A Game Theoretic Approach to the Formation of Clustered Overlay Networks

Georgia Koloniari and Evaggelia Pitoura

Abstract—In many large-scale content sharing applications, participants or nodes are connected with each other based on their content or interests, thus forming clusters. In this paper, we model the formation of such clustered overlays as a strategic game, where nodes determine their cluster membership with the goal of improving the recall of their queries. We study the evolution of such overlays both theoretically and experimentally in terms of stability, optimality, load balance and the required overhead. We show that, in general, decisions made independently by each node using only local information lead to overall cost-effective cluster configurations that are also dynamically adaptable to system updates such as churn and query or content changes.

Index Terms—Clustering, Distributed Systems, Data Sharing

1 Introduction

Recently, there has been an explosion in the use of content sharing applications such as those involving social networking and peer-to-peer (p2p) file sharing. Measurements from the deployment of such large-scale systems have shown that the interactions among their participants or nodes indicate the existence of groups or *clusters* of nodes having similar content or interests. For example, in measurements of popular on-line social networks [14], it was 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. The formation of implicit groups centered around topics described by common keywords has also been observed in the blogosphere [2]. Furthermore, the peer selection algorithm used in BitTorrent was shown to lead to the formation of clusters of peers having similar interests and upload capacities [13].

In this paper, we focus on large scale distributed systems where nodes form clusters by creating logical links to other nodes that share similar content or interests, thus, creating a *clustered overlay network* on top of the physical one. The main reason for the formation of such clusters is that clustered overlays enable their participants to find and exchange data relevant to their queries with less effort. For example, traces of popular p2p systems have indicated that nodes exhibit the property of interest-based locality, that is, if a node holds content satisfying some query of another node, then it most likely also maintains additional content of interest to this other node [8], [15]. Thus, placing such nodes in the same cluster would increase the recall of their queries. Although, there has been a large body of research on the discovery and construction of clustered overlays, e.g., [3], [4], [18], their dynamic maintenance has been mostly ignored.

• *G. Koloniari and E. Pitoura are with the Department of Computer Science, University of Ioannina, Greece.*
E-mail: {kgeorgia,pitoura}@cs.uoi.gr

We address the dynamic creation and adaptation of clustered overlays by taking a game theoretic approach. We model the problem of cluster formation as a strategic game with nodes as the players. Each node plays by selecting which clusters to join. This selection or strategy is determined individually by each node so as to minimize a utility function that depends on the cluster membership cost and on the cost of evaluating queries outside of the clusters the node belongs to. Game-theoretic models have been previously proposed for creating overlays based on the connection cost or the radius of the network graph, e.g., [6], [17]. The originality of our approach lies in that we model clustered overlays and in that we aim at increasing the recall of queries.

We present an uncoordinated cluster formulation protocol that relies on local decisions made independently by each node based only on its partial view of the system. We also study both theoretically and experimentally the formation and evolution of clusters under the individual actions of each node. Our theoretical results provide conditions under which the game reaches a stable state as well as the associated cost and the achieved load balance. Our experimental results show that the uncoordinated protocol leads to cost-effective cluster configurations adaptable to system changes including content, query and topology (nodes join/leave) updates. Moreover, our experiments show that using coordination and global decisions does not improve the quality of the attained configurations considerably.

The rest of this paper is organized as follows. In Section 2, we introduce the cluster formation game, while in Section 3, we study the stability, optimality and load balance of specific configurations. In Section 4, we present distributed protocols that implement the game. Section 5 reports experimental results and Section 6 related research. Section 7 offers conclusions.

2 A Cluster Formation Game

We consider dynamic large-scale content sharing distributed systems. In such systems, it is not possible for a node to know and directly communicate with all other nodes in the system. Instead, each node establishes logical links with only a few other nodes. These logical

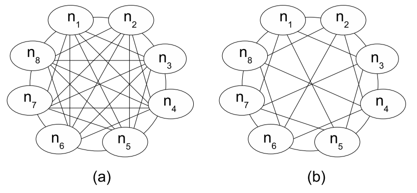


Fig. 1: Examples of cluster topologies

links create a logical *overlay network* on top of the physical one (e.g., the Internet). In this paper, we consider *clustered overlays* where nodes with similar content or interests form groups, called *clusters*. The nodes inside each cluster are highly connected with each other to achieve efficient intra-cluster communication (Fig. 1).

Let V be the current set of nodes. We model the problem of cluster overlay formation as a strategic game, where each node is a player whose strategy is defined by the set of clusters it joins. In particular, each node $n_i \in V$ chooses which clusters to join from the set C of clusters in the system, thus, defining its strategy $s_i \subseteq C$. We constrain the maximum number of clusters in C to $|V|$, i.e., it cannot exceed the number of nodes. Note that some of the clusters may be empty.

From the set of strategies $S = \{s_1, s_2, \ldots, s_{|V|}\}$ that the nodes in V deploy, i.e., the *strategy profile*, we can derive the set of nodes belonging to each cluster in C. The set S is also called *(cluster) configuration*. For example, consider $V = \{n_1, n_2, n_3, n_4\}$ and $C = \{c_1, c_2, c_3, c_4\}$, and let the corresponding strategies be $s_1 = \{c_1, c_2\}$, $s_2 = \{c_1\}$, $s_3 = \{c_3\}$ and $s_4 = \{c_2, c_3\}$ meaning that n_1 belongs to clusters c_1 and c_2, n_2 belongs to c_1, n_3 to c_3 and n_4 to c_2 and c_3. Each node decides which clusters to join so as to minimize a *utility* function that expresses its *individual cost*. A node plays more than once to cope with system updates, thus, the cluster configuration is dynamic.

