Recall-Based Cluster Reformulation by Selfish Peers

Georgia Koloniari and Evaggelia Pitoura
Computer Science Department, University of Ioannina, Greece
{kgeorgia, pitoura}@cs.uoi.gr

ABSTRACT

Recently, clustered overlays in which peers are grouped based on the similarity of their content or interests have been proposed to improve performance in peer-to-peer systems. Since such systems are highly dynamic, the overlay network needs to be updated frequently to cope with changes. In this paper, we introduce an approach for updating a clustered overlay based on local decisions made by individual peers. We model the cluster-reformulation problem as a game where peers determine their cluster membership based on potential gains in the recall of their queries. We also define global criteria for the overall quality of the system and propose strategies for peer relocation that consider different behavioral patterns for the peers. Our preliminary experimental evaluation shows that our strategies cope well with changes in the overlay network.

1. INTRODUCTION

The popularity of file-sharing systems such as Gnutella, Napster and Kazaa has spurred much attention to peer-to-peer (p2p) computing. Nodes (peers) in a p2p system are dynamic and autonomous. They have equal roles acting as both data providers and data consumers. Each peer connects to a small subset of other peers, thus logical overlay networks are formed on top of the physical one (usually the Internet). Queries are routed over the overlay network to locate peers that maintain relevant data. The basic challenge lies in locating these peers and retrieving as many results as possible, i.e. increasing the query recall, with a small communication cost.

To address the above issue, clustering has been proposed [1, 2, 6, 8, 4]. Clustering has been widely used in databases to reduce the I/O cost by placing data items with similar properties together in the hard disk, since it is expected that similar data are accessed together. Similarly, by clustering together peers that share the same content or have the same interests (similar query workload), queries can locate their results more efficiently. Once the appropriate cluster is located, all other results are found nearby.

However, previous work mostly focused on the formation of clusters and ignored the maintenance of the clustered overlay, which is needed for coping with the dynamic nature of the peers. Peers that join or leave the system constantly and change their content and query workload frequently may render the original clustered overlay inappropriate under the current system conditions. One solution to the problem is to re-apply the clustering procedure that was used to form the original overlay from scratch taking into account the updated state. However, this incurs large communication costs. It also requires global knowledge about the system state that compromises peer autonomy.

In this paper, we present an approach for handling the maintenance issues based on decisions that each peer makes based on local criteria. We model the problem of cluster reformulation as a game in which the peers are the players that decide whether they should move to another cluster so as to minimize an individual cost function. This cost function depends on the membership cost entailed in belonging to a cluster and the cost of evaluating the peer query workload at remote clusters. Furthermore, we introduce global system quality criteria to measure the performance of the system as a whole. Similar approaches based on game-theoretic models have been proposed for creating overlays based on the connection cost and radius of the network graph [3, 5, 7]. The novelty of our approach is that it relies on the queries and aims at increasing their recall.

In addition, we propose strategies that the peers follow to decide whether they need to relocate to another cluster based on the above criteria, that model either selfish or altruistic behavior. Finally, we perform a number of experiments that show how our strategies help in coping with the changes in the overlay configuration without compromising its quality. Furthermore, our experiments show that given (the usual assumption for clustering) that the underlying data share some similar properties, our strategies converge to well formed clusters for most initial system configurations.

The rest of the paper is organized as follows. In Section 2, we present the cluster reformulation problem as a game and define the cost function and the global quality criteria. We also discuss the convergence of the system. In Section 3, we introduce the strategies for peer relocation and describe how they can be applied. Section 4 presents our experimental evaluation and Section 5 refers to related research. We conclude in Section 6.

2. RECALL-BASED CLUSTERING

We consider a peer-to-peer system with highly dynamic autonomous nodes that share content. We use P to denote the current set of peers. We do not assume any specific model for the data items shared by the peers, but adopt a rather generic approach where each data item is described by a set of attributes (e.g., keywords for text documents). Queries are sets of attributes. We say that a query q *matches* a data item d of peer p, if its attributes are a subset of the attributes describing d and denote the number of such matching items as *result(q, p)*.

