A Recall-Based Cluster Formation Game in Peer-to-Peer Systems

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

ABSTRACT

In many large-scale content sharing applications, participants or peers are grouped together forming clusters based on their content or interests. In this paper, we deal with the maintenance of such clusters in the presence of updates. We model the evolution of the system as a strategic game, where peers determine their cluster membership based on a utility function of the query recall. Peers are guided either by selfish or altruistic motives: selfish peers aim at improving the recall of their own queries, whereas altruistic peers aim at improving the recall of the queries of other peers. We study the evolution of such clusters both theoretically and experimentally under a variety of conditions. We show that, in general, local decisions made independently by each peer enable the system to adapt to changes and maintain the overall recall of the query workload.

1. INTRODUCTION

Large content sharing applications such as social networks and peer-to-peer (p2p) file sharing systems have become highly popular. Measurements from the deployment of such large-scale systems have shown that the interactions among their participants (peers) indicate the existence of implicit groups (*clusters*) of peers having similar content or interests. For example, the formation of implicit groups centered around topics described by common keywords has been observed in the blogosphere [2]. In measurements of popular on-line social networks [16], it was also observed that the network structure is such that users form clusters based on common interests, social affiliations or the wish to exploit their shared content.

We particularly focus on clustering in p2p systems. In such systems, peers form clusters by creating logical links to other peers that share similar content or interests, thus, creating a *clustered overlay network* on top of the physical one. The underlying reason behind the formation of such clusters is that they enable the peers to find and exchange data relevant to their interests with less effort. The clustered overlay is exploited for routing the queries that the users pose for locating content of interest. Once the appropriate cluster for a query is identified, the peers in the cluster possess relevant content that can be exploited to evaluate and refine the query efficiently. In particular, traces of popular p2p systems have indicated that peers exhibit the property of interest-based locality, that is, if a peer holds content satisfying some query of another peer, then it is most likely that it also maintains additional content of interest to this other peer [18, 10]. Thus, placing the two peers in the same cluster would increase the recall of their queries.

While, there is a large body of research on the discovery and construction of clustered overlays [3, 5, 15, 21, 11, 8, 4, 6], their maintenance, which is imperative for coping with the dynamic nature of peers, has been mostly ignored.

In this paper, we study the dynamics of clustered overlay networks by adopting a game-theoretic perspective. We model the problem of cluster formation as a strategic game with peers as the players. Each peer plays by selecting which clusters to join. This selection or strategy is determined individually by each player, so as to minimize a utility function that depends on the membership cost entailed in belonging to a cluster and the cost of evaluating its query workload at remote clusters. Game-theoretic models have been proposed for creating overlays based on the connection cost and radius of the network graph [7, 14, 17]. The originality of our approach lies on the fact that we consider clustered overlays, focus on queries and aim at increasing their recall.

We model both selfish and altruistic behavior of peers as demonstrated in real content-sharing systems by proposing appropriate utility functions. We also introduce global system quality criteria to measure the performance of the system as a whole.

To cope with dynamics, our game is a repeated one: peers re-evaluate their strategy and potentially relocate to other clusters. We define appropriate relocation policies for both selfish and altruistic peers and propose an uncoordinated cluster reformulation protocol based on local decisions made independently by each peer. We study both theoretically and experimentally the evolution of clusters under the individual actions of each peer. Our experimental results show that the uncoordinated protocol efficiently copes with the changes in the overlay, while maintaining approximately the same quality with a coordinated protocol that relies on global decisions.

The rest of this paper is organized as follows. In Section 2, we present the cluster formation problem as a game and define the utility functions for selfish and altruistic peers along

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.
VLDB '09, August 24-28, 2009, Lyon, France
Copyright 2009 VLDB Endowment, ACM 978-1-60558-948-0/09/08

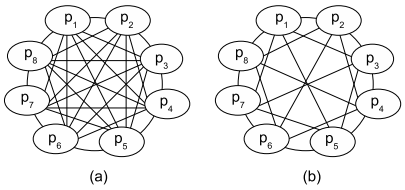

Figure 1: Examples of cluster topologies

with corresponding global quality criteria. In Section 3, we study stability and optimality. In Section 4, we describe our reformulation protocol for cluster maintenance. Section 5 presents our experimental evaluation, and Section 6 refers to related research. Section 7 concludes the paper.

2. RECALL-BASED CLUSTERING

We consider a distributed system consisting of highly dynamic nodes (peers) that share content. Usually, such distributed systems need to scale up to a large number of peers (Internet-scale). Thus, a peer is unable to know and directly communicate with all other peers in the system. Instead, it establishes logical links with only a few other peers, creating logical overlay networks on top of the physical one. Queries are routed through this overlay to locate peers that hold content of interest.

The efficiency of query evaluation depends heavily on the topology of the overlay network. In this paper, we consider *clustered overlays* in which peers with similar content or interests form groups, called *clusters*. In such overlays, the peers inside each cluster usually follow a topology that ensures high connectivity, thus, making the evaluation of queries within a cluster very efficient. For example, Fig. 1 shows 8 peers forming a cluster following a fully connected topology (Fig. 1(a)), where each peer can reach any other peer in the cluster with 1 hop, while (Fig. 1(b)) a structured Chord-like ([19]) topology in which finding any peer takes at most log(8) hops.

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. We denote the number of results for a query q against the documents of peer p_i as $result(q, p_i)$.

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_i in the evaluation of a query q in Q based on the results that p_i offers for q with regards to the total number of available results (i.e. the recall achieved when q is evaluated solely on p_i). Specifically:

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

We also define as *local workload* of peer p_i, $Q(p_i)$, the list of queries 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 $Q(p_i)$.

Clustering as a Game: We model the problem of cluster formulation as a strategic game. Each peer p_i represents a player in the game and its strategy s_i is defined by the set of clusters it joins. In particular, each peer p_i 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 $s_i \subseteq C$. For example, consider $P = \{p_1, p_2, p_3, p_4\}$ and $C = \{c_1, c_2, c_3\}$, and let us assume that p_1 belongs to clusters c_1 and c_2, p_2 belongs to c_1, p_3 to c_3 and p_4 to c_2 and c_3. Then, the corresponding strategies are $s_1 = \{c_1, c_2\}$, $s_2 = \{c_1\}$, $s_3 = \{c_3\}$ and $s_4 = \{c_2, c_3\}$.

We can describe any cluster configuration by the set of strategies $S = \{s_1, s_2, \ldots, s_{|P|}\}$ that the peers in P deploy, since from this set, we can derive the set of peers belonging to each cluster in C. In this paper, we constrain C_{max} to be equal to $|P|$, i.e. it cannot exceed the number of peers, and assume that some clusters may be empty if needed. To cope with peer dynamics, each peer plays more than once, thus, the cluster configuration is not static.

The goal of the game is for each player (peer) to minimize or maximize a *utility* function. We discern between two types of peers, *selfish* and *altruistic* ones, and define a corresponding utility function for each type.

2.1 Individual Peer Measures

A selfish peer is interested in increasing the recall of its local query workload by joining those clusters whose peers would increase the recall of its local workload the most. Specifically, let $P(s_i)$ be the set of all peers belonging to any cluster $c \in s_i$. The gain for a peer p_i for choosing 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 for p_i associated with s_i 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 recall-based 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 and the topology of the cluster. The larger the size of the cluster, the higher the cost of joining, leaving and maintaining the cluster. Furthermore, a highly connected topology, where each peer maintains links to a large number of other peers, increases the cluster membership cost. To capture this, the cluster membership cost is defined as a monotonically increasing function θ of the number of peers belonging to the cluster, i.e. as a function of the cluster size $|c|$. This function depends on the cluster topology, for instance, when all peers are connected to each other (Fig. 1(a)), θ may be linear, whereas in the case of structured overlays (Fig. 1(b)), θ may be logarithmic.