Cost Function. Each node both stores content and poses queries for content. We call *global workload*, and denote with Q, the multi-set of all queries in the system. Note that the same query may appear in Q more than once. Similarly, for each node n_i, we denote with $Q(n_i)$ its *local workload* defined as the multi-set of all queries issued by n_i. For a local or global workload Q' and a query q, $num(Q')$ stands for the size of Q', i.e., the number of (not necessarily distinct) queries in Q' and $num(q, Q')$ for the number of times q appears in Q'.

We characterize the importance of a node n_i in the evaluation of a query q based on the number of results that n_i offers for q with regards to the total number of available results for q, i.e., the recall achieved when q is evaluated solely on n_i. We assume that each result is distinct. Specifically, let $result(q, n_i)$ be the number of results for a query q provided by node n_i, we define:

$$r(q, n_i) = \frac{result(q, n_i)}{\sum_{n_k \in V} result(q, n_k)}$$

Let $V(s_i)$ be the set of all nodes belonging to clusters in s_i. Our premise is that intra-cluster query evaluation is very efficient. Thus, the benefit for a node n_i from choosing strategy s_i is the recall attained by evaluating the queries in $Q(n_i)$ against the nodes in $V(s_i)$. Stated differently, the recall cost for n_i associated with strategy s_i is the cost for obtaining query results from nodes located in clusters that do not belong to s_i, that is, from nodes not in $V(s_i)$.

The recall cost is minimized, if a node joins all clusters. However, cluster membership imposes communication and processing costs. The cluster membership cost depends 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 it. Furthermore, a highly connected topology, where each node maintains links to a large number of other nodes, also increases the cluster membership cost. To capture this, the cluster membership cost is defined as a monotonically increasing function θ of the number of nodes belonging to the cluster, i.e. as a function of the cluster size $|c|$. This function depends on the cluster topology, for instance, for a mesh topology (Fig. 1(a)), θ may be linear, whereas for structured overlays (Fig. 1(b)), θ may be logarithmic.

Definition 1 (Individual Node Cost): In a cluster configuration S, the individual cost for a node n_i for choosing strategy s_i is:

$$icost(n_i, S) = \alpha \sum_{c_k \in s_i} \frac{\theta(|c_k|)}{|V|} + \sum_{q \text{ in } Q(n_i)} \frac{num(q, Q(n_i))}{num(Q(n_i))} \sum_{n_j \notin V(s_i)} r(q, n_j)$$

The first term models the cluster membership cost, while the second one models the recall cost for obtaining results from nodes outside the selected clusters, that is, the average result loss from not participating in all clusters. The recall cost for each query is weighted by its frequency in the local workload of n_i. The factor $1/|V|$ is used for normalizing the cluster membership cost. Parameter α ($\alpha \geq 0$) determines the influence of the cluster membership cost in cluster formation.

From a system perspective, α characterizes the ratio between updates and queries. For a given θ, a large value of α means that updates are rather frequent and therefore the cost for cluster maintenance is high, while a small value indicates that query evaluation is more important in the overall system performance. Observe that the two terms of the cost function tend to guide the nodes towards selecting opposite strategies. For example, in a cluster configuration where all nodes form a single cluster, the membership cost is maximized, while the recall loss is minimized, since for each node all results for its queries are located within its cluster. In contrast, the recall loss is maximized when each node forms a cluster of its own, while, in this case, the membership cost is minimized.

The local workload and the query loss for a node n_i do not need to be fixed or known a priori for the whole game. Instead, each node plays, i.e., re-evaluates its individual cost, multiple times, to cope with changes of its local query workload and of the content offered by the other nodes.

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

$$SCost(S) = \sum_{n_i \in V} icost(n_i, S)$$

Stability and Optimality. The goal of each player (node) 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)? That is, will the system reach a Nash equilibrium?

Formally, a (pure) Nash equilibrium is a configuration S such that, for each node n_i with strategy $s_i \in S$, and for all alternative configurations S' which differ only in the i-th component (different cluster sets s'_i for n_i):

$$icost(n_i, S) \leq icost(n_i, S') \quad (1)$$

Let us first present a couple of properties of stable configurations (see [9] for proofs).

Lemma 1: In any stable cluster configuration, there are no clusters c_i, c_j such that $c_i \subseteq c_j$, $i \neq j$.

From Lemma 1, it holds that:

Corollary 1: A cluster configuration where a node forms both a cluster of its own and also belongs to another cluster is not stable.

The following lemma expresses the fact that in a stable configuration, there is no incentive for a node to join a cluster whose other members offer no results to its queries, since participation in such a cluster would only increase its cost.

Lemma 2: For any strictly increasing function θ, there is no stable configuration in which $\exists c_i$, $|c_i| > 1$, $c_i \in s_j$ and $\sum_{q \text{ in } Q(n_j)} \sum_{n_k \in c_i} r(q, n_k) = 0$.
It can be also shown that:

Proposition 1: A pure Nash equilibrium does not always exist for the cluster formation game.

The *optimal social cost* is the minimum social cost over all possible configurations. In general, the cost of a stable configuration is not necessarily optimal.

Load Balance. First, we define *size-based* load balance.

Definition 2 (Size-based Balanced Configuration): A configuration S is (δ_s)-size balanced if and only if $\forall c_i, c_j \in C$, it holds that: $(|c_i| - |c_j|)/|V| \leq \pm\delta_s$,
where $0 \leq \delta_s \leq 1$. For small δ_s, such clusters have small differences in the number of nodes that belong to them and thus, small differences in their membership cost.