Let Q be the list of all queries in the system. Note that a query q may appear more than once in Q. Let $num(Q)$ be the number of all queries in Q and $num(q, Q)$ be the number of appearances of query q in Q. We characterize the importance of a peer p in the evaluation of a query q in Q based on the results that p offers for q with regards to the total number of available results (i.e., the recall achieved when q is evaluated solely on p). Specifically:

$$r(q,p) = \frac{result(q,p)}{\sum_{p_k \in P} result(q,p_k)}.$$

We also define as *local workload* of peer p_i, $Q(p_i)$, the list of queries that were issued by peer p_i. Again $num(Q(p_i))$ stands for the number of all queries in $Q(p_i)$ and $num(q, Q(p_i))$ for the number of appearances of query q in $Q(p_i)$.

2.1 Individual Peer Cost

To increase the recall of their queries, peers form sets, called *clusters*. The main motivation for clustering is that inside each cluster, the evaluation of a query is cost efficient. We model the problem of cluster formulation as a game. Each peer represents a player in the game and its strategy is defined by the set of clusters it joins. In particular, each peer p chooses which clusters to join from the set of C_{max} clusters in the system, $C = \{c_1, c_2, \ldots, c_{C_{max}}\}$, thus, defining its strategy $s_i \subseteq C$. In this paper, we let C_{max} be equal to $|P|$, i.e., it cannot exceed the number of peers in the system, and assume that some clusters may be empty if needed. The goal of the game is for each player (peer) to minimize a cost function. The cost function for each peer p is defined based on the recall of its local query workload. Specifically, each peer chooses to join those clusters whose peers would increase the recall of its local workload the most.

In other words, let $P(s_i)$ be the set of peers belonging to any cluster $c \in s_i$. The gain for a peer p for choosing a strategy s_i is the recall of its local workload achieved by evaluating its queries in the peers $P(s_i)$. Stated differently, the cost associated with a strategy s_i for a peer p is the cost (recall) for obtaining query results from peers located in clusters that do not belong to s_i, that is, for peers not in $P(s_i)$.

Clearly, this cost is minimized, if a peer joins all C_{max} clusters in the system. However, participation in a cluster imposes communication and processing costs. Such costs depend on the size of the cluster. The larger the size of the cluster, the higher the cost of joining, leaving and maintaining the cluster as well as for processing queries within it. We define a monotonically increasing function θ of the number of peers belonging to a cluster, i.e., the cluster size $|c_i|$ to capture this cost. This function depends on the cluster topology, for instance, when all peers are connected to each other, θ is linear, whereas in the case of structured overlays, θ may be logarithmic.

Thus the individual cost for a peer p for choosing strategy s_i is defined as follows:

$$pcost(p, s_i) = \alpha \sum_{c_k \in s_i} \frac{\theta(|c_k|)}{|P|} + \sum_{q in Q(p_i)} \frac{num(q, Q(p_i))}{num(Q(p_i))} \sum_{p_j \notin P(s_i)} r(q, p_j) \quad (1)$$

The first term expresses the cost for cluster membership and the second one the cost (in terms of recall) for obtaining results from peers outside the selected clusters, that is, the average result loss from not participating in all clusters. Parameter α ($\alpha \geq 0$) determines the extent of influence of the cluster participation cost in cluster formation. Finally, factor $1/|P|$ is used for normalizing the cluster membership cost.

2.2 Global Cost

We can describe any cluster configuration by the set of strategies $S = \{s_1, s_2, \ldots, s_{|P|}\}$ the peers in P have deployed, since from this set we can derive the set of peers belonging to each cluster in C. One way to measure the overall quality of a cluster configuration is by evaluating the achieved *social cost* ($SCost$), i.e. the sum of the individual costs of all peers in P.

$$SCost(S) = \sum_{p_i \in P} pcost(p_i, s_i), \quad (2)$$

where s_i denotes the strategy of peer p_i.