Definition 1 (Individual Peer Cost). *In a cluster configuration S, the individual cost for a selfish peer p_i for choosing strategy s_i is:*

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

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. The recall loss of each query is weighted by its frequency in the local workload of p_i. Parameter α ($\alpha \geq 0$) determines the extent of influence of the cluster membership cost in cluster formation. From a system perspective, parameter α characterizes the ratio between updates and queries in the system. For a given θ, a large value of α means that updates in the

system are rather frequent and therefore the cost for cluster maintenance is high, while a small value indicates that query evaluation efficiency is more important for determining the overall system performance. Finally, factor $1/|P|$ is used for normalizing the cluster membership cost.

Observe that the two terms of the cost function tend to guide the peer towards selecting opposing strategies. For example, assume that a peer can join only one cluster and that $\alpha \geq 1$. In a cluster configuration in which all peers form a single cluster, the membership cost, that is θ, is maximized ($\theta(|P|)$), while the recall loss is minimized (0) since for any peer all results for its queries are located within its cluster. In contrast, the recall loss is maximized when p_i forms a cluster by its own, while the membership cost is in this case minimized.

In addition to modeling the behavior of selfish peers, we also want to model the behavior of altruistic peers that are not concerned about their own queries, but instead, about offering to other peers. Therefore, we define the corresponding utility function, called *individual peer contribution* (*pcontr*) that an altruistic peer p_i aims at maximizing based on how much p_i improves the recall of the other peers that belong to the clusters of its strategy. Thus, analogously to Def. 1, the individual contribution is defined as follows:

Definition 2 (Individual Peer Contribution). *In a cluster configuration S, the individual contribution of an altruistic peer p_i by choosing strategy s_i is:*

$$pcontr(p_i, S) = \frac{1}{|P|} \sum_{p_j \in P(s_i)} \sum_{q \in Q(p_j)} \frac{num(q, Q(p_j))}{num(Q(p_j))} r(q, p_i) - \frac{\alpha}{|P|^2} \sum_{c_k \in s_i} |c_k| \theta(|c_k|)$$

While *pcost* measures the cost p_i pays for its query workload and membership to clusters in s_i, *pcontr* is a positive measure showing what other peers gain when p_i chooses strategy s_i. The first term of the sum measures the contribution of peer p_i to the peers in the clusters of its strategy, while the second term measures the membership cost these peers pay if p_i joins their clusters. Similarly to the individual cost, the membership cost also takes its lowest value when the peer forms a cluster by its own and its largest when all peers form a single cluster (if we consider that each peer joins only a single cluster), whereas the recall it contributes to other peers takes its largest value in the single cluster and its lowest when it forms a cluster by its own. Individual contribution is defined from the perspective of each beneficiary peer, that is, the queries weights are defined based on their relative frequencies per such peer.

Besides the pure selfish and the pure altruistic behavior, hybrid behavior can be captured by trying to minimize the following cost function:

$$hpcost(p_i, S) = d\ pcost(p_i, S) - (1 - d)\ pcontr(p_i, S),$$

where $d \in [0, 1]$ captures the degree of selfishness of peer p_i. A hybrid peer considers both its own cost (with degree d) and its contribution to the others (with degree $1 - d$).

Finally, note that our game is a non-cooperative asymmetric game. A game is *asymmetric*, if the value of the utility function or *payoff* differ if different players select the same strategy.

2.2 Global Cost Measures

We measure the overall quality of a cluster configuration by the achieved *social cost* ($SCost$) defined as:

Definition 3 (Social Cost). *The social cost of a cluster configuration S is defined as the sum of the individual costs of all peers in P:*

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

We can also evaluate the overall quality of the configuration from a query workload perspective, by considering the average cost for attaining results for all queries in Q.

Definition 4 (Workload Cost). *The workload cost of a cluster configuration S is:*

$$WCost(S) = \alpha \sum_{c_k \in C} \frac{|c_k| \theta(|c_k|)}{|P|} + \sum_{q\ in\ Q} \frac{num(q, Q)}{num(Q)} \sum_{p_i\ s.t.\ q\ in\ Q(p_i)} \frac{num(q, Q(p_i))}{num(q, Q)} \sum_{p_j \notin P(s_i)} r(q, p_j)$$

The first term expresses the cost for maintaining the clusters. The second term expresses the cost for all queries, i.e., the cost for evaluating them outside the clusters of their initiator.

The main difference between the social and the workload cost lies on 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, but for equally demanding peers the following proposition holds.

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

Proof. Using the definition of individual cost (Def. 1), the social cost can be written as:

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

The membership cost 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 $|c_k|$. The second term differs from the second term of $SCost$ only on how much the workload of each peer is taken into account. 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.□

Proposition 1 implies that improving the social cost improves the workload cost and vice versa.

In accordance to the social and workload cost, we define the corresponding social and workload contribution as:

Definition 5 (Social Contribution). *The social contribution of a cluster configuration S is defined as the sum of the individual contributions of all peers in P:*

$$SContr(S) = \sum_{p_i \in P} pcontr(p_i, S)$$

DEFINITION 6 (WORKLOAD CONTRIBUTION). *The workload contribution for a cluster configuration S is:*

$$WContr(S) = \sum_{q\ in\ Q} \frac{num(q,Q)}{num(Q)} \sum_{p_i\ s.t.\ q\ in\ Q(p_i)} \frac{num(q,Q(p_i))}{num(q,Q)} \sum_{p_j \in P(s_i)} r(q,p_j) - \frac{\alpha}{|P|^2} \sum_{p_i \in P} \sum_{c_k \in s_i} |c_k| \theta(|c_k|)$$

Similarly to $SCost$ and $WCost$, the $SContr$ and $WContr$ are also proportional for specific workload distributions, in particular, when the query workload is uniformly distributed among the peers.

PROPOSITION 2. *If for all p_i, $p_j \in P$ and all q in Q, $num(q,Q(p_i))/num(Q(p_i)) = num(q,Q(p_j))/num(Q(p_j)) = num(q,Q)/num(Q)$, the social and the workload contribution measures are proportional to each other.*

Intuitively, social contribution favors queries that are popular to specific peers, whereas its workload counterpart favors overall popular queries.

Let us now examine the relationship between the workload cost and the workload contribution.

PROPOSITION 3. *For $\alpha = 0$, that is, if ignore the cluster membership cost, it holds: $WCost(S) = 1 - WContr(S)$, which means that the two measures are complementary.*
Proof. It holds that:

$$\sum_{p_j \notin P(s_i)} r(q,p_j) + \sum_{p_j \in P(s_i)} r(q,p_j) = 1, \forall q\ in\ Q, s_i \in S. \quad (1)$$

For $\alpha = 0$, we have: $WCost(S) = 1 - WContr(S)$.□

Let us consider now, the social cost and the social contribution.

COROLLARY 1. *For uniform query workload among peers the social cost and social contribution are complementary: $SContr(S) = 1 - SCost(S)$.*
Proof. Again, for $\alpha = 0$, we can rewrite $SCost(S)$ using (1) and if we assume that $\frac{num(q,Q(p_i))}{num(Q(p_i))}$ is the same for all peers p_i, then we have that: $SContr(S) = 1 - SCost(S)$.□

3. STABILITY AND OPTIMALITY

The goal of each player (peer) is to minimize/maximize its individual cost/contribution. We will refer in the following to selfish peers, but the same results are applicable for altruistic behavior.

3.1 Stability

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)? That is, will the system reach a Nash equilibrium?
NASH EQUILIBRIUM: 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 p_i):

$$pcost(p_i, S) \leq pcost(p_i, S') \quad (2)$$

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.

