Noname manuscript No.
(will be inserted by the editor)

Efficient Search based on Content Similarity over Self-Organizing P2P Networks

Christos Doulkeridis · Akrivi Vlachou · Kjetil Nørvåg · Yannis Kotidis · Michalis Vazirgiannis

the date of receipt and acceptance should be inserted later

Abstract The advent of the World Wide Web has made an enormous amount of information available to everyone and the widespread use of digital equipment enables end-users (*peers*) to produce their own digital content. This vast amount of information requires scalable data management systems. Peer-to-peer (P2P) systems have so far been well established in several application areas, with file-sharing being the most prominent. The next challenge that needs to be addressed is (more complex) data sharing, management and query processing, thus facilitating the delivery of a wide spectrum of novel data-centric applications to the end-user, while providing high Quality-of-Service. In this paper, we propose a self-organizing P2P system that is capable to identify peers with similar content and intentionally assign them to the same super-peer. During content retrieval, fewer super-peers need to be contacted and therefore efficient similarity search is supported, in terms of reduced network traffic and contacted peers. Our approach increases the responsiveness and reliability of a P2P system and we demonstrate the advantages of our approach using large-scale simulations.

Keywords Peer-to-peer systems · content search · self-organizing overlay networks

Christos Doulkeridis · Akrivi Vlachou · Kjetil Nørvåg
Dept. of Computer and Information Science, NTNU
7491 Trondheim, Norway
E-mail: {cdoulk,vlachou,Kjetil.Norvag}@idi.ntnu.no

Yannis Kotidis · Michalis Vazirgiannis
Dept. of Informatics, AUEB
10434 Athens, Greece
E-mail: {kotidis,mvazirg}@aueb.gr

1 Introduction

Peer-to-peer (P2P) systems have so far been well established in several application areas, with file-sharing being the most prominent. In P2P file-sharing networks (eMule, KaZaA), users both share their files and submit requests for files. Indeed the importance of P2P systems that fulfill this self-arising user requirement is immense, as there exist studies [15] that report that they are responsible for 20% of Internet traffic, and this tendency is increasing. Recently P2P technology has also been employed in applications in order to improve dependability, one notable example is Skype [2].

The widespread use of digital equipment enables end-users to produce their own digital content, which they are willing to share based on the extensive use of popular applications such as Flickr[1] or Facebook[2]. Due to the enormous amount of information available, large-scale architectures supporting user requests for similar content to the query are necessary. For such applications it is evidently suitable to follow the P2P paradigm. The challenge that needs to be addressed is (more complex) data sharing, management and query processing, thus facilitating the delivery of a wide spectrum of novel applications to the end-user. In such a scenario, peers can be considered autonomous databases that join the network in order to 1) access data residing on other peers and 2) provide controlled access to their data. A notable fact is that peers do not wish to publish their data to the system due to privacy reasons, so data remains resident on the peer. Instead, only indexing information is distributed to the rest of the network.

We consider the context of our work as *data-centric*, in contrast to existing P2P file-sharing applications,

[1] `http://www.flickr.com/`
[2] `http://www.facebook.com/`

and we outline important differences between these areas. In the file-centric case, shared content is in the form of mostly static files, thus allowing replication of popular files, in order to support retrieval. Moreover, the size of file metadata is small, for example the file name, therefore it can be easily distributed and maintained. In the data-centric case, the data is highly dynamic and much more complex. Also, data is more voluminous compared to file metadata. Querying data requires query processing capabilities individually on each peer, to support efficient processing of data. Thus, the P2P system needs to support distributed mechanisms for advanced querying, such as similarity search, that go beyond exact matching of file names and pose new challenges. Moreover, replication of the actual data may lead to bottlenecks, due to the high rate of updates, therefore compact and robust data summaries need to be defined and used as representatives of the actual data. It should be noted that data-centric capabilities can also be added as functionality to traditional file-centric applications, by providing more advanced search functionality.

Super-peer networks – also called hybrid P2P – constitute a scalable extension of the classic client-server architecture. In this paper, we focus on a hybrid P2P system, with many simple peers and few enhanced peers (in terms of processing power, storage capacity or network connectivity) that play the role of super-peers [9, 22]. Content producers and providers (mentioned as peers) that want to share their data connect to a super-peer, as in the case of the file-sharing applications. By connecting to a super-peer some descriptive features (represented as points in a high dimensional space) of the peer data are transferred to the super-peer, in order to make the peer's data searchable.