We also consider the distribution of the system load among the clusters of the system. The system load includes both the query workload and content. Specifically, for each cluster c_i, we define the *cluster query load*, denoted $L_q(c_i)$, as the percentage of queries generated from its nodes over all system queries:

$$L_q(c_i) = \sum_{n_k \in c_i} num(Q(n_k))/num(Q)$$

Similarly, we define, for each cluster c_i, the *cluster content load*, denoted $L_r(c_i)$, as the percentage of content offered by its nodes over all available content:

$$L_r(c_i) = \sum_{n_k \in c_i} \sum_{q \text{ in } Q} r(q, n_k) / \sum_{n_k \in V} \sum_{q \text{ in } Q} r(q, n_k)$$

Definition 3 (Content and Query Balanced Configuration): A configuration S is (δ_q, δ_r)-load balanced, if and only if, $\forall$ c_i, c_j $\in$ C, it holds that:

$$L_q(c_i) - L_q(c_j) \leq \pm\delta_q$$
$$L_r(c_i) - L_r(c_j) \leq \pm\delta_r$$

where $0 \leq \delta_q, \delta_r \leq 1$.

Observation 1: If we assume a uniform distribution of the global query workload and content among all nodes, then a (δ)-size balanced configuration is also (δ, δ)-load balanced.

3 Case Studies

In this section, we focus on a number of characteristic special cases of content sharing and study for each one of them, the properties of a number of cluster configurations. The formal definition of each case as well as the derivation of our results can be found in [9].

CASE I: NO UNDERLYING CLUSTERING. In this case, there is no content or query similarity among the nodes.
CASE II: SYMMETRIC SCENARIO. This case corresponds to the most favorable scenario for clustering. There are g ($g \geq 1$) content categories and the nodes belong to g disjoint groups of the same size ($|V|/g$) such that the members of each group offer and request content from the same category.
CASE III: ASYMMETRIC SCENARIO. The shared content again belongs to t ($t > 1$) different categories, however, each node has content that belongs to one category but poses queries for content that belongs to a single different category.

The selected cases represent extremes between which the usual behavior in a content sharing system lies. In the first case, the content and the queries of the users are too diverse to favor clustering. Also, while a perfect symmetric scenario may not occur, in practice, most nodes have only one or a few prominent interests and most of their content and queries are focused around them. Thus, we expect clustering to occur based on these few common interests. Similarly, while having completely disjoint content and queries is uncommon, this may occur when the interests and queries of a user change and the content it maintains is no longer of interest, while new content has not been attained yet.

We assume that both the content and the queries are distributed among the system nodes and among the nodes in each category (when applicable) uniformly. For each of these three cases, we consider a number of cluster configurations and study them in terms of stability, optimality and load balance.

In particular, we consider: a configuration (A) where there is just a single cluster and all nodes belong to it, a configuration (B) where each node forms a cluster of its own, and a configuration (C) where the nodes form m, $1 < m < |V|$, equal-sized clusters, where for case II, $m = g$ and each cluster contains nodes from the same group, and for Case III, $m = t(t-1)/2$ and half of the nodes in each cluster maintain content from a category t_i and pose queries for another category t_j and the other half the other way around. Table 1 summarizes the results of our analysis.

For stability, we ensure that the cost of each node in the given configuration is smaller than in any other feasible configuration (see Ineq. (1)) using Lemmas 1 and 2 to reduce the number of configurations to be considered. Various observations can be made from Table 1. For example, we see that even for CASE I, where there is no similarity among the nodes, depending on the membership versus the recall cost, some cluster configurations are stable. As an example, for a θ function corresponding to a linear function of the form: $\theta(n) = \lambda n$, $0 < \lambda \leq 1$, configuration (A) where all nodes form a single cluster is stable for $\alpha \leq 1/\lambda$. Recall that large values of α mean that the membership cost is weighted more than the recall cost. Thus, for the same θ, for values of α larger than this threshold, the membership cost would surpass the

TABLE 1: Conditions for stability, $\surd$ indicates optimality for θ as in Lemma 3, k stands for the maximum number of clusters a node may belong to

	CONFIG. A	CONFIG. B	CONFIG. C
CASE I: NO UNDERLYING CLUSTERING			
Stability	$\alpha \leq \frac{\lvert V\rvert-1}{\theta(\lvert V\rvert)-\theta(1)}$	$\alpha \geq \frac{k}{k\theta(2)-\theta(1)}$	$\alpha \leq \frac{\lvert V\rvert/m-1}{\theta(\lvert V\rvert/m)-\theta(1)}$, $\alpha \geq \frac{\lvert V\rvert}{m\theta(\lvert V\rvert/m+1)}$, $\alpha \geq \frac{\lvert V\rvert/m(k-1)+1}{k\theta(\lvert V\rvert/m+1)-\theta(\lvert V\rvert/m)}$
Optimality	$\surd$	$\surd$	$\surd$
$\delta_s - \delta_q, \delta_r$	$(0), (0,0)$	$(0), (0,0)$	$(0), (0,0)$
CASE II: SYMMETRIC SCENARIO			
Stability	$\alpha \leq \frac{\lvert V\rvert-g}{\theta(\lvert V\rvert)-\theta(1)}$	$\alpha \geq \frac{kg}{k\theta(2)-\theta(1)}$	$\alpha \leq \frac{\lvert V\rvert-g}{\theta(\lvert V\rvert/g)-\theta(1)}$
Optimality		$\surd$	$\surd$
$\delta_s - \delta_q, \delta_r$	$(0), (0,0)$	$(0), (0,0)$	$(0), -$
CASE III: ASYMMETRIC SCENARIO			
Stability	$\alpha \leq \frac{\lvert V\rvert}{\theta(\lvert V\rvert)-\theta(1)}$	$\alpha \geq \frac{kt}{k\theta(2)-\theta(1)}$	$\alpha \leq \frac{\lvert V\rvert}{(t-1)(\theta(\frac{2\lvert V\rvert}{t(t-1)})-\theta(1))}$, $\alpha \geq \frac{\lvert V\rvert}{(t-1)\theta(\frac{2\lvert V\rvert}{t(t-1)}+1)}$, $\alpha \geq \frac{(k-1)\lvert V\rvert}{(t-1)(k\theta(\frac{2\lvert V\rvert}{t(t-1)}+1)-\theta(\frac{2\lvert V\rvert}{t(t-1)})}$
Optimality		$\surd$	
$\delta_s - \delta_q, \delta_r$	$(0), (0,0)$	$(0), (0,0)$	$(0), -$