We shall first prove an interesting property of the cluster formation game. Due to the form of our cost function, the stable states in our system have the following property that constraints the number of possible configurations:

LEMMA 1. *In any stable state, there are no clusters c_i, c_j such that $c_i \subseteq c_j$, $i \neq j$.*
Proof. Let S be a cluster configuration, c_i, c_j be two clusters in C such that $c_i \subseteq c_j$. Consider a peer p_k, $p_k \in c_i$. Clearly, $p_k \in c_j$. Let the individual cost of p_k be: $pcost(p_k, S) = \alpha\gamma + \delta$, where γ is the membership cost for p_k when following strategy $s_k \in S$ and δ the respective recall it loses from the peers that do not belong to $P(s_k)$. Assume for the purposes of contradiction that S describes a stable configuration, then p_k can not select a strategy that would reduce its cost. Let us examine the strategy $s'_k = s_k - \{c_i\}$. Let S' be the configuration resulting by replacing s_k with s'_k in S. Then, $pcost(p_k, S') = \alpha(\gamma - \frac{\theta(|c_i|)}{|P|} + \delta) < pcost(p_k, S)$. The recall part of the cost function remains the same, because $P(s_k) = P(s'_k)$. Thus, p_k can reduce its cost by selecting the strategy s'_k, and therefore S is not a stable state, which contradicts our assumption.□

Because of Lemma 1, it holds:

COROLLARY 2. *When a peer forms a cluster by itself, it cannot belong to any other cluster.*

It is rather simple to show that for the cluster formation game, a pure Nash equilibrium does not always exist.

PROPOSITION 4. *A pure Nash equilibrium does not always exist for the cluster formation game.*
Proof. Let us consider a simple scenario of two peers p_1 and p_2. Consider also that $Q(p_1)$ consists of a single query q_1 satisfied by p_2 (i.e. $r(q_1, p_2) = 1$) and $Q(p_2)$ consists of q_2 also satisfied by p_2. Let $C = \{c_1, c_2\}$ be the clusters in the system. Using Lemma 1, the following cluster configurations are possible: $p_1 \in c_1$ and $p_2 \in c_2$, described by $S_1 = \{\{c_1\}, \{c_2\}\}$, $p_1 \in c_2$ and $p_2 \in c_1$, described by $S_2 = \{\{c_2\}, \{c_1\}\}$ and both $p_1, p_2 \in c_1$ or c_2 described by $S_3 = \{\{c_1\}, \{c_1\}\}$ and $S_4 = \{\{c_2\}, \{c_2\}\}$, respectively. Let us assume a linear θ function, $\theta(n) = n$. 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: $pcost(p_1, S_1) = \alpha\frac{1}{2} + 1$ and $pcost(p_2, S_1) = \alpha\frac{1}{2}$. If p_1 moves to cluster c_2, then the system configuration becomes $\{\{c_2\}, \{c_2\}\}$, that is, configuration S_4, and the cost for p_1 becomes $pcost(p_1, S_4) = \alpha \leq pcost(p_1, S_1)$. Thus, configuration S_1 is not a Nash equilibrium, since p_1 can reduce its cost by moving to c_2. Let us consider now the configuration S_3 (S_4 is symmetric) in which both peers belong to the same cluster. Their individual costs are now: $pcost(p_1, S_3) = \alpha$ and $pcost(p_2, S_3) = \alpha$. Peer p_2 can reduce its cost by moving to the (empty) cluster c_2 (resulting in configuration S_1) and therefore S_3 is not a Nash equilibrium. Table 1 summarizes the payoff (cost) table for this two-player game.□

3.2 Social Optimum

Even if the system does eventually reach a stable state (Nash equilibrium), it is not always the case that this stable state has a satisfying cost. A measure widely used for evaluating how far from the best possible outcome a stable state is, is the *price of anarchy* defined as the ratio between the social cost of the worst Nash equilibrium and the "social optimum". The social optimum is obtained by minimizing

the social cost measure over all possible configurations, even for those configurations that do not correspond to a stable state.

We can acquire a rough bound of the social optimum by considering each peer separately and evaluating its individual cost over all possible configurations. Then, by selecting for each peer the configuration that yields the minimum individual cost and adding these values, we obtain a bound for the minimum value of the social cost in the system, i.e., for the social optimum. Note that we are adding together individual costs that may correspond to different configurations, thus, the estimated social cost may refer to a configuration that cannot exist and may be very far from the actual value of the social optimum that we can achieve.

3.3 Case Studies

Although, in the general case, a Nash equilibrium does not always exist, there are cases in which, for specific configurations and data and query workload distributions, stable clusters may be formed. Next, we present two scenarios:

Case I: No Underlying Clustering: In this case, all peers in P are considered similar in the following sense:

$$num(Q(p_i)) = num(Q(p_j)) = num(Q)/|P|, \forall p_i, p_j \in P$$
$$r(q, p_i) = r(q, p_j) = 1/|P|, \forall q \text{ in } Q, \forall p_i, p_j \in P$$

This corresponds to a data and query distribution for which no physical grouping among the peers exist. Note that our game becomes a symmetric one, since all players yield the same payoffs when applying the same strategy.

Case II: Symmetric Clusters: In this case, the data and query distribution are such that a perfect underlying clustering/grouping exists among the peers. In particular, the peers in P belong to m ($m > 1$) different groups of the same size $|c| = |P|/m$. The members in each group offer and demand data only within their group. Formally, for all pairs of peers p_i, p_j in the same group, it holds $num(Q(p_i)) = num(Q(p_j))$ and $\forall q$ in $Q(p_j)$, $r(q, p_i) = 1/|c|$, whereas for all pairs of peers p_i, p_j not in the same group, the lists $Q(p_i)$ and $Q(p_j)$ have no queries in common and $\forall q$ in $Q(p_j)$, $r(q, p_i) = 0$.

For each of these two scenarios, we consider a number of cluster configurations and study each of them in terms of stability and optimality.

Stability

To determine whether a cluster configuration constitutes a Nash equilibrium, we need to ensure that the individual cost of any peer is not smaller in any possible configuration that can result from the current one by changing only the strategy of this peer, by evaluating Inequality (2).

For the first scenario (Case I), we study the following cluster configurations:

Case(I.a): A single cluster. In this case, all peers form a single cluster. From Corollary 2, the only way a peer p_i can change its strategy is by forming a cluster by its own.

Case(I.b): Each peer forms a cluster by its own. In this case, each peer forms a cluster by its own. The only way for a peer p_i to change its strategy is to leave its own cluster and join k other clusters, where $1 \leq k \leq |P| - 1$.

Case(I.c): m non-overlapping clusters. The peers form m *non-overlapping clusters* of the same size $|c|$. Consider a peer $p_i \in c_j$. The available options for p_i for changing its strategy are to: (1) form a cluster by its own; (2) additionally to c_j, join k other clusters, where $1 \leq k < m$; or (3) leave c_j and join k other clusters.

Table 2(line 1) presents the results of our evaluation for selfish peers that aim at minimizing their individual cost, while Table 2(line 2) for altruistic peers that aim at maximizing their contribution.

This shows that, even if there is no underlying clustering according to the data and query workload distribution, a system can still reach a stable state. This state depends on the cluster maintenance costs and the portions of data and query workload each peer offers or demands.

To make this more concrete, let us assume a θ function corresponding to a linear function of the form: $\theta(n) = \lambda n$, $0 < \lambda \leq 1$, and rewrite the conditions regarding α. Then, a configuration in which all peers belong to a single cluster is stable for $\alpha \leq 1/\lambda$. Recall that large values of α mean that maintenance costs are more important than query recall. Thus, for the same θ, for values of α larger than this threshold, the maintenance cost would surpass those gained by recall and would lead to splitting the cluster. Note also, that whether a single cluster is stable or not depends also on the topology as captured through function θ. For instance, when λ is small (less connected topology), a single cluster remains stable for larger values of α.

For the symmetric clusters scenario, we limit our analysis to the case in which each peer can belong only to one cluster and study the same configurations as in Case I.

Case(II.a): A single cluster. Same as Case (I.a).

Case(II.b): Each peer forms a cluster by its own. The only option for p_i is to join another peer p_j, which either is in the same group with p_i, or belongs to a different group. This configuration is the same, whatever group out of the $m-1$ we consider, since all such peers are symmetric to p_i, i.e., they do not satisfy any of its local query workload.

Case(II.c): m non-overlapping clusters. In this case, we consider that each of the m clusters contains peers of a single group. Then, the individual peer cost for each peer $p_i \in P$ is equal to its cluster membership cost, since the cost for computing queries outside its cluster is zero (there are no results for $Q(p_i)$ in peers not in $P(s_i)$). If p_i wants to change its strategy s_i, then it can move either to a cluster on its own or to a different existing cluster.