An important aspect and one of the main motivations for P2P networks, is their inherent contributions in dependability. Considering the main aspects of dependability, i.e., availability, reliability, safety, integrity and maintainability [1], these relate to P2P as follows. Increased availability, reliability and safety are achieved by means of replication and self-organization of service-providers, thus avoiding single points of failure. In addition, peer and super-peer failures are timely detected and the establishment of new connections ensures the connectivity of the P2P network. Maintainability is also supported, as the service provided is not compromised by peer failures. Another viewpoint of dependability that is also an important issue in a P2P system is security. However, this issue is mostly orthogonal to our research focus and will not be covered in this paper.

In this work, we focus on P2P applications that aim at facilitating data exchange. These data-centric systems provide the service of data retrieval by means of distributed query processing. Correct service is considered to be delivered, when a sufficient amount of matching data is found. Otherwise, a service failure has appeared. The described application can also be considered as a soft real-time system, since results have to be provided to the user within an accepted time. It should be noted that this contrasts to file-sharing applications, where it is usually sufficient to find the location of only one copy of a file, and real-time response for file-retrieval is not required.

During query processing, the responsiveness of the P2P system affects its overall performance and dependability. One of the most important factors that influences the performance of P2P systems is the underlying data distribution to peers (and super-peers). In the general case, when a peer joins the network, it is assigned to one available super-peer, usually in a random way. This leads to near-uniform data distribution at super-peer level, which degrades query processing performance, since a query may – in worst case – have to contact all super-peers. The problem with this traditional allocation will also be evident from our experimental results later in this paper. However, data on peers is usually clustered into a few thematic areas that reflect the user's interests, forming virtual user communities. It is therefore necessary to change the data uniformity at super-peer level, detect peers with similar content that belong to the same community and maintain a clustered data distribution at super-peer level, as shown in Fig. 1. Additionally, this should occur in a self-organizing way, without explicit intervention from the P2P system designer. Then, queries can be directed to specific super-peers only, thus improving query processing performance.

The main topic of this paper is a self-organizing P2P system that is capable to identify data clusters on peers and intentionally make peers that belong to the same cluster attach to the same super-peer. Each peer can connect to as many as k super-peers, where k denotes the number of the peer's clusters. This leads to a clustered super-peer overlay topology, which increases the dependability of our system, in terms of responsiveness and availability. Then, we show the advantages of the proposed approach by studying query evaluation, namely range query processing, over the new overlay topology. Our experimental study proves that the new organization scheme is meaningful, as it achieves significant improvement over the initial data-agnostic peer to super-peer assignment.

The individual contributions of this paper are:

1. Focusing on a data-centric context, we propose a novel decentralized algorithm that identifies similar

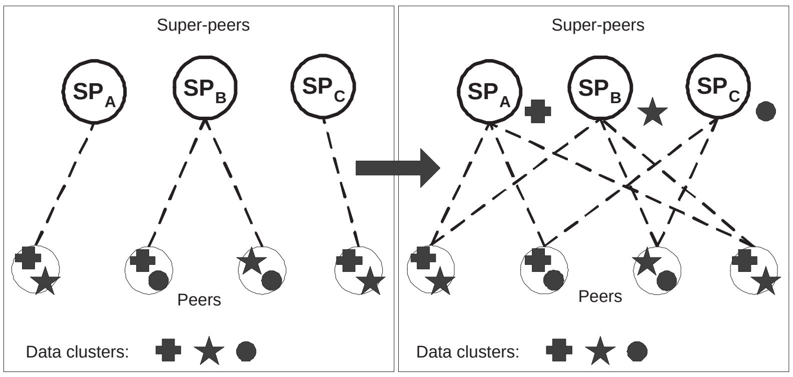


Fig. 1 Data distribution is clustered on peers but uniform at super-peers (left). In the new super-peer topology (right) data becomes clustered at super-peer level, with each super-peer responsible for one cluster.

data clusters on different peers, therefore allowing us to change the overall data distribution on super-peers from uniform to clustered. This is a particularly challenging problem for improving the performance of data-oriented P2P systems, which has not been addressed yet.

2. We sketch algorithms for performing efficient query processing and show how the responsiveness of a P2P system increases during range query processing over the new organization of peers.
3. We provide an extensive experimental study showing in a quantitative way the benefits of our approach during query processing.