We can also evaluate the overall quality of the configuration from the query workload perspective, by considering the average cost for attaining results for all queries in Q. Then, the *workload cost* ($WCost$) is defined as:

$$WCost(S) = \alpha \sum_{c_k \in C} \frac{|c_k|\theta(|c_k|)}{|P|} + \sum_{q_m in Q} \frac{num(q_m, Q)}{num(Q)} \sum_{p_i, q_m in Q(p_i)} \frac{num(q_m, Q(p_i))}{num(q_m, Q)} \sum_{p_j \notin P(s_i)} r(q_m, p_j) \quad (3)$$

The first term expresses the cost of maintaining the clusters. The second term expresses the cost of all queries, i.e., the recall of evaluating them outside the clusters of their initiator. The main difference between the social and the workload cost is how they assign weights to the queries. In the social cost, each peer assigns weights to its queries based on their frequency in its local workload, whereas in the workload cost, the weight assigned to each query is based on the frequency of the query in the overall query workload. Intuitively, while the social cost regards all peers as equals, the workload cost considers more demanding peers, i.e. peers that pose more queries, as more important than low demanding ones.

The two cost measures are not equal in the general case. Let us see how the two costs relate to each other. Using the definition of individual cost, we get for the social cost:

$$SCost(S) = \alpha \sum_{p_i \in P} \sum_{c_k \in s_i} \frac{\theta(|c_k|)}{|P|} + \sum_{p_i \in P} \sum_{q in Q(p_i)} \frac{num(q, Q(p_i))}{num(Q(p_i))} \sum_{p_j \notin P(s_i)} r(q, p_j) \quad (4)$$

The first term of $SCost$ is equal to the first term of $WCost$. Just consider that each cluster c_k appears in the sum of $SCost$ as many times as the peers that belong to it, i.e., its size $|c_k|$. By simple manipulations, we get for the second term of WCost:

$$\sum_{q_m in Q} \sum_{p_i, q_m in Q(p_i)} \frac{num(q_m, Q(p_i))}{num(Q)} \sum_{p_j \notin P(s_i)} r(q_m, p_j)$$

Thus, the two terms differ only on how much the workload of each peer is taken into account in the cost. It is easy to see, that if peers get an equal part of the query workload, i.e., $num(Q(p_i)) = num(Q(p_j))$, for all peers p_i, $p_j \in P$, the recall parts of the two costs are proportional, that is, improving the social cost improves the workload cost and vice versa.

PROPERTY 1. *If for each peer $p_i \in P$, $num(Q(p_i)) = \frac{num(Q(p))}{|P|}$, the social and the workload cost measures are proportional to each other.*

2.3 System Convergence

The goal of each player (peer) is to minimize its individual cost. The question that arises is: if we leave the players free to play the game to achieve their goal, will the system ever reach a stable state in which no players desire to change their strategy (the set of clusters they belong to)? Formally, a (pure) Nash equilibrium is a set of strategies S such that, for each peer p_i with strategy $s_i \in S$, and for all alternative set of strategies S' which differ only in the i-th component (different cluster sets s'_i for peer p_i), $pcost(p_i, s_i) \leq pcost(p_i, s'_i)$. This means that in a Nash equilibrium, no peer has an incentive to change the set of clusters it currently belongs to, that is, Nash equilibria are stable.

For simplicity, in the rest of this paper, we focus on the case where each peer belongs to only one cluster ($s_i = c_j$). So instead of $pcost(p_i, s_i)$, we shall write $pcost(p_i, c_j)$.