The results of the same analysis as in the first case are presented in Table 2(lines 3) for selfish peers. The results for altruistic ones can be easily computed and are omitted.

By comparing Case I (no underlying clustering) with $k = 1$ and Case II (perfect underlying clustering), we see that in Case II, configuration (b) in which each peer forms its own cluster (no clustering) is stable for larger values of α, whereas the other two configurations (a) and (c) (with some form of clustering) are stable for smaller values of α.

Social Optimum

We examine whether any of the Nash equilibria that we have previously computed achieve a social cost equal to the social optimum. To this end, we need to compare their social cost against that of any other possible configuration. We assume selfish peers and a linear θ function. Our results can be easily adapted for altruistic peers.

In Case I, where all peers are symmetric, minimizing the individual cost of any peer suffices to minimize the social cost.

Case(I.a): A single cluster. We already know that a configuration in which each peer forms its own cluster has a

Table 1: Payoff Table

	p_2 joins c_1	p_2 joins c_2
p_1 joins c_1	α,α	$\frac{\alpha}{2}+1, \frac{\alpha}{2}$
p_1 joins c_2	$\frac{\alpha}{2}+1, \frac{\alpha}{2}$	α,α

larger cost than Case (I.A), since we assume that $\alpha \leq 1/\lambda$ and Case (I.A) is an equilibrium. The only other possible configuration is when a peer p_i joins k clusters with different sizes. The best case (the case with the lowest cost) is the one where the k clusters have no overlapping members. It also holds that $|P(s_i)| < |P|$, otherwise we would have a single cluster. By comparing the social cost of this configuration to the cost of Case (I.A), we see that for $\alpha \leq 1/\lambda$, Case (I.A) has the lowest cost. Thus, the value of the cost of Case (I.A) corresponds to the social optimum.

By applying a similar analysis, we conclude that: Case (I.B) has a cost equal to the social optimum for $a \geq 1/\lambda$, while Case (I.C) does not reach the social optimum for any $m > 1$.

For Case II, the corresponding conclusions are: Case (II.A) does not reach the social optimum for any value of $\alpha > 0$, since separating the m groups always results in a configuration with a lower social cost. Case (II.B) and Case (II.C) correspond to states with cost equal to the social optimum for $a \geq m/\lambda$ and $\alpha \leq 1/\lambda$, respectively.

A detailed analysis of the above can be found in [13].

4. CLUSTER EVOLUTION

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 peer content and 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 policies that each peer follows so as to move to the most appropriate cluster under the given system conditions. Such protocols can also be used to bootstrap the system, for example, by applying them on an initial configuration in which all peers belong to a single cluster or each peer forms a cluster by its own. We describe first the relocation policies followed by each peer, and then how they are applied to form a new cluster configuration.

4.1 Relocation Policies

Unlike most network creation games, our game is not a one-shot game but a repeated one, where the peers reexamine their strategy selection through time to cope with the system dynamics.

Let S_{cur} be the current cluster configuration. When it is its turn to play, each peer p_i considers all possible configurations S_j that differ from S_{cur} only at their i-th component, i.e., the strategy s_i that peer p_i follows. For simplicity, in the rest of this paper, we focus on the case where each peer belongs to a single cluster. Let C_{cur} be the current set of (nonempty) clusters in the system. Then, for each peer p_i, it holds: $s_i = \{c_l\}$, for some $c_l \in C_{cur}$. In this case, the possible strategies for p_i besides its current one are: either moving to a cluster c_v, $c_v \neq c_l$ for $c_v \in C_{cur}$ or if $c_l \neq \{p_i\}$, forming a cluster by its own.

Based on the behavior of each peer, we consider two types of relocation policies: *selfish* and *altruistic*. A peer with a selfish policy chooses the strategy s_{new} for which the corresponding cluster configuration S_{new} is:

$$S_{new} = \arg\min_{S_j} pcost(p_i, S_j)$$

Analogously, a peer with an altruistic policy chooses the strategy s_{new} for which the corresponding S_{new} is:

$$S_{new} = \arg\max_{S_j} pcontr(p_i, S_j)$$

A hybrid relocation policy that uses the hybrid peer cost, *hpcost*, is also feasible.

To measure how much a peer benefits from moving to a new cluster, we define a new measure, the *gain*. The gain is defined for a selfish peer as:

$$gain_{p_i} = pcost(p_i, S_{cur}) - pcost(p_i, S_{new})$$

If $gain_{p_i} > 0$, then p_i benefits from selecting the new strategy. Analogously, gain is defined for altruistic peers.

To implement the policies, we assume that each cluster has a unique identifier, *cid*, known by all its peers, which is assigned based on peer IPs and timestamps. For example, when the first peer joins a cluster, its *cid* is formed by the IP of the peer concatenated with a timestamp. When other peers join the cluster, they are informed of its *cid*. Query results are annotated with the corresponding *cid*s of the clusters that provide them. Thus, peers do not need to know all system *cid*s, but they gradually learn them, as their queries acquire results annotated with new *cid*s. Therefore, when all peers leave a cluster, its *cid* just becomes unused. Recycling *cids* is beyond the scope of this paper.

In the selfish relocation policy, since, all query results received by a peer are annotated with the *cid* of the cluster they came from, each peer can monitor its recall with respect to all clusters in the system and use it to evaluate its individual cost for the different configurations it needs to consider when it plays. In the altruistic relocation policy, instead of its recall, each peer records the number of results it sends to each cluster so as to evaluate its individual contribution for all different s_i components.

Since no global view of the system is available, a peer can not be aware of all available results for a query and we instead use as recall, in the cost evaluation, the fraction of results returned to peer p_i for query q by a cluster c_j to the total number of results returned for the query.

4.2 Cluster Reformulation Protocol

The relocation policies along with the gain are used to form the reformulation protocols.

Coordinated Protocol. We first consider a *coordinated reformulation protocol*. Cluster representatives are used to achieve this coordination by gathering and exchanging information about their clusters. Each peer applies its relocation policy and determines the cluster it needs to move to. Then, all relocation requests are gathered by the representatives and ordered according to non-increasing value of gain. Each representative grants a percentage x of them (Alg. 1).

The cluster representative does not need to remain the same. Representative selection is local within each cluster and may be random or based on specific properties of the peers. When a peer stops acting as a representative, it suffices to redirect all requests to the new representative. Furthermore, for a peer to join a cluster it just needs to know one of its members. It then sends a relocation request to that member, which forwards the request to the current cluster representative.

Uncoordinated Protocol. We argue that the use of coor-

Table 2: Conditions for Stability

	Case (I.a)	Case (I.b)	Case (I.c)
Cost-based	$\alpha \leq \frac{\|P\|-1}{\theta(\|P\|)-\theta(1)}$	$\alpha \geq \frac{k}{k\theta(2)-\theta(1)}$	$\alpha \leq \frac{\|c\|-1}{\theta(\|c\|)-\theta(1)}$, $\alpha \geq \frac{\|c\|}{\theta(\|c\|+1)}$, $\alpha \geq \frac{k(\|c\|-1)+1}{k\theta(\|c\|+1)-\theta(\|c\|)}$
Contribution-based	$\alpha \leq \frac{\|P\|-1}{\|P\|\theta(\|P\|)-\theta(1)}$	$\alpha \geq \frac{2k-1}{2k\theta(2)-\theta(1)}$	$\frac{1}{\theta(\|c\|)} \leq \alpha \leq \frac{\|c\|-1}{\|c\|\theta(\|c\|)-\theta(1)}$
	Case (II.a)	Case (II.b)	Case (II.c)
Cost-based	$\alpha \leq \frac{\|P\|(\|P\|-m)}{m\theta(\|P\|)-\|P\|\theta(1)}$	$\alpha \geq \frac{m}{\theta(2)-\theta(1)}$	$\frac{\|P\|^2-m}{\|P\|(\theta(\|P\|/m+1)-\theta(\|P\|/m))} \leq \alpha \leq \frac{(\|P\|-m)}{(\theta(\|P\|/m)-\theta(1))}$

dination is not necessary and instead propose an *uncoordinated reformulation protocol* in which each peer determines when to play locally and independently from the other peers. When a peer determines that it is its turn to play, it applies its relocation policy locally and moves to the cluster the policy indicates. Cluster representatives may be used to facilitate moves between clusters and reduce the overhead.