The rest of this paper is organized in the following way: In Section 2, we overview the related work, while in Section 3 we describe the preliminaries necessary for exposing our approach. In Section 4, we present our algorithm for constructing the clustered super-peer overlay network. In Section 5, we discuss aspects of fault-tolerance. Finally, in Section 6 we show our experimental evaluation and we conclude in Section 7.

2 Related work

Varying the super-peer topology is an important issue for designing scalable P2P networks. Studying the effect of increasing super-peer outdegree is first studied in [22]. The authors argue in favor of maximizing super-peer outdegree in order to decrease the load on the super-peers and increase the number of results. The importance of intentionally specifying the connectivity of super-peers is highlighted. Our work is motivated by this observation.

Dynamically changing peer connections has also been studied in the context of pure unstructured P2P systems [4], where the authors propose Gia, which improves the scalability of the original Gnutella protocol. A similar approach for dynamic adaptation of the P2P topology has been examined in [5], where the notion of *acquaintances* is introduced. The basic idea is that peers gradually create neighboring links to other peers based on their interests. In a similar context to our work regarding the P2P architecture, *associative overlays* [6] have been proposed as a means to identify associations on peers. This approach reflects an observation made by several papers, namely that a peer that has previously satisfied a query is a likely candidate for future requests by the same originator. All these approaches focus on dynamic topology adaptation in unstructured P2P systems, however the focus is on file-sharing applications, which is quite different to data-centric applications. Topology adaptation does not always aim at increasing search quality or reducing bandwidth usage, but also focuses on the discovery and creation of connections to more trustworthy peers [16].

In recent work, we proposed a dynamically formed, hierarchical P2P overlay network, named DESENT [8], for decentralized and distributed clustering of peer contents. However, the focus of DESENT is on information retrieval over widely distributed document collections. Therefore, extensions are necessary in order to support complex query processing. This paper partly addresses this need as well, capitalizing on DESENT and its findings. Other approaches for creation of hierarchies have been proposed, for example by Mathy et al. [17], where the aim is an efficient overlay for multicasting.

A self-organizing super-peer network architecture is presented in [10], named SOSPNET, which deals with the issue of how clients connect to a super-peer. This work addresses the same problems as DESENT, namely: 1) the restriction of one peer connection to super-peer, 2) the static and random assignment to super-peers, and 3) the indexing of a peer's contents by one super-peer only. Although relevant to our approach, the main difference is that SOSPNET organizes peers according to requests for files, while our approach is data-centric and focuses on data organization. Moreover, our approach can handle more complex data types than files and process complex queries. In [14], Jesi et al. present a decentralized, self-organizing general protocol for the construction of proximity-aware, super-peer based overlay topologies.

Except from the work described above, little work exists on assigning peer to super-peers. However, a related topic is adaptation of connections between peers in a pure P2P network (i.e., non-hierarchical with no super-peers, instead all peers are considered equal). In such a network, peers initially only have connections to their nearest neighbors. By adaptation or adding links to peers sharing similar contents, search efficiency can be improved. The challenge in this setting is to know contents of remote peers. One way of solving this, is to use results of queries to know remote contents and create links [7,18,19]. A second approach is to use gossiping to distribute information about contents, so that remote peers can introduce links [21]. Yet another approach is to consider interest, for example by considering overlap in query results or cache contents [3,20].

Although there exist some papers that deal with P2P data management [11,12], they usually focus on schema mappings and SQL query processing. These papers are not directly relevant to our approach, as they assume different schemas on peers. Our approach studies focused user communities that share similar interests, hence they are expected to have common schemas. Furthermore, only few papers have tackled the same issue and their context is request-based [10] or interest-based [5]. In contrast, we focus on a data-centric context, which is more challenging as complex query types (that go beyond file searching using keywords) can be supported.

3 Preliminaries

In this section, we briefly present some preliminaries, namely an algorithm (DESENT) for decentralized and distributed generation of clusters that span the entire P2P network, and a framework (SIMPEER) that relies on data clustering for processing range queries over a super-peer architecture. DESENT generates clusters, which are used by our proposed approach to deliberately assign peers to super-peers, reflecting a clustered distribution at super-peer level. This improves the performance of our system, since the query is routed only to few super-peers. Then, in Section 4 we show how to improve the performance of SIMPEER, and therefore the responsiveness of our system, in a self-organizing way through the distributed generated clusters.