It is rather simple to prove that a Nash equilibrium does not always exist. Let us consider a simple scenario of two peers p_1 and p_2. Consider now that $Q(p_1)$ consists of a single query q_1 satisfied by p_2 and $Q(p_2)$ consists of q_2 also satisfied by p_2. There are three possible cluster configurations for the system, $p_1 \in c_1$ and $p_2 \in c_2$, $p_1 \in c_2$ and $p_2 \in c_1$ and $p_1, p_2 \in c_1$ or c_2 equivalently. Let us assume a linear θ function. Then, for any value of $\alpha > 0$, we can show that none of the possible configurations is a Nash equilibrium. In particular, since the first two configurations are symmetric let us examine the first one. The individual costs of the two peers are: $pcost(p_1, c_1) = \alpha\frac{1}{2} + 1$ and $pcost(p_2, c_2) = \alpha\frac{1}{2}$. If p_1 moves to cluster c_2 then its cost becomes $pcost(p_1, c_2) = \alpha \leq pcost(p_1, c_1)$. Thus, this configuration is not a Nash equilibrium since p_1 can reduce its cost by moving to c_2. Let us consider now the configuration in which both peers belong to the same cluster. Their individual costs are now: $pcost(p_1, c_1) = \alpha$ and $pcost(p_2, c_1) = \alpha$. Peer p_2 can reduce its cost by moving to the empty cluster c_2 and therefore this configuration is not a Nash equilibrium.

3. CLUSTER REFORMULATION

Assume some initial cluster configuration. As the system evolves, the recall achieved by this cluster configuration may deteriorate. Changes that affect the quality of clustering include topology updates as peers enter and leave the system, as well as changes of the peer content and the query workload. We propose a suite of protocols to keep the clustered overlay up-to-date with respect to these changes. Our protocols are based on local relocation strategies that each peer follows so that it moves to the most appropriate cluster under the given system conditions. We describe first, the relocation strategies followed by each peer and then how such strategies are coordinated towards creating a new cluster configuration.

3.1 Relocation Strategies

Due to updates, a peer may no longer benefit from the clusters it belongs to. In such a case, the peer may decide to move to a new cluster. We consider periodic strategies, where each peer re-evaluates its cluster memberships at periods of T. We assume that each cluster has a unique identifier, *cid*, and that all peers in the cluster are aware of this unique id. We also assume that the results of each query are annotated with the corresponding *cid*s of the clusters that provided them.

Since a peer can not be aware of all available results for a query in the system, we define a measure called *cluster recall* as the fraction of results returned to peer p for query q by a cluster c_i to the total number of results returned for the query. The number of these results depends on the routing algorithm used, and if a query is evaluated against all clusters in the system, it is equal to the total number of results for q in the system.

Based on the behavior of each peer, we consider two types of relocation strategies: *selfish* and *altruistic*.

3.1.1 Selfish Relocation Strategy

This strategy is governed by peer selfishness. A peer determines that its current cluster is no longer suitable, when it observes that its queries receive a low recall from it.

Since, all query results received by a peer are annotated with the *cid* of the cluster they came from, each peer can keep track of its recall with respect to all clusters in the system. In particular, for the time period T, each peer p incrementally updates the $pcost(p, c_i)$ for each of the clusters c_i in the system. At the end of T, each peer selects the c_i for which:

$$pcost(p, c_i) = min_{c_j \in C}(pcost(p, c_j)) \quad (5)$$

The motivation behind this strategy is that a peer chooses to move to the cluster that has yielded the most results for its query workload. After choosing this cluster, say cluster c_{new}, the peer computes a measure called *individual peer gain* (*pgain*) as follows: $pgain(p, c_{new}) = pcost(p, c_{cur}) - pcost(p, c_{new})$, where c_{cur} denotes the cluster the peer currently belongs to.

3.1.2 Altruistic Relocation Strategy

In contrast to selfish relocation, altruistic relocation assumes that all peers act with altruism. In particular, in this strategy, the peers decide to move to the cluster whose recall could improve the most by this movement.

To this end, each peer keeps track of the number of results it sends to queries coming from a particular cluster. In particular, it incrementally updates over the time frame T the following measure for each c_i:

$$contribution(p, c_i) = \frac{\sum_{p_i \in c_i} \sum_{q_m in Q(p_i)} result(q_m, p)}{\sum_{p_j \in P} \sum_{q_m in Q(p_j)} result(q_m, p)} \quad (6)$$

At the end of T, it selects the cluster c_{new} with the maximum contribution value and evaluates a new measure for that cluster (*cluster gain (clgain)*) that is the increase in the membership cost of c_{new} p will cause if it joins it, minus p's contribution to it.