recall benefits 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 by function θ. For instance, when λ is small (less connected topology), a single cluster remains stable for larger values of α. As another example, comparing Case I with $k = 1$ and Case II (perfect underlying clustering), we see that in Case II, configuration (B) where each node forms a cluster of its own is stable for larger values of α. For instance, for a linear θ, it is stable for $\alpha \geq g/\lambda$ as opposed to $\alpha \geq 1/\lambda$. Note also that in configuration (C), small values of g (which correspond to a smaller number of larger groups) require smaller values of α for being stable.

We also study whether the social cost of the stable configurations is optimal. To this end, we acquire a bound on the optimal social cost by considering the best individual cost for each node in any possible configuration using the following lemma.

Lemma 3: Let θ be a function such that for any x, $x_j \in N$, $1 \leq j \leq k$, if $x < \sum_{j=1}^{k} x_j \Rightarrow \theta(x) < \sum_{j=1}^{k} \theta(x_j)$, then for any node n_i, a configuration S with minimum $icost(n_i, S)$ is such that n_i belongs to just one cluster.

As indicated in Table 1, in some cases, the stable configurations have cost equal to this estimation. In terms of load balance, by construction all three configurations are (0)-size balanced. Case I is also (0, 0)-load balanced, while II and III are (0, 0)-load balanced only if the total amount of query workload and content of each of the g and t categories is the same. We have also studied a couple of configurations with non-equal sized clusters [9].

4 CLUSTER FORMATION PROTOCOLS

In this section, we describe how the game is played by the individual nodes in a distributed system. By playing the game, the nodes reconstruct the overlay network to form clusters to improve their recall. The game can be deployed both to bootstrap a non-clustered overlay network and to reformulate an already clustered one. Cluster reformulation is necessary when the recall achieved in the current cluster configuration deteriorates. Changes that affect the quality of clustering include topology updates as nodes enter and leave the system, as well as updates of the content or queries of each node.

Uncoordinated Protocol. Let S_{cur} and C_{cur} be the current cluster configuration and set of (non-empty) clusters respectively. Each node n_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 n_i follows. Then, n_i chooses the strategy s_{new} for which the corresponding cluster configuration S_{new} is such that:

$$S_{new} = \arg\min_{S_j} icost(n_i, S_j)$$

To measure how much a node benefits from changing its strategy, we use *gain* defined as:

$$gain(n_i) = icost(n_i, S_{cur}) - icost(n_i, S_{new})$$

If $gain(n_i) > 0$, n_i selects the new strategy. Each node plays, i.e., re-examines its strategy selection, repeatedly to cope with the system dynamics. In our *(basic) uncoordinated protocol*, each node n_i autonomously decides to play, i.e., re-evaluates its gain, after the evaluation of each of its local queries in $Q(n_i)$. Besides local queries, other non-local events (such as other nodes joining or leaving a cluster) may affect the gain of a node. For completeness, we also consider a variation, called *uncoordinated protocol with monitoring*, where a node re-evaluates its gain after any (local and non-local) event. Note that this protocol makes the unrealistic assumption that a node monitors the system continuously to detect potential updates that may affect its gain. We have also considered a *coordinated protocol* where cluster updates are ordered based on the achieved gain, but this was shown not to improve the social cost [9].

The individual cost, *icost*, of each node depends on the recall of its queries and its cluster membership cost. Both quantities are estimated. To this end, we assume that each cluster has a unique identifier, *cid*, known by all its nodes, which is assigned based on node IPs and timestamps. For example, when the first node joins a cluster, its *cid* is formed by the IP of the node concatenated with a timestamp. When other nodes join the cluster, they are informed of its *cid*. When all nodes leave a cluster, its *cid* just becomes unused. Recycling *cids* is beyond the scope of this paper. Query results are annotated with the corresponding *cid*s of the clusters that provide them. A node does not need to know all system *cid*s, but gradually learns them, as its queries acquire results annotated with new *cid*s. Based on the annotated query results, each node can monitor its recall with respect to the other clusters in the system and use it to evaluate its individual cost for the different configurations it needs to consider when it plays.

Coordinated Protocol. For comparison, we also consider a *coordinated protocol* where one node r_i at each cluster c_i serves as a cluster representative. Representatives achieve coordination by gathering and exchanging information with the nodes in their cluster. In particular, unlike the basic uncoordinated protocol, the coordinated protocol is triggered by each global event. The representatives inform the nodes in their cluster who in turn re-evaluate their strategy similarly to the uncoordinated protocol. Then, nodes send an update request for the new strategy they want to select along with their corresponding gain to one of the representatives of the clusters they belong to. Finally, the representatives exchange these requests, order them by non-increasing value of gain and the overall top-K percentage of them is granted. The algorithm is presented in detail in [9].