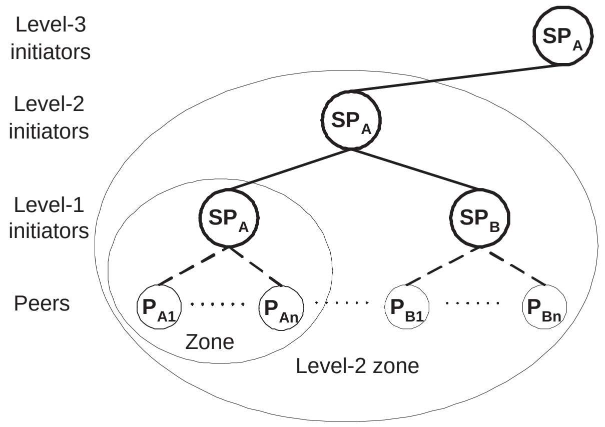


Fig. 2 DESENT hierarchy of zones and initiators.

3.1 Decentralized and distributed overlay creation

In order to discover clusters of data that belong to any peer in a pure unstructured (Gnutella-like) P2P network, no matter its network distance, we employ a variant of DESENT [8]. The reasons for the choice of DESENT are the completely *distributed* and *decentralized* creation of the DESENT hierarchy, its low creation cost and robustness. The most important details of the basic DESENT algorithm are now described. For more detailed overview and algorithms we refer to [8].

In order to perform global clustering, DESENT creates a hierarchy of peers in a decentralized way. This hierarchy can later be used for building overlays for searching as described in [8], but also for other purposes like aggregation of data or statistics about contents from participating peers.

Before the creation of the hierarchy, each peer that joins the network runs a local clustering algorithm to create its local clusters. For each cluster C_i there exists a cluster description, which in the case of multidimensional data typically is the centroid and radius of the cluster, denoted as $C_i(K_i, r_i)$.

After local clustering, the DESENT hierarchy is formed. For an illustrative example of the DESENT hierarchy see Fig. 2. The bottom level consists of the individual peers. Then, neighboring peers (network-wise) create *zones* of approximate size S_Z peers (i.e. groups of peers) around an initiating peer (initiator) which acts as a zone controller. These level-1 initiators are mostly uniformly distributed over the network, and are selected independently of each other in a pseudo-random way. The initiators form the next level of the hierarchy, they are responsible for the peers in their zones, and they aggregate the local clusters of their peers into super-clusters.

In the following phases, super-zones are created, which consist of a number of neighboring zones from the previous level. Each super-zone is represented by a super-zone initiator that is responsible for the initiators in its zone and aggregates the clusters of these initiators. The zone initiators essentially form a P2P network similar to the original P2P network, and the aforementioned process is repeated recursively, using the zone initiators as peers. In this way, a hierarchy of initiators is created, with each initiator creating clusters that refer to the contents of all peers in the tree rooted at that initiator. At the end, at the top-level initiator, a set of clusters that span the contents of the entire network are available.

3.2 Query processing

SIMPEER [9] relies on a three-level clustering scheme and supports efficient P2P similarity search in metric spaces. Given a super-peer network, each peer connects to any super-peer and maintains its own data, represented in a high dimensional space. In a construction phase, each peer applies a clustering algorithm on its local data, in order to create data summaries that make its data searchable by other peers. Thereafter, each super-peer gathers the clusters of its associated peers and applies on them a clustering algorithm resulting in a new cluster set that describe the data indexed by this super-peer. These clusters are broadcasted at the super-peer network, in order to form *routing clusters* at super-peers.

At query time, each super-peer decides where to forward a query, based on its routing clusters. Assume a range query $R(q, r)$ initiated at a super-peer SP_q. First, SP_q examines its routing clusters to find to which of its neighboring super-peers the query should be forwarded to. Each recipient super-peer SP_r checks whether its local peers can provide any results, by inspecting their clusters. If the query overlaps with some clusters, SP_r contacts only the peers responsible for these clusters. Otherwise, SP_r simply forwards the query to its neighboring super-peers (using its routing clusters). In order to support efficient local query processing, SIMPEER applies state of the art indexing techniques [13] for maintaining the local data points and the routing clusters. As shown in [9], SIMPEER also supports efficient nearest neighbor search, and therefore the approach presented in this paper can also support nearest neighbor search, however we focus on range queries in the rest of the paper.