3.2 Cluster Reformulation Protocol

The cluster update protocol takes place periodically. The protocol proceeds in rounds where each round has two phases:

in the first phase, relocation requests are gathered, and in the second phase, they are served. To avoid broadcasting messages among all peers, we assume that one peer per cluster acts as the cluster representative. The representatives of each cluster do not need to be the same in all rounds of the protocol. Note also that, it is not required that the cluster representatives of all clusters are known to each other. If the cluster representative forwards the request to a peer in the cluster, the peer can then propagate it to its representative.

In the first phase of each round, each peer evaluates its gain factor (depending on the relocation strategy applied) and sends this value to its cluster representative. Each cluster representative selects the peer with the highest gain value in its cluster and sends a relocation request to all other cluster representatives including its own *cid*, the *cid* of the cluster the peer wants to move to and the value of the gain. In the case where no peer needs to relocate, the representative sends just a message with its *cid*. When all representatives have received relocation requests from all other clusters in the system, the second phase of the protocol begins.

In the second phase of each round, each cluster representative sorts the relocation requests that it has received according to their gain value. To speed-up this phase, we try to avoid cycles due to groups of peers moving in loops among the same set of clusters. To achieve this, we enforce the following rule: if peer $p \in c_i$ moves to c_j, then c_i is locked with direction *leave* and c_j with direction *join*. In the same round, no more peers can join c_i or leave c_j. To enforce this condition, after the representatives have sorted the requests in decreasing order of gain value, the first relocation request in the list is granted. The two cluster representatives that are involved in the request communicate with each other to satisfy the request. Each cluster representative locks the two clusters, i.e., it removes all other requests in the list that involve either of them with a direction that violates the rule. The process continues by serving the next request remaining in the ordered list. Note that cluster representatives can process their lists independently. After, all cluster representatives have processed their lists, the protocol proceeds to the next round. The protocol ends, when the peer representatives receive no further relocation requests.

Since each round imposes considerable overheads, we enforce a stop condition for the protocol. In particular, before issuing a relocation request, each peer compares its gain with a system-defined threshold ϵ and issues the request, only if the gain is larger than this threshold. Such decisions may also be taken at the system level. Since cluster representatives exchange information about the costs of all peers in their cluster, they can evaluate the social cost of the system. In this case, if the difference from the original value is satisfactory, the cluster representatives can stop the maintenance procedure.

Finally, besides peers changing clusters, there are two special conditions, namely deleting a cluster and creating a new one. It is possible, for all members of a cluster to leave it, after the application of the maintenance procedure. In this case, the cluster is considered as an empty cluster. Regarding the creation of a new cluster, consider as an example, a scenario where the query workload in the system changes and some peers start posing queries for a data item for which they have no previous interest. It is possible, that the creation of a new cluster would improve recall. If the peer observes that no move to any existing cluster can improve its individual cost and also that its cost has significantly been increased since the last time period, then the peer decides to leave its cluster and move to one of the empty clusters in the system, automatically becoming the representative of this cluster. It is possible that more than one peer make this decision in the same round. However, in the following rounds, all such peers will tend to move to just one of these newly-formed clusters.

Table 1: Results for fixed query workload and content

	# Rounds		# Clusters		SCost		WCost	
	Self	Alt	Self	Alt	Self	Alt	Self	Alt
Data and Queries in Same Category								
i	9	10	10	10	0.1	0.1	0.1	0.1
ii	21	19	10	10	0.1	0.1	0.1	0.1
iii	18	20	10	10	0.1	0.1	0.1	0.1
iv	15	17	10	10	0.1	0.1	0.1	0.1
Data and Queries in Different Categories								
i	84	81	90	90	0.31	0.33	0.29	0.31
ii	67	65	90	90	0.33	0.32	0.3	0.27
iii	132	129	90	90	0.36	0.33	0.3	0.29
iv	84	78	90	90	0.32	0.32	0.28	0.3
Data and Queries Uniformly Distributed								
i	-	-	46	51	0.35	0.42	0.26	0.32
ii	-	-	90	89	0.27	0.37	0.23	0.29
iii	-	-	67	57	0.48	0.31	0.39	0.25
iv	-	-	76	81	0.31	0.29	0.28	0.23