Partial Knowledge. To play the game, a node evaluates the cost of various strategies and chooses the best one among them. However, in practice, a node n_i may know only a subset $C(n_i) \subseteq C_{cur}$ of the current clusters. Note that if $C(n_i) \subset C_{cur}$, then a node may need to occasionally refresh the clusters in $C(n_i)$. For instance, if a node is aware only of its own cluster, it may become isolated. Moreover, since the cost of contacting all clusters may be large, a node n_i may choose to contact only a subset of the clusters known to it. To model this, we also consider for n_i the sets, $RC(n_i, q) \subseteq C(n_i)$ such that n_i routes each query $q \in Q(n_i)$ to all clusters in $RC(n_i, q)$. We assume that each node n_i uses these $RC(n_i, q)$ sets to estimate its membership and recall costs. We use the term *full set routing* for the case in which, a node n_i evaluates all of its queries using all of its known clusters, i.e., $RC(n_i, q) = C(n_i)$, $\forall q \in Q(n_i)$. All other protocols that evaluate queries using subsets of the known clusters are called *subset routing* protocols.

Instead of considering all possible configurations when it re-evaluates its strategy, a node only considers the clusters it is aware of and has received annotated results from. Thus, the selected strategy may not be the best among all possible ones. Moreover, while in full set routing, each node n_i evaluates the actual value of its recall cost for each cluster $c_j \in C(n_i)$, in subset routing, this value is only an estimation based on the subset of the queries in the local query workload of n_i that are forwarded to c_j. On a positive note, due to Lemma 2, lack of knowledge of clusters that do not provide any results to a node does not affect the correctness of the best strategy selection. We further study how partial knowledge affects the outcome of the game in our experiments.

Overhead Control. The gains that individual nodes attain from strategy changes may not always worth the re-organization costs. To this end, we present parameters for overhead control that can be used either alone or in combinations. The values of the various parameters can either be the same for all nodes or be set individually by each node. They express a trade-off between consuming system resources for re-clustering and tolerating low recall values from a poor clustering.

Stopping Condition. To restrict the number of cluster updates, we consider a threshold value or *stopping condition* ϵ, such that each node decides to update its strategy if its gain is larger than ϵ.

Update Probability. Instead of changing its strategy each time there is a positive gain, a node changes its strategy with an *update probability* P_u. The update probability determines how aggressive a player is.

Batches of Events. A node re-evaluates its gain not after each single event, but after a number b (batch) of events. The events are either local queries, in the case of the basic uncoordinated protocol, or all events, in the case of monitoring.

Quota. Each node is assigned a quota of ch changes, which is the maximum number of cluster changes it is allowed for a specified period T_q. After the end of T_q, the quota is replenished and the node has again ch available changes for the next T_q. T_q can be either specified as a time interval or as a number of events. Using events to measure T_q allows more demanding nodes to play more often, since their quota will be replenished more often.

5 Experimental Evaluation

We consider a system of nodes sharing documents (content) belonging to different semantic categories. To capture locality, we use the model of [7] derived from measurements from real traces of content sharing systems. The model uses a parameter L as a measure of interest-based locality [15] that we set accordingly to match our three case studies (see [9] for details).

With respect to the degree of local knowledge we study three cases. The first one assumes full knowledge (FK): for each node n_i, $RC(n_i, q) = C(n_i) = C_{cur}$, $\forall q \in Q(n_i)$, that is, each node knows all current clusters and each query is routed to all of them. In the second case, for each node n_i, the set $C(n_i)$ corresponds to 50% of C_{cur}, n_i uses a full set routing approach (FSR), $RC(n_i, q) = C(n_i)$, $\forall q \in Q(n_i)$, and a refresh period of 10 events. Finally, we consider a subset routing approach (SSR), where for each node n_i, $C(n_i) = C_{cur}$, and $\forall q$, in $Q(n_i)$, $RC(n_i, q)$ corresponds to a set of clusters randomly selected from 50% of C_{cur}. Note that, in both the FSR and SSR cases, the same number of clusters are considered. Additional cases can be found in [9]. We report the actual social cost of each configuration, not the one resulting by the estimations of the nodes.

We use Newsgroup articles belonging to 10 different categories as our data set. The articles were preprocessed, stop words were removed, lemmatization was applied and the resulting words were sorted by frequency of appearance. The articles are distributed among 10,000 nodes. Nodes inside each cluster are organized in a Chord-like topology (logarithmic θ) [16]. We report the average value of 100 runs of each experiment; confidence intervals for mean values of level 95% are included in [9]. We present results for the case where each node belongs to a single cluster, that is, for each node n_i, $s_i = \{c_l\}$, for some $c_l \in C_{cur}$. The other possible strategies for n_i are either moving to a cluster c_v, $c_v \neq c_l$ for $c_v \in C_{cur}$ or if $c_l \neq \{n_i\}$, forming a cluster of its own.

Sensitivity Analysis. We start with a sensitivity analysis of the uncoordinated protocol with respect to its tuning parameters. We consider both the basic uncoordinated protocol (*unc*) and the uncoordinated protocol with monitoring (*mon*), which makes the unrealistic assumption of knowledge of all global events. In each experiment, we vary the value of one of the tuning parameters and set the rest to their default values (Table 2a). We present

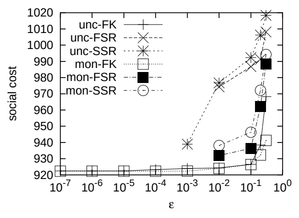
TABLE 2: (a) Input parameters and (b) results regarding cluster formation

(a)