Protocol Variations. Based on *when* the peers determine that there is their turn to play, we define two variations of the reformulation protocol: an *event* and a *trigger-based* one.

The coordinated event-based protocol is initiated after each system event, i.e., a query or an update. In the uncoordinated event-based protocol, a peer determines that it is its turn to play after it becomes aware of a relevant event. Selfish peers consider as relevant the evaluation of queries of their local query workload, while altruistic peers the provision of results to another peer's query. A hybrid peer may choose either one or both types of events as relevant. A variation of the event-based protocol, the *batch-based* protocol, is initiated after a number (batch) of events instead of just after a single one.

In the coordinated trigger-based protocol, the social or workload cost (or corresponding contributions) are continuously updated and the protocol is initiated when the respective global gain becomes greater than zero. For the uncoordinated protocol, each peer continuously updates its individual gain and plays whenever this value becomes greater than zero. For updating the measures, trigger-based protocols require to monitor the system (workload and content) continuously and thus, introduce additional overheads.

4.3 Controlling Parameters

The gains that individual peers attain from relocation may not always worth the re-organization cost. To this end, we present three mechanisms for overhead control, which can be applied to all variations of both the uncoordinated and coordinated protocols, either individually or in combinations.

Stopping Condition. After applying its relocation policy, each peer compares its gain against a system-defined threshold ϵ. The peer determines that it needs to move only if its gain is larger than ϵ. Consequently, reformulation stops without the system reaching an equilibrium, but rather an ϵ-stable state. In the coordinated protocol, the stopping condition may also be applied on a global level, if we measure the gain with respect to the social cost or contribution.

Playing Probability. Instead of allowing a peer to play (i.e. re-evaluate its strategy) every time it is its turn, we introduce the use of a *playing probability* Pr, which determines how aggressive a player is, i.e., how high is the player's chance to play. The playing probability can either be the same for all peers, so as to treat all peers as equals or it may differ for each peer. For example, giving higher probability to peers that change their content or workload often allows them to adapt faster to these changes. Alternatively, a higher probability may be given to peers with more content or heavier query workload, since they are the ones that influence the workload cost and contribution the most.

Quota. Overhead can finally be controlled by enforcing a movement quota. Each peer is assigned a quota of n possible moves, which is the maximum number of moves it is allowed for a specified period T_q. After the end of T_q, the quota is replenished and the peer has again n available moves for the next T_q. T_q can either be a time interval or a number of events. Note, that using a time interval corresponds to treating all peers as equals, while using events to measure T_q may allow more demanding peers to play more often, as they are affected by more events.

The value of n expresses a trade-off between consuming system resources for re-clustering and tolerating low recall values from a poor clustering.

Algorithm 1 Coordinated Event-Based Protocol

$|P|$: number of peers, $C = \{c_1, \ldots, c_n\}$: cluster set,
$R = \{r_1, \ldots, r_n\}$: cluster representatives

```
1: for all global events do
2:   for all r_i ∈ R do
3:     send a game initialise request to all p_j ∈ c_i
4:     for all p_j ∈ c_i do
5:       evaluate gain_{p_j}
6:       send to r_i a relocation request with gain_{p_j} for c_new
7:     end for
8:     send all relocation requests to all other r_i ∈ R
9:     sort relocation requests in non-increasing order of gain
10:  end for
11:  for all gain_{p_j} within x% of the list do
12:    p_j moves from c_i to c_new
13:  end for
14: end for
```

5. EXPERIMENTAL EVALUATION

We model a system of peers sharing data belonging to different semantic categories. To capture locality, we use the model introduced in [18], which was derived from measurements in real traces of p2p systems. Each peer is associated with a data category j and maintains documents belonging to it. The local query workload of each peer is generated by first selecting a data category with probability $P(j)$ following a zipf distribution, and then a document d from that category with probability $P(d, j)$ following another zipf distribution within each category. We define $P_{x \in l}(d, j)$ as the probability of peer x associated with category l posing a query about document d of category j as:

$$P_{x \in l}(d, j') = \begin{cases} (1-m)P(d,j), l \neq j \\ ((1-m) + m/P(j))P(d,j), l = j \end{cases}$$

Parameter m is a measure of the interest-based locality peers exhibit. We consider three general scenarios. In the *symmetric scenario*, in which $m = 1$, both queries and data of each peer belong to the same category. In the *asymmetric*

scenario, again $m = 1$ but for a $j \neq l$ which is selected randomly from the remaining categories. That is, each peer has data from one category but poses queries for a single different category. Thus, the symmetric scenario exhibits maximum interest-based locality, while the asymmetric none. Finally, in the *random scenario* (no underlying clustering), $m = 0$. In this case, there is no interest-based locality and each peer has both data and queries uniformly distributed from all categories.

We use Newsgroup articles belonging to 10 different categories as our data set. The articles were pre-processed, stop words were removed, lemmatization was applied and the resulting words were sorted by frequency of appearance. The texts are distributed among 10000 peers. Peers inside each cluster are organized in a Chord-like topology (logarithmic θ). Table 3 summarizes our parameters. We present four sets of experiments evaluating our protocols.

Table 3: Input Parameters

Parameter	Range	Default Value
Topology and Strategy		
number of peers ($\|P\|$)	-	10000
parameter α	1-100	10
membership cost function (θ)	log, linear	log
strategy	-	self-alt-hybrid-mix
degree of selfishness d	0.25-0.75	-
Data-Query Distribution		
number of categories	-	10
interest locality degree (m)	-	0-1
Controlling Parameters		
stopping condition (ϵ)	0-10^{-8}	$10^{-4}, 10^{-3}, 10^{-6}$
playing probability (Pr)	0-1	0.5
movement quota (n)	1-15	∞
quota period in events (Tq)	-	20, (5*batch size)
% of granted requests (x)	10-100	50

5.1 Comparison with coordinated protocols

In the first set of experiments, we evaluate the coordinated and uncoordinated protocols and their variations. We compare uncoordinated and coordinated versions of the event-based, trigger-based and batch-based protocols with batches of 20, 50 and 100 events, which correspond to the 1/10th, 1/4th and 1/2nd of the average local query workload of a peer respectively. We consider asymmetric peers since they pose the greatest challenge when applying clustering. We assume selfish peers; we explore the influence of strategy selection later. We start with an unclustered overlay, in which each peer forms its own cluster. We measure the social cost when we reach a stable state, and the induced overhead in terms of required moves and turns.

Controlling Parameters. We consider the three controlling parameters: stopping condition (Fig. 2), playing probability (Fig. 3) and quota (Fig. 4(center-left)-(center-right)). In each experiment, we vary the value of one of the parameters and set the rest to their default values (Table 3). The stopping condition refers to the individual cost for the uncoordinated protocols and to the social cost for the coordinated ones. Similarly, instead of the playing probability, for the coordinated protocols, we evaluate the influence of the percentage of granted requests (parameter x in Alg. 1).

The value of the *stopping condition* ϵ is the main controlling factor of the achieved social cost (Fig. 2). For each protocol variation, we can determine an appropriate value for ϵ such that for values of ϵ lower than this, the improvement in the social cost does not justify the required overhead. Protocols that entail smaller overheads, such as the uncoordinated batch-based ones, allow for lower ϵ values (10^{-6}), while protocols with large overheads such as the coordinated trigger-based one require larger values to work efficiently (10^{-2}). We set ϵ to these appropriates values for each protocol for the rest of our experiments.

The *playing probability* does not influence the value of the achieved social cost but only the overhead required to reach it (Fig. 3). For all protocols, a smaller Pr reduces the number of moves, but increases the required turns, due to failed probability checks that do not allow the peers to play at each of their turns. In particular, the batch-based approaches with larger batch sizes are even more sensitive to $Pr < 0.5$ as their turns are increased without a corresponding reduction in the moves.