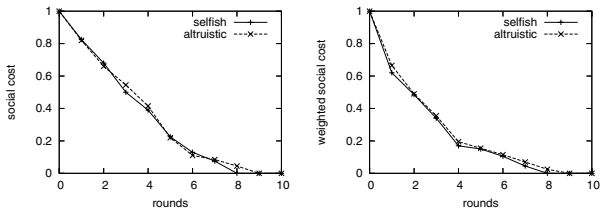

Figure 1: (left) Social Cost and (right) Workload Cost through progressing rounds

4. EVALUATION

We present two sets of experiments. In the first set, we start from a random peer configuration and examine whether the peer relocation protocol leads to the desired cluster configuration for the given data content and workload. In the second set, we start from a "good" cluster configuration for a given content and workload, and examine how well the periodic reformulation protocol adapts to changes of the content and the workload. The peers share Newsgroup articles belonging to 10 different categories. The articles were preprocessed, stop words were removed from the text, lemmatization was applied and the resulting words were sorted by frequency of appearance. The texts are distributed among 200 peers. Queries are generated by choosing a random word from the texts. We set the α parameter equal to 1 and use a linear function (corresponding to a system where all the peers in a cluster are fully connected) for the θ function. Both the social and workload cost measures reported in the experiments are normalized.

4.1 Fixed Query Workload and Content

For the data and query distribution, we consider three different scenarios. For the first scenario, both queries and data of a peer fall into the same category. In the second scenario, each peer has data of a single category and queries about data of a single but different category from its own. In the third scenario, both data and queries for each peer are

selected uniformly at random from all categories. The distribution in the first two scenarios is such that ideally would result to equal sized clusters. The queries are distributed among the peers using a zipf distribution, thus, some peers are more demanding than others. Let M be the optimal number of data categories in the system for each scenario. We consider four different cases for the initial system configuration: (i) each peer forms its own cluster; (ii) peers are randomly distributed to $m = M$ clusters; (iii) peers belong to $m < M$ clusters and (iv) peers belong to $m > M$ clusters.

Peers run the two relocation strategies for multiple rounds. We check whether they have reached an equilibrium and if so, what is the required number of rounds. We also provide the achieved social and workload cost. Table 1 summarizes our results. In the first scenario (Table 1 lines 1-4), all strategies reach a Nash equilibrium rather fast (20 rounds in the worst case). The peers form the desired number of clusters (10). Since all data requests of a peer are contained into its own cluster, the cost for the recall is zero, thus both the social and workload costs are equal to the cluster membership cost. In the second scenario (Table 1 lines 5-8), again, we reach the desired number of clusters, but since this number is larger than the one in the first scenario, it takes more rounds to do so. The recall factor in this case is not zero, since there are data items from the same category a peer requires that are located in different clusters. Also, since the queries are not uniformly distributed among the peers, the social cost differs from the workload cost. Finally, the third scenario (Table 1 lines 9-12) does not reach convergence. In this case, the data and query distributions are such that do not favor the creation of clusters. Both costs exhibit larger values than in the other two scenarios and the number of clusters created varies widely.

A useful conclusion from this experiment is that our proposed strategies can also be applied to cluster discovery. If the distribution of queries and data allows it, our strategies result to well-formed clusters that minimize the social cost (thus, the recall) of the system.

We also measured the progress of the social and workload cost during the different rounds of the relocation protocol. We report the results for the first scenario in Fig. 1. The more demanding peers are granted their relocation requests first. Thus, as Fig. 1(right) shows, the workload cost decreases faster in the first rounds when the demanding peers are catered, while the social cost decreases linearly through all rounds (Fig. 1(left)).