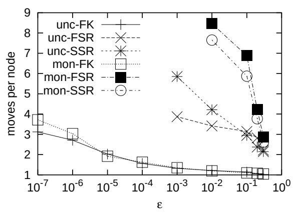
Parameter	Range	Default
Topology and Knowledge Degree		
number of nodes ($\|V\|$)	-	10000
parameter α	1-100	10
membership cost function (θ)	log, linear	log
knowledge degree	20%-100%	FK,50%-FSR-SSR
refresh period in events (Ref)	1-100	10
Content-Query Distribution		
number of categories	-	10
interest locality degree (L)	-	0-1
Tuning Parameters		
stopping condition (ϵ)	0.3-10^{-8}	10^{-2},10^{-3}
update probability (P_u)	0-1	0.7
batch size in events (b)	1-100	1
quota (ch)	1-15	∞
quota period in events (T_q)	20	-
granted requests pecentage (K)	-	70%

(b)

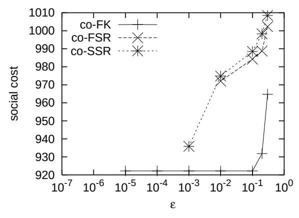
	Moves per Node			Number of Clusters			δ_s			SCost		
	FK	SSR	FSR	FK	SSR	FSR	FK	SSR	FSR	FK	SSR	FSR
Symmetric Scenario												
i	1.49	2.48	2.92	10	10.42	10.76	0.102	0.171	0.108	10.05	10.21	10.28
ii	1.46	2.07	2.75	10	10.27	10.59	0.143	0.176	0.110	10.24	10.12	10.32
iii(a)	0.91	1.89	2.47	10	10.61	10.82	0.078	0.102	0.112	10.52	11.24	11.43
iii(b)	1.05	2.42	2.92	10	11.08	10.92	0.102	0.023	0.015	11.25	11.78	12.01
iii(c)	0.82	1.81	2.61	10	11.21	10.92	0.115	0.108	0.120	10.48	11.25	11.52
iv	0	0	0	10	10	10	0	0	0	10.09	10.09	10.09
v	0	0	0	10	10	10	0	0	0	10.09	10.09	10.09
Asymmetric Scenario												
i	1.41	2.54	3.58	44.65	44.19	46.55	0.015	0.129	0.167	922.01	995.26	989.35
ii	1.52	2.38	3.27	45.45	44.86	46.76	0.035	0.112	0.109	926.15	1022.82	1027.14
iii(a)	0.88	1.87	3.05	45.25	46.12	46.75	0.078	0.109	0.120	924.21	1014.15	1016.05
iii(b)	0.78	1.79	2.92	44.45	43.27	45.23	0.109	0.092	0.097	924.15	1018.52	1027.41
iii(c)	0.71	2.02	2.85	44.71	44.78	45.27	0.092	0.150	0.087	924.24	1031.45	1041.15
iv	0.54	1.98	2.89	45.17	44.89	44.55	0.090	0.122	0.141	929.25	1056.45	1059.22
v	0.57	1.76	2.07	45.20	45.75	44.96	0.098	0.107	0.156	918.67	1032.65	1042.75
Random Scenario (No Underlying Clustering)												
i	1.74	3.45	4.02	1	1	1	0	0	0	11.33	11.33	11.33
ii	0	0	0	1	1	1	0	0	0	11.33	11.33	11.33



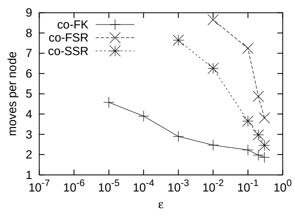
(a)



(b)

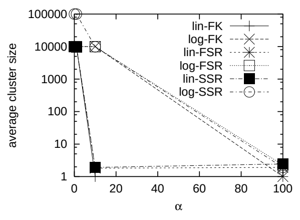


(c)

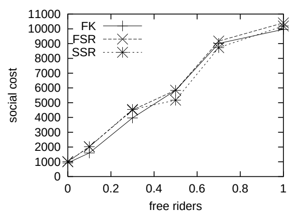


(d)

Fig. 2: (a) - (b) Uncoordinated and (c) - (d) coordinated protocols with varying ϵ.



(a)



(b)

Fig. 3: Effect of (a) α and (b) free riders.

results for the asymmetric scenario, since this is the case with the largest cost and overhead in terms of node moves (i.e., cluster changes). Our initial configuration is an overlay where each node forms a cluster of its own. We measure the overhead in terms of the average number of moves per node and the attained social cost. The social cost of the initial configuration is above 10,000 since all nodes have recall cost equal to 1.

The achieved social cost is mainly controlled by the *stopping condition* ϵ. For all values of ϵ, the social cost is reduced to at most 1,050, that is to almost 1/10th of its initial value (Fig. 2a). The basic uncoordinated protocol achieves the best social cost for the smaller value of ϵ (10^{-4}) within a few moves, while the one with monitoring achieves similar social cost for $\epsilon = 10^{-2}$ for the same number of moves (Fig. 2b). If we choose a smaller ϵ, the number of moves increases without improving the social cost. With partial knowledge (SSR or FSR), both protocols involve large overheads that for small ϵ prevent the system from reaching a stable state. As a result, the social cost is higher than the one achieved in the case of full knowledge. The results for the other parameters can be found in [9].

We repeat the experiment with the coordinated protocol (co). We set the percentage (K) of requests the clusters representatives grant to 70% of the total received requests, similarly to the update probability of the uncoordinated protocol that also implicitly controls the number of updates that occur. Coordination does not improve the social cost (Fig. 2c). The coordinated protocol (co) requires also high values of ϵ and for these values achieves approximately the same social cost with the uncoordinated ones with similar overhead (Fig. 2d). Lack of knowledge affects the coordinated protocol similarly to the uncoordinated ones.