Using *quota* influences the protocols similarly to the playing probability (Fig. 4(center-left)-(center-right)). For protocols with large overheads, combining the use of quota with the playing probability can further improve performance. For example, for the uncoordinated trigger-based protocol, it reduces the required moves by 10% for the same Pr, without increasing the number of required turns as in the case of lowering the playing probability.

The coordinated protocols achieve approximately the same values for social cost with the uncoordinated ones, while imposing a much larger overhead, especially with regards to the number of required turns. Among the uncoordinated protocols, the trigger-based one has the greatest overhead, while the batch-based ones have the smallest.

An advantage of the controlling parameters is that they can be dynamically adjusted locally by each peer. By observing its processing and communication cost caused by its moves and the fluctuations in its utility function, a peer can achieve appropriate tuning. For example, a low individual cost may cause the peer to decrease its ϵ value, while a large number of moves that do not considerably improve its cost may cause the peer to increase ϵ. Similarly, if a peer observes too many moves, it may decrease its playing probability, or in the presence of frequent updates increase it to react faster to changes. Similar observations may be applied to tuning quota.

Progress through Turns. The values reported so far correspond to the final value of the social cost, when the system stabilizes. We also evaluate how the value of the social cost changes through progressing turns. The batch-based protocols are the ones that react the slowest to changes, while the coordinated trigger and event-based protocols are the ones which react the fastest (Table 4).

The basic reason is that the coordinated protocols favor the most cost influencing peers by granting their requests first, while the uncoordinated ones treat all peers as equals. If this policy is not enforced, then, the average social cost per turn for the coordinated protocols becomes up to 15% higher, especially when x is small (Fig. 4(center-right)).

For the uncoordinated protocols, we can achieve a similar effect with regards to the workload cost of the system by adjusting the playing probability of each peer according to its demand level or by using quota (Fig. 4(right)).

5.2 Cluster Formation

In this set of experiments, we start from a random peer configuration and examine whether the peer reformulation protocol leads to the desired cluster configuration. For the

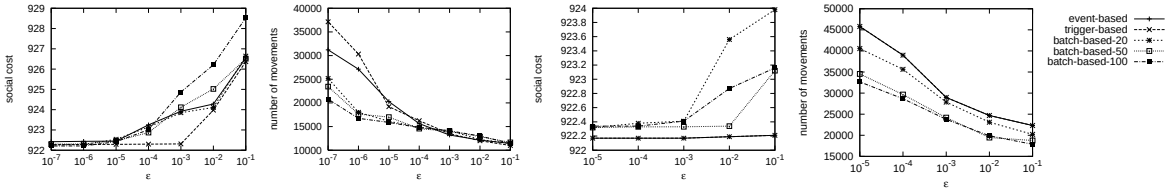

Figure 2: Varying ϵ (left), (center-left) with coordinated and (center-right), (right) uncoordinated protocol.

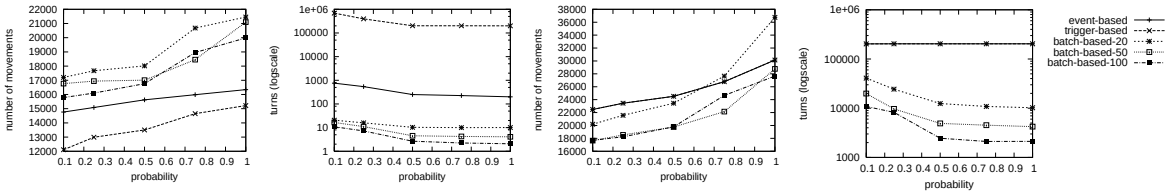

Figure 3: Varying Pr (left), (center-left) with coordinated and (center-right), (right) uncoordinated protocol.

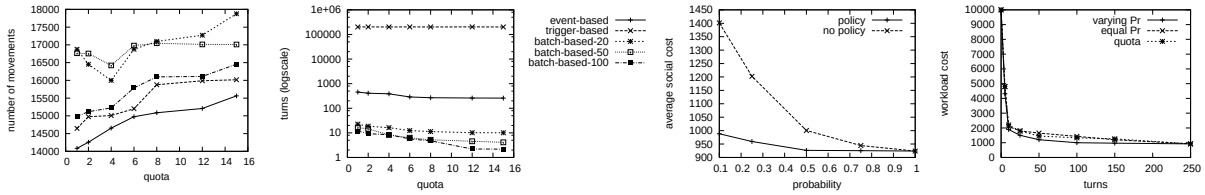

Figure 4: (left), (center-left) Varying quota, (center-right) effect of policy and (right) varying Pr.

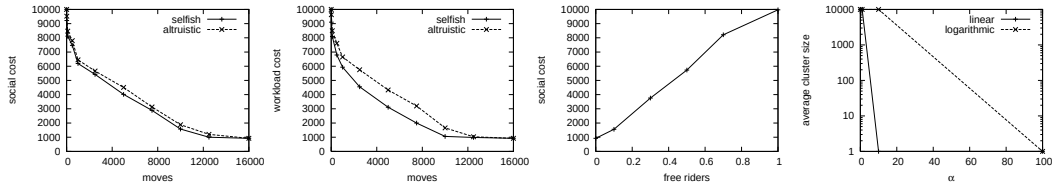

Figure 5: (left) Social and (center-left) workload cost with moves, effect of (center-right) free riders and (right) α.

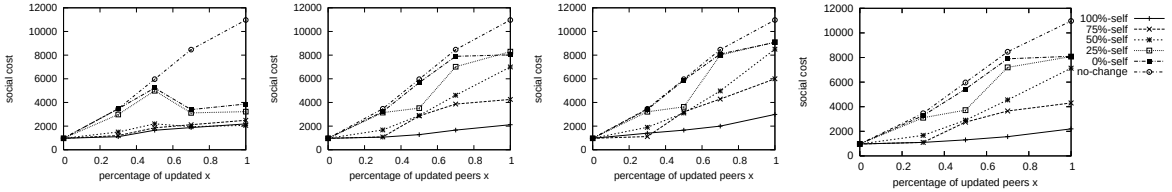

Figure 6: Update workload scenarios (left) $WSc1$, (center-left) $WSc2$, (center-right) $WSc3$ and (right) $WSc4$.

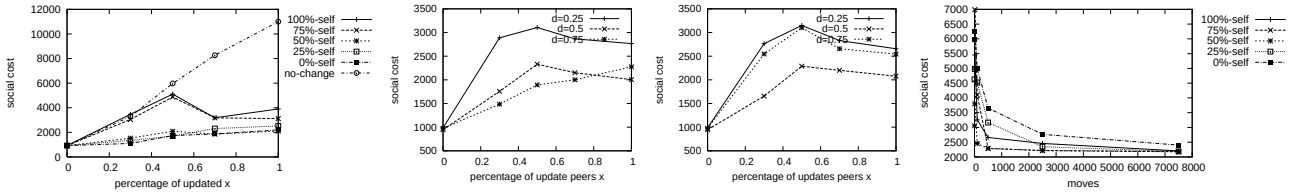

Figure 7: (left) Update content scenario $CSc1$, (center-left) $WSc1$ and (center-right) $CSc1$ with hybrid peers, and (right) updating both workload and content.