4.2 Changing Query Workload and Content

The goal of this set of experiments is to evaluate our strategies with regards to sustaining system quality under changing conditions. The initial system configuration is the clustering achieved for the first scenario of the previous set of experiments. We maintain the number of clusters fixed and the only change we allow is the relocation of peers to different non-empty clusters. We assume that the total query workload is assigned uniformly to peers (each peer is assigned the same portion) and report only the social cost. The threshold value of the gain used as a stop condition is $\epsilon = 0.001$.

We consider updates affecting peers in a single cluster, say cluster c_{cur}. These updates follow one of the following two scenarios: (a) affect a varying number of peers in c_{cur} or (b) affect all the peers in c_{cur} with a varying degree. For both

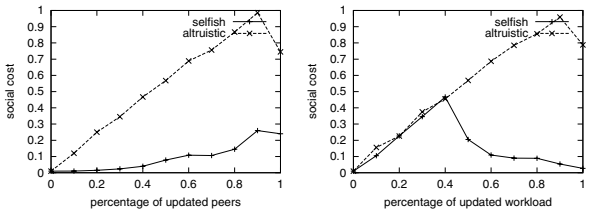

Figure 2: Social Cost for different percentages of updated (left) peers and (right) query workload

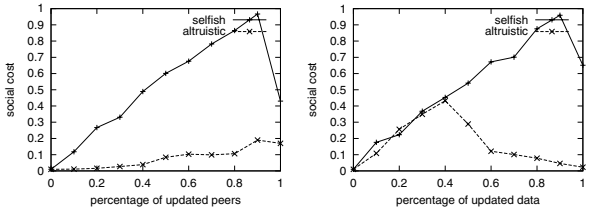

Figure 3: Social Cost for different percentages of updated (left) peers and (right) data

scenarios, we apply our strategies until no more relocation requests are issued and measure the social cost.

First, we consider workload updates. For the first scenario, the workload of a varying number of peers in c_{cur} changes completely, that is, while the peers were interested in data located in c_{cur}, now they become interested in data located at some other cluster c_{new}. For the second scenario, the query workload of all peers in c_{cur} changes by a varying percentage. Figure 2 reports the social cost for each round of the protocol for the first (Fig. 2(left)) and the second (Fig. 2(right)) scenario. In both scenarios, for the selfish strategy, the individual cost of the peers whose workload change increases immediately and these peers issue requests to move to c_{new}. However, there needs to be a significant change in the peers query workload (above 50%), for improving the overall social cost, since these movements would result in increasing the individual cost of those peers whose workload has not changed. For the altruistic strategy, the peers from c_{new} that hold the data requested by peers in c_{cur} would start to move to c_{cur} to cater the new workload. However, for the peers in c_{new} to move, the number of peers in c_{cur} that need their data must become equal or larger to the number of peers in c_{new} that they are currently serving. This requires a significant change in the workload of c_{cur}. Note that none of the scenarios achieves the initial social cost, since by adding new peers to clusters, their size increases along with the peer membership cost.

We now vary the data content in c_{cur}, instead of the local workload of its peers for both scenarios. In particular, the data in the cluster are replaced by data belonging to a different category. As Fig. 3 shows, the altruistic strategy performs similar to the selfish one in the previous workload based experiment. This is because when the content of a peer changes, an altruistic peer no longer offers data to the cluster it belongs to, thus, it is motivated to change cluster. Whereas, selfish peers have no motive to leave the cluster, since their query workload does not change and the specific category of the data that they request is not contained in any other cluster.

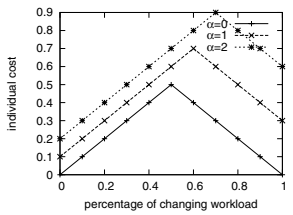

Figure 4: Influence of α

Finally, we examine the influence of parameter α. We consider the case of peers following the selfish strategy and evaluate the individual cost of a single peer when its query workload gradually changes over time. As the value of α increases, the membership cost becomes more expensive, thus a larger portion of the query workload needs to change for a peer to benefit from joining a cluster with more members (Fig. 4).