In the rest of the experiments, we use the basic uncoordinated protocol, since it achieves a comparable social cost without global monitoring or coordination and set ϵ to 10^{-3}, which is an appropriate value for this protocol.

Cluster Formation In this set of experiments, we study the (basic) uncoordinated protocol. We focus on the three scenarios (i.e, symmetric, asymmetric and random from the case studies) and apply the uncoordinated protocol at a number of initial configurations. We then evaluate the quality of the resulting cluster configurations in terms of their social cost, the overhead required for achieving this cost in number of moves (i.e., cluster changes), the number of clusters formed, the achieved balance and the effect of the lack of global knowledge.

Let M be the number of groups as defined for configuration (C) for the symmetric and the asymmetric scenarios, for example, $M = 10$ for the symmetric one. We consider five initial system configurations: (i) each node forms a cluster of its own; (ii) all nodes form a single cluster; (iii) nodes are randomly distributed to l groups and we discern for different values of l the subcases: (a) $l = M$, (b) $l < M$ and (c) $l > M$; (iv) nodes are clustered according to their content and (v) nodes are clustered according to their queries. The results are summarized in Table 2b.

In all scenarios, the nodes reach an ϵ-Nash equilibrium regardless of the number of clusters in the initial configuration. The protocol does not require a predefined number of clusters, but this is dynamically determined. The value of δ_s is in all cases small (< 0.2), i.e., the formed clusters are (0.2)-size balanced. Similarly, δ_q and δ_r are around 0.4.

The system reaches a stable state even in the case of partial knowledge. However, while the social cost is only slightly worse, the number of moves per node increases considerably (the worst case being the asymmetric scenario). Comparing FSR and SSR, we observe that FSR requires more moves to reach a stable state, but achieves a better social cost than SSR. This is because, while it takes longer for the FSR to consider more clusters, it always evaluates their cost accurately. In contrast, while SSR locates relevant clusters faster, it cannot accurately determine the best among them.

For the symmetric scenario, the social cost depends only on the membership cost; the recall cost is in most cases zero, since all results of the local queries for each node are located within its cluster (Table 2b, lines 3-10). For the given $\alpha = 10$, the social cost achieved is close to the social optimum we have computed for Case Study (II.C) (when θ is logarithmic) which also considers similar symmetric nodes. Let us now discuss further the input configurations where the nodes are already clustered according to their content (case (iv)) or queries (case (v)). In this case, for symmetric nodes, the appropriate clusters are already formed (Table 2b, lines 9-10). We also observe that the lack of global knowledge does not affect the formed clusters, i.e., it does not damage good cluster configurations. For asymmetric nodes, both configurations are not stable. Thus, relying solely on content or query workload is not enough to provide the appropriate clustering.

In the random scenario, there is no underlying clustering (Table 2b, lines 19-21). We consider two initial configurations: a single cluster (case (i)) and singleton clusters (case(ii)). In this case, the resulting cluster configuration depends mostly on the membership cost. Due to its low value (logarithmic θ and small α), a social cost close to the optimum is achieved by the configuration where all nodes form a single cluster. To further see the role of α, we performed experiments with different values of α (Fig. 3a) and both a logarithmic (log) and a linear (lin) θ function. For example, for a large value of α ($\alpha = 100$), both initial configurations converge to the configuration of singleton clusters. Thus, by tuning α we can favor configurations with either a small number of larger clusters or a large number of smaller clusters.

Finally, let us consider clustering in the case of free riders (e.g., [1]), which are nodes that use data offered by others, without contributing any content. We model a free-rider n_i as a node with $r(q, n_i) = 0$, $\forall$ q in Q. Increasing the number of free riders gradually degenerates the overlay to one in which each node forms a cluster of its own (Fig. 3b), since free riders do not acquire any gain by forming clusters among themselves.

Cluster Updates. In this set of experiments, we evaluate how the protocol reacts to changes. We start from a "good" cluster configuration and apply query and content updates. In particular, we focus on a general query update scenario ($WSc1$), in which a data category becomes more popular; $x\%$ of nodes selected randomly from all clusters change their query workload to queries for this category.

Compared to a static overlay in which no measures are taken to deal with updates (*no-change*), our protocol reduces the social cost up to $1/3$ of its value (Fig. 4a). Figure 4b shows the associated overhead, i.e., how the social cost improves as the average number of moves per node increases, for 50% of updated nodes.

We also compare the social cost achieved by playing the game repeatedly with applying the protocol from scratch. Re-clustering from scratch (Fig. 4c) reduces the social cost of up to 10% compared to our incremental reformulation protocol (Fig. 4a). However, re-clustering entails 270 turns (total times any node evaluates a positive gain) even with FK, while the incremental reformulation entails only $10 - 15$ turns. Also, re-clustering requires more than $|V|$ moves, regardless of the number of updated nodes, unlike reformulation that mostly affects the updated nodes. For FSR and SSR, the difference between re-clustering and reformulation are even smaller in terms of the social cost, but the overhead is increased.

Finally, we consider a content update scenario ($CSc1$) corresponding to $WSc1$, where $x\%$ of the nodes selected randomly from all clusters change their content to a specific category. Nodes react faster to query changes than to content updates (Fig. 4d). This is because a change in the content of a node does not instigate a strategy change for this specific node, since it does not affect its cost, while for a node to become aware of the change in the content of others, this change has to affect a large number of nodes. Results including additional query and update scenarios as well as topology updates (nodes join/leave) can be found in [9].

6 Related Work

There has been a lot of work on the application of game theory to network creation [17]. In the basic form of the network formation game [6], nodes act as uncoordinated selfish agents whose goal is to choose other nodes to link with. The nodes pay for the creation of a link but gain by reducing their shortest distance to all other nodes in the system. In [5], theoretical results regarding the achieved social cost are presented including variations of the game, such as a bilateral version where links are created only with the consent of both endpoints.

