-Peer environments. *Proc. ICEC*, 2003.
33. T. Repantis and V. Kalogeraki. Decentralized trust management for ad-hoc peer-to-peer networks. *Proc. MPAC*, 2006.
34. B. Richard, D. Nioclais, and D. Chalon. Clique: A transparent, peer-to-peer replicated file system. *Proc. MDM*, 2003.
35. J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1), 2005.
36. S. Saroiu, P.K. Gummadi, and S.D. Gribbler. A measurement study of P2P file sharing systems. *Proc. MMCN*, 2002.
37. V. Srinivasan, P. Nuggehalli, C.F. Chiasserini, and R. R. Rao. Cooperation in wireless ad hoc networks. *Proc. INFOCOM*, 2003.
38. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *Proc. ACM SIGCOMM*, 2001.
39. T. Straub and A. Heinemann. An anonymous bonus point system for mobile commerce based on word-of-mouth recommendation. *Proc. ACM SAC*, 2004.

40. D.A. Turner and K.W. Ross. A lightweight currency paradigm for the P2P resource market. *Proc. Electronic Commerce Research*, 2004.
41. W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 41:8–37, 1961.
42. O. Wolfson, B. Xu, and A.P. Sistla. An economic model for resource exchange in mobile Peer-to-Peer networks. *Proc. SSDBM*, 2004.
43. B. Xu, O. Wolfson, and N. Rishe. Benefit and pricing of spatio-temporal information in Mobile Peer-to-Peer networks. *Proc. HICSS-39*, 2006.
44. Yuan Xue, Baochun Li, and Klara Nahrstedt. Channel-relay price pair: Towards arbitrating incentives in wireless ad hoc networks. *Journal of Wireless Communications and Mobile Computing, Special Issue on Ad Hoc Networks, Wiley InterScience*, 2005.
45. Yuan Xue, Baochun Li, and Klara Nahrstedt. Optimal resource allocation in wireless ad hoc networks: A price-based approach. *IEEE Transactions on Mobile Computing*, 2005.
46. C. Yu, K.G. Shin, B. Lee, S.M. Park, and H.N Kim. Node clustering in mobile peer-to-peer multihop networks. *Proc. PERCOMW*, 2006.
47. Y. Zhang, W. Lou, W. Liu, and Y. Fang. A secure incentive protocol for mobile ad hoc networks. *ACM Wireless Networks*, 13(5):663–678, 2006.
48. S. Zhong, J. Chen, and Y.R. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. *Proc. IEEE INFOCOM*, 2003.
49. S. Zhong, L. (Erran) Li, Y. G. Liu, and Y. (Richard) Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: an integrated approach using game theoretical and cryptographic techniques. *Proc. Proc. International conference on Mobile computing and networking*, 2005.