Table 4: Social Cost Per Round

Asynchronous					Coordinated				
Event	Trigger	Batch-20	Batch-50	Batch-100	Event	Trigger	Batch-20	Batch-50	Batch-100
989.32	945.56	1207.05	1876.15	2876.45	926.41	926.41	1100.25	1543.55	2454.67

Table 5: Cluster Formation

	Moves			Clusters			Cluster Size			SCost			SCont		
	Self	Alt	Mix	Self	Alt	Mix	Self	Alt	Mix	Self	Alt	Mix	Self	Alt	Mix
Symmetric Scenario															
i	15358	15436	14900	10	10	10	1087.5	1100	1100.5	10.07	10.23	10.15	9.67	9.85	9.84
ii	14786	14365	14657	10	10.5	10	1079	1109.5	1125.5	10.42	10.67	10.93	9.45	9.63	9.58
iii(a)	9654	9812	9867	10	10	10.5	1112	1085.5	1119.5	10.4	10.97	11.35	9.98	9.64	9.69
iii(b)	10211	10210	11270	10	10.5	10.5	1010.5	1056.25	1011.75	11.45	10.81	11.59	9.82	9.36	9.42
iii(c)	9841	9554	9456	10	10.5	10	1012.25	1009.5	1016.5	10.65	10.83	11.72	9.32	9.22	9.29
iv	0	0	0	10	10	10	1100	1100	1100	10.09	10.09	10.09	9.94	9.94	9.94
v	0	0	0	10	10	10	1100	1100	1100	10.09	10.09	10.09	9.94	9.94	9.94
Asymmetric Scenario															
i	16875	16758	17008	89.25	89.5	90.25	110.75	110.9	111.15	919.9	922.35	924.85	987.45	974.15	992.09
ii	16257	16109	16431	89.25	89.15	90.01	111.9	111.75	111.45	924.75	929.25	926.05	974.01	959.25	964.26
iii(a)	9405	9115	9650	90.1	90.2	90	111.05	110.5	110.33	922.86	926.41	941.02	947.05	961.67	961.08
iii(b)	9180	9320	9125	89.9	90.05	90.2	111.15	110.9	110.67	922.65	921.15	924.66	967.33	965.09	964.23
iii(c)	8502	8745	8790	89.75	89.5	90.1	110.5	110.24	111	923.08	924.44	926.12	978.02	967.15	976.45
iv	5865	6120	6275	90	90.15	90.15	111.4	111.75	111.05	929.05	926.10	924.57	966.14	967.86	968.1
v	6015	6104	6436	90	89.9	90	110.9	111.09	110.89	917.95	919.05	920.69	952.21	954.33	956
Random Scenario															
i	19450	19710	19340	1	1	1	10000	10000	10000	11.33	11.33	11.33	11.04	11.04	11.04
ii	0	0	0	1	1	1	10000	10000	10000	11.33	11.33	11.33	11.04	11.04	11.04

rest of the experiments, we use the uncoordinated event-based protocol with $Pr = 0.5$ and $\epsilon = 10^{-4}$. We define a *clique* as a set of peers that have workload and content belonging to the same semantic category. Let M be the number of cliques for each scenario, i.e., for the symmetric scenario $M = 10$. We consider five different cases for the initial system configuration: (i) each peer forms its own cluster; (ii) all peers form a single cluster; (iii) peers are randomly distributed to n groups and we discern for different values of n the subcases: (a) $n = M$, (b) $n < M$ and (c) $n > M$; (iv) peers are clustered according to their content and (v) peers are clustered according to their workload. We apply our protocols and check whether the system reaches an equilibrium and if so, what is the total number of moves, the number of clusters formed and the average size of these clusters, along with the achieved social cost and the absolute value of the social contribution (Table 5).

In all scenarios, all strategies reach an ϵ-Nash equilibrium and form the desired number of clusters regardless of the number of clusters in the initial configuration. Thus, our protocol does not require a predefined number of clusters, but dynamically determines the appropriate number and may also change it over time to cope with updates as the desired number may also change over time. Selfish and altruistic peers do not differ significantly and a mixed strategy usually requires more moves, but has the same social cost.

For symmetric peers both social and workload cost are the same and depend only on the membership cost; the cost for the recall is zero, since all results to the local query workload of each peer are located within its cluster (Table 5 lines 1-7). In fact, the social cost achieved is for the given $\alpha = 10$ almost equal to the social optimum (almost 10) (Case Study (II.C) when θ is logarithmic). The value is not exact because the peers and the clusters are not perfectly symmetrical. For asymmetric peers, the social cost is higher and the contribution lower than the ones observed for symmetric ones. Since queries are not uniformly distributed among peers, the social and the workload cost differ slightly. For example, in case (i), the workload cost is 914.77, 920.01 and 925.86 for selfish, altruistic and mixed populations respectively, only smaller by 5 in the best case.

When symmetric peers are clustered according to their content or workload, the appropriate clusters are already formed (Table 5 lines 6-7). For asymmetric peers, both configurations are not stable, i.e., the peers can improve their cost, though they require less moves to reach stability than the other configurations. Thus, relying solely on content or query workload is not enough to provide the appropriate clustering.

For the third scenario, we consider all peers forming a cluster by their own and all peers in one cluster (Table 5 lines 16- 17). This scenario is similar to Case Study I where no underlying clustering exists. Due to the small value of the membership cost (logarithmic) and α, the peers form a single cluster in both cases with a social cost around the optimum. The second case is already the desired configuration. For the same α, if we use a linear θ, peers split into smaller clusters. Similarly, a larger value of α forces the peers to form more clusters. For example, for $\alpha = 100$ both configurations would converge to the first case of single membered clusters. Consequently, by tuning α we can favor configurations with either a small number of larger clusters or a large number of smaller clusters (Figure 5(right)).

Social vs Workload Cost. As we showed for asymmetric peers, the achieved workload cost is different from the corresponding social cost. We consider how the two measures progress as the peers make more moves. By adjusting the playing probability Pr according to the peer's demand levels, the workload cost is reduced faster when we consider selfish peers (Fig. 5(center-left)). The social cost that considers all peers as equals decreases linearly (Fig. 5(left)) to the number of moves. For altruistic peers, the workload cost is not reduced much faster than the social one. To have similar results as for selfish peers we need to increase Pr for peers with large size.

Free Riders Free-riders, which appear often in p2p systems ([1]), are peers that use data offered by others, without contributing any content. We model a free-rider p as a selfish peer with $r(q, p) = 0$, for all q in Q. Increasing the number of free riders gradually degenerates the overlay to one in which each peer forms its own cluster (Fig. 5(center-right)).

5.3 Cluster Adaptation

This set of experiments, evaluates how well the reformulation protocol adapts to changes. We start from a "good" cluster configuration for given content and workload, and consider content, workload and topology updates (peers join/ leave). The initial configuration consists of clusters of symmetric peers. We use mixed populations of both selfish and altruistic peers with different ratios, i.e., from all selfish (100% $selfish$) to all altruistic (0% $selfish$).

We consider four general workload update scenarios (WSc) (Fig. 6). In $WSc1$, a data category becomes more popular; y% of the peers that are selected randomly from all the clusters change their query workload to this specific category. In

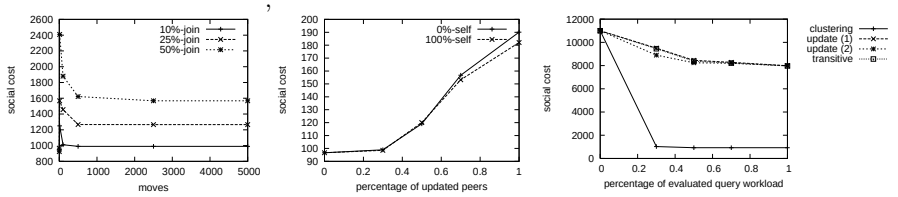

Figure 8: (left) Peers joining the system, (center) re-clustering and (right) clustering vs caching.

$WSc2$, a new data category that was not queried so far becomes popular. In $WSc3$, k existing categories become popular, and in $WSc4$, k categories seize to be popular, i.e., are not queried anymore. The four scenarios can also be applied as the corresponding content update scenarios (CSc).

For all the update scenarios, the reformulation protocol copes with the changes very efficiently. Compared to a static overlay in which no measures are taken to deal with the changes, our protocol reduces the social cost up to 1/3 of its value. In general, update scenarios in which the relocations request are more evenly distributed among the clusters ($WSc2$) perform slightly better, while scenarios that create more asymmetric peers ($WSc3$) behave worse.

Overall, selfish peers react faster to workload changes (Fig. 6), correcting system behavior faster, while altruistic ones exhibit the same behavior for content changes (Fig. 7 (left)). The reason is that a change in the workload of an altruistic peer does not instigate a reaction, and for the peer to become aware of the change in the others' workload that change has to affect a large number of peers.