Various versions of the game have been proposed for overlay networks. The novelty of our approach lies in using games to model clustered overlay creation towards improving query recall. We present next a couple of approaches with some resemblance to our game (additional related work can be found in [9]). In [11], degrees of selfishness are introduced through a social range matrix f, where f_{ij} shows how much node n_i cares about node n_j. Then, the cost of n_i is the sum of the costs of all other nodes n_j weighted by f_{ij}. As a case study, a network formation game is presented where the cost has two components: one modeling the connection cost and one modeling the gain of being a member of a connected group as a function of the number of nodes within distance at maximum r-hops. In our model, we consider

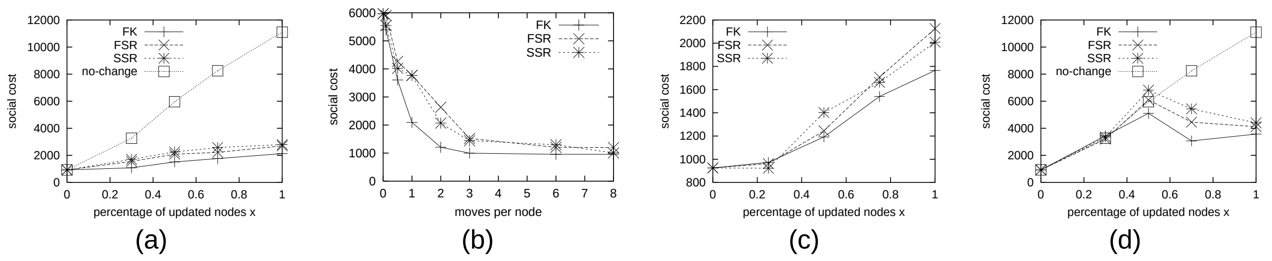


Fig. 4: Updates (a) with social cost, and (b) moves, (c) re-clustering and (d) content updates.

clustered overlays instead of unstructured ones and the gain depends on the recall not on the number of nodes. In [12], a model is introduced that enforces strict bounds on the out-degree of the nodes, links are directed and nodes are allowed to express preferences regarding the choice of their neighbors. Our approach can be viewed as setting these preferences based on recall benefits.

Many recent research efforts in p2p overlays have focused on organizing nodes in clusters [9]. In most cases, the focus is on cluster formation and query processing and the adaptation of the overlay is not addressed. None of these efforts takes a game-theoretic approach.

This paper is a revised version of a conference paper [10] with an extended theoretical and experimental evaluation that includes partial knowledge.

7 Summary and Future Work

In this paper, we have modeled the creation and maintenance of clustered overlays as a game. Nodes act as players that choose their strategy, i.e., which clusters to join, so as to minimize a utility function of the cluster membership cost and query recall. To cope with churn and query and content updates, nodes re-evaluate their strategies resulting in dynamic re-clustering.

There are many directions for future work. One is to consider altruistic nodes that aim at improving either the cost of other nodes or the overall system cost. Some preliminary results for the first case were reported in [10]. With regards to the model, possible extensions include adding to the cost function an explicit load-balance component as well as a component for the inter-cluster communication. Also, we plan to modify our protocol to efficiently handle multiple cluster membership without increasing its complexity. Finally, another direction is identifying possible connections between our game-based approach to clustering and traditional approaches based on distance measures.

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] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems, Technical Report, Computer Science Department, Stanford University, 2002.
[5] E. D. Demaine, M. Hajiaghayi, H. Mahini and M. Zadimoghaddam. The price of anarchy in network creation games. In *PODC*, 2007.
[6] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *PODC*, 2003.
[7] P. Garbacki, D. H. J. Epema, and M. van Steen. Optimizing peer relationships in a super-peer network. In *ICDCS*, 2007.
[8] 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.
[9] G. Koloniari and E. Pitoura. A Game Theoretic Approach to the Formation of Clustered Overlay Networks. Supplementary Material, 2010.
[10] G. Koloniari and E. Pitoura. A recall-based cluster formation game in peer-to-peer systems. In *VLDB*, 2009.
[11] P. Kuznetsov and Stefan Schmid. Towards network games with social preferences. In *SIROCCO*, 2010.
[12] N. Laoutaris, G. Smaragdakis, A. Bestavros, and J. W. Byers. Implications of selfish neighbor selection in overlay networks. In *INFOCOM*, 2007.
[13] A. Legout, N. Liogkas, E. Kohler and L. Zhang. Clustering and sharing incentives in BitTorrent systems. In *SIGMETRICS*, 2007.
[14] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *IMC*, 2007.
[15] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, 2003.
[16] 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.
[17] E. Tardos, and T. Wexler. Network formation games. Chapter 19 In *Algorithmic Game Theory*, Cambridge University Press, 2007.
[18] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos. Towards high performance peer-to-peer content and resource sharing systems. In *CIDR*, 2003.

Georgia Koloniari Georgia Koloniari has received her Ph. D. (2009) in Computer Science from the Department of Computer Science of the University of Ioannina, Greece. She also holds a B.Sc (2001) in Computer Science and an M.Sc. (2003) in Computer Science in the area of Computational Systems from the University of Ioannina, Greece.

Evaggelia Pitoura Evaggelia Pitoura is an associate professor at the Department of Computer Science of the University of Ioannina, Greece. She holds a B.Sc (1990) in Computer Engineering from the University of Patras, Greece and an M.Sc. (1993) and Ph.D.(1995) in Computer Science from Purdue University, USA.