5. RELATED WORK

Game theoretic approaches have been applied to model the behavior of peers in p2p systems. In [3], the creation of an Internet-like network is modeled as a game with peers acting as selfish agents without central coordination. The aim of the game is for each peer to choose the peers with which to establish links. The peers pay for the creation of a link, but gain by reducing the shortest distance to any other peer in the system. In our approach, instead of establishing links randomly, we consider content and query workload for creating clusters of peers with similar properties. [5] considers a more sophisticated model, in which strict bounds are enforced on a peer's out-degree, links are directed and peers are allowed to express preferences regarding the choice of their neighbors. In a way, our approach can be viewed as setting these preferences based on recall benefits. In [7], the authors show that allowing peers to act completely freely is much worse than collaboration and prove that even a static p2p of selfish peers may never reach convergence. This result is in accordance to our findings that show that in only specific scenarios, we reach a Nash equilibrium. [9] considers altruistic peers that decide the level of their contribution to the system based on a utility function that depends on a variety of parameters such as the amount of data they upload and download, unlike our altruistic strategy where the choice of the cluster a peer joins depends on its contribution to this cluster.

Recent research efforts are focusing on organizing peers in clusters. In most cases, the authors focus on cluster formation and query processing using the clusters and do not address the adaptation of the overlay to changing conditions. In [1], peers are partitioned into topic segments based on their documents. A fixed set of M clusters is assumed, each one corresponding to a topic segment. Knowledge of the M centroid is global. Clusters of peers are formed in [8] based on the semantic categories of their documents; the semantic categories are predefined. Similarly, [2] assumes predefined classification hierarchies based on which queries and documents are categorized. Clustering in [6] is based on the schemes of the peers and on predefined policies provided by human experts. Besides clustering based on peers content, clustering based on other common features, such as the interests of peers [4], is possible.

6. CONCLUSIONS

In this paper, we have modeled peers in a clustered overlay as players that dynamically change the set of clusters they belong to according to an individual cost function, which is based on a cluster membership cost and the recall for their local query workload. Each peer aims at minimizing its individual cost, therefore maximizing its recall. In addition, we have defined cost criteria for measuring the global system quality. We have proposed strategies based on different motives behind the peers behavior (selfish or altruistic) and showed how by following them, the peers can change the clustered overlay to reflect the current system conditions thus, maintain its quality under updates. Our experimental results showed that our strategies are able to cope with the changes and gradually correct system performance. Furthermore, they indicated that the proposed strategies can also be used for the discovery of clusters in p2p systems, when the underlying data distribution permits it.

There are many open issues for future work. One issue is to derive some theoretical results regarding convergence perhaps by considering more restricted forms of the cluster reformulation problem. Also, practical issues such as the maximum number of clusters that a realistic p2p system can support and the expected look-up cost with respect to the number of clusters and their sizes, need to be addressed. Furthermore, there are variations to the proposed strategies that may be worth exploring, for example, a hybrid strategy taking into consideration both the individual cost and the contribution measure and also variations of the game, i.e., with asynchronous players.

7. REFERENCES

[1] M. Bawa, G. Manku, and P. Raghavan. Sets: Search enhanced by topic segmentation. In *SIGIR*, 2003.

[2] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems, technical report, computer science department, stanford university, 2002.

[3] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *PODC*, 2003.

[4] M. Khambatti, K. Ryu, and P. Dasgupta. Efficient discovery of implicitly formed peer-to-peer communities. *IJPDSN*, 5(4):155–164, 2002.

[5] N. Laoutaris, G. Smaragdakis, A. Bestavros, and J. W. Byers. Implications of selfish neighbor selection in overlay networks. In *INFOCOM*, 2007.

[6] A. Loser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden. Semantic overlay clusters within super-peer networks. In *DBISP2P*, 2003.

[7] T. Moscibroda, S. Schmid, and R. Wattenhofer. On the topologies formed by selfish peers. In *PODC*, 2006.

[8] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards high performance peer-to-peer content and resource sharing systems. In *CIDR*, 2003.

[9] D. K. Vassilakis and V. Vassalos. Modelling real p2p networks: The effect of altruism. In *P2P*, 2007.