Hybrid peers consider both selfish and altruistic criteria to determine their cluster membership. When the degree of selfishness d is small, the peers are affected from content changes more (Fig. 7(center-right)), while for d closer to 1, from workload ones (Fig. 7(center-left)). When both selfish and altruistic criteria are weighted equally ($d = 0.5$), hybrid peers react faster than selfish peers to content changes but slower than altruistic ones and vice versa for workload changes.

Workload and Content Change. Besides from updates that affect only the workload or the content of the peers, we also consider updates that affect both (Fig. 7(right)). In particular, changes in the workload of a peer, usually entail changes in its content, i.e., if a peer becomes interested in a new data category, then it will gradually acquire data of this category. At first, such an update increases the social cost as there are not enough data to satisfy the query workload of the updated peers. Gradually, as they change more of their data, the updated peers form a new cluster, thus, effectively reducing the social cost.

Topology Updates. We now consider updates in the topology, i.e., peers joining and leaving the system. When new peers join the system, they initially cause a considerable increase in the social cost. However, as the new peers pose queries and are gradually informed about the clusters in the system, they select an appropriate cluster effectively reducing the social cost (Fig. 8(left)). When peers leave the system, the social cost is actually reduced as the number of peers is reduced. No immediate adaptation is required but if large percentages of a cluster's members leave, then the other members may require to move to improve recall, or peers from other clusters may move to this cluster to exploit the low membership cost.

Cluster Reformulation vs Re-Clustering We compare the reformulation protocol to an alternative approach for coping with changes by reapplying the clustering procedure from scratch. Reclustering from scratch (Fig. 8(center)) reduces the social cost of the resulting configuration up to 10% compared to a configuration in which we applied our reformulation protocol to cope with the update (Fig. 6(center)). However, reclustering entails a much larger overhead, requiring about 250 turns, while the reformulation protocols only requires 10. Also, reclustering requires $> |P|$ moves, regardless of the number of updated peers, unlike reformulation that mostly affects the updated peers.

5.4 Comparison with a Caching Scheme

We compare our reformulation protocol with a caching scheme [18], in which peers that provided results to previous queries are cached and future queries are first forwarded to them. If the peers receiving a query also forward it to peers in their cache, we have the *transitive* variation. The cache contents are updated after each query. For the peers already in the cache, an aggregated recall value is updated according to the results that they provided for the latest query. The y peers, which provided the most results for a query but are not in the cache, are inserted in the cache replacing the ones with the lowest overall recall, if there is not enough space (*update* (y)).

We assume a maximum number of links for each peer equal to the number of links required for establishing a Chord-like topology within the clusters, i.e., $log(|c_i|)$. When caching is used, 3 of these links are used for the underlying network and the rest are allocated as cache entries. We consider asymmetric peers which are more appropriate for caching than symmetric ones that favor clustering. In caching, a peer p_i may belong to a peer p_j's cache, and provide results to p_j, without forcing p_j to belong to its cache. Whereas in clustering, for p_i to provide results to p_j, they must both belong to the same cluster. However, by deploying an efficient topology within the clusters (such as Chord-like), clustering is again able to outperform caching. Furthermore, clustering reacts to updates faster and more efficiently than caching (Fig 8(right)). While caching changes one to two neighbours, clustering changes all neighbours at once, thus, achieving lower social cost faster.

6. RELATED WORK

Game theoretic approaches have been applied to model the behavior of peers in p2p systems. In [7], the creation of an Internet-like network is modelled as a game with peers as uncoordinated selfish agents. The goal is for each peer to choose the peers to link to. 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 proper-

ties. [14] considers a more sophisticated model, in which strict bounds are enforced on the out-degree of the peers, links are directed and peers are allowed to express preferences regarding the choice of their neighbors. Our approach can be viewed as setting these preferences based on recall benefits. In [17], the authors show that allowing peers to act completely freely performs much worse than collaboration, and prove that even a static p2p system of selfish peers may never reach convergence. This result agrees with our findings that show that only in specific scenarios, we reach stability. In [22], altruistic peers determine the level of their contribution based on a utility function that depends on parameters such as the amount of data they upload and download, whereas in our altruistic policy, the choice of the cluster depends on the peer contribution to it.

Many recent research efforts have focused on organizing peers in clusters. In most cases, the focus is on cluster formation and query processing and the adaptation of the overlay to changing conditions is not addressed. In [9], a superpeer-based architecture is proposed in which peers with common interests are organized based on their caches. The paper exploits the idea of [18], and since it is based on caches it implicitly addresses the issue of cluster adaptation, but does not focus on it. Furthermore, as a cache-based scheme it is better suited for selfish symmetric peers, while our model can encompass more types of peers. In [3], peers are partitioned into topic segments based on their data. A fixed set of M clusters with centroids that are globally known is assumed, each one corresponding to a topic segment. Clusters of peers are formed in [21] based on the semantic categories of their data; the semantic categories are predefined. Similarly, [5] assumes predefined classification hierarchies based on which queries and data are categorized. Instead of predefined categories, [8] uses a learning approach that based on generalizing the shared data, learns the semantic categories they belong to and then uses those for clustering. Clustering in [15] 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 [11], is possible. In [6], clustering is first applied on the documents of each peer, and then recursively on the derived feature vectors by selected peer representatives. While this approach does not assume predefined categories, it still requires the use of cluster representatives unlike our uncoordinated protocol. In [4], peers maintain sets of guide rules, which are formed by the users either explicitly based on their interests, or implicitly through query history, thus defining semantic clusters. A somewhat different approach to clustering is taken in pSearch [20] that maps peer documents on a DHT, based on their term vectors and exploiting only the most important terms. Thus, semantically related documents are "clustered" in the DHT, limiting the search space.

A preliminary version of the model for selfish peers (Section 2) and of the relocation policies (Section 4.1) have been presented in [12].

7. CONCLUSIONS

In this paper, we model peers in a clustered overlay as players that dynamically change the set of clusters they belong to according to an individual utility function, based on a cluster membership cost and query recall. We model both selfish peers that aim at minimizing their individual cost, i.e., maximizing their recall, and altruistic peers that try to maximize their contribution to others. We define measures for evaluating global system quality and propose a reformulation protocol to cope with system maintenance. Our experimental results show that our protocol succeeds in gradually correcting system performance.

8. REFERENCES

[1] E. Adar and B. A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), 2000.

[2] N. Bansal, F. Chiang, N. Koudas, and F. W. Tompa. Seeking stable clusters in the blogosphere. In *VLDB*, 2007.

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

[4] E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. In *INFOCOM*, 2003.

[5] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems, Technical Report, Computer Science Department, Stanford University, 2002.

[6] C. Doulkeridis, K. Norvag, and M. Vazirgiannis. Desent: decentralized and distributed semantic overlay generation in p2p networks. *JSAC*, 25(1):25–34, 2007.

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

[8] A. Fast, D. Jensen, and B. N. Levine. Creating social networks to improve peer-to-peer networking. In *KDD*, 2005.

[9] P. Garbacki, D. H. J. Epema, and M. van Steen. Optimizing peer relationships in a super-peer network. In *ICDCS*, 2007.

[10] S. B. Handurukande, A.-M. Kermarrec, F. L. Fessant, L. Massouli, and S. Patarin. Peer sharing behaviour in the eDonkey network, and implications for the design of server-less file sharing systems. In *EuroSys*, 2006.

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

[12] G. Koloniari and E. Pitoura. Recall-based cluster reformulation by selfish peers. In *NetDB*, 2008.

[13] G. Koloniari and E. Pitoura. A recall-based cluster formation game in peer-to-peer systems (extended version), Technical Report TR 3-2009, Computer Science Department, University of Ioannina, 2009.

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

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

[16] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, 2007.

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

[18] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, 2003.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, 2001.

[20] C. Tang and Z. Xu and M. Mahalingam. psearch: information retrieval in structured overlays. *Computer Communication Review*, 33(1):89–94, 2003.

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

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