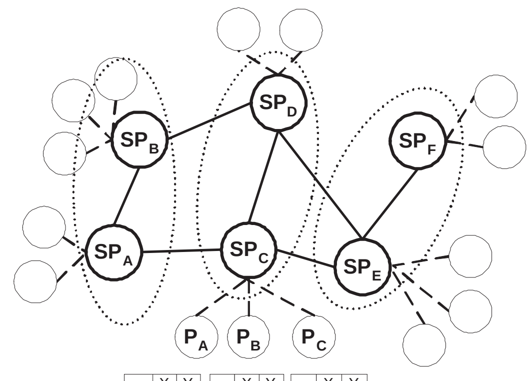
SIMPEER has been shown to work well in the case of clustered data distribution at super-peer level. This means that only few super-peers and peers that can provide results that match the query are queried, resulting in reduced network traffic and response time. However, the performance of SIMPEER degrades in the case of uniform data distribution to super-peers, since SIMPEER may have to contact all super-peers to retrieve the correct result. The challenge addressed in the following is generating a clustered distribution at super-peer level from a uniform distribution of data over the super-peer network.

4 Peer self-organization

Consider a super-peer network that consists of N_{sp} super-peers connected to a limited set of at most DEG_{sp} other super-peers. Each super-peer SP_i is responsible for DEG_p simple peers, which connect to SP_i directly. Each peer P_i holds n_i d-dimensional points, denoted as a set O_i $(1 \leq i \leq N_p)$. We assume horizontal data distribution to the N_p peers, hence the size of the complete set of points is $n = \sum_{i=1}^{N_p} n_i$ and the dataset O is the union of all peers' datasets O_i $(O = \cup O_i)$. Each peer maintains its own data objects, such as images or documents, while the d-dimensional points are features extracted from the objects, in order to make any peer's data searchable. In Fig. 3, we depict for example the 2-dimensional data that are stored by peers P_A, P_B and P_C, which are connected to super-peer SP_C.

Prior to the actual content delivery, our approach for content organization consists of two phases. First, a global cluster creation process is performed, using DESENT as a content aggregation network. The initial super-peers are organized in a DESENT hierarchical overlay, aggregating and clustering information of super-peers of the level below. As a result N_C global clusters are created at the top super-peer in the hierarchy. In the second phase, the N_C clusters are dynamically decomposed to eventually form N_{sp} groups, essentially one group for each initial super-peer. In this way the initial random peer to super-peer assignment changes and the initial super-peers are assigned with peers with similar content in an intentional manner.

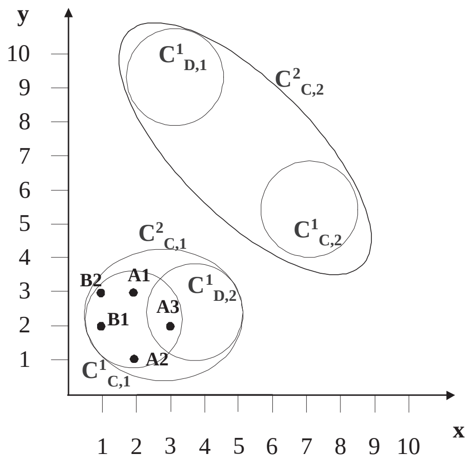
It should be emphasized that assembling all clusters at one super-peer (the root) is infeasible, since it does not constitute a scalable solution, especially for large super-peer networks. For example, in the case that the root fails, all clustering information will be lost. Also the particular super-peer would be a single point of failure, whereas in our approach the information held by any super-peer can be reassembled fast by its super-peers below, thus achieving fault-tolerance.



Local Data

	X	Y		X	Y		X	Y
A1	2	3	B1	1	2	C1	8	6
A2	2	1	B2	1	3	C2	7	9
A3	3	2	B3	2	2	C3	7	8

(a) Initial super-peer topology.



(b) Data space.

Fig. 3 Super-peer network and data space.

4.1 Phase 1: Global cluster creation

During global cluster creation, each initiator super-peer – at any level of the hierarchy – assembles the clusters of other super-peers within its zone and runs a clustering algorithm to produce k intra-zone clusters. In general, any centralized clustering algorithm can be used, but for ease of exposition in the following let us assume we use k-means. As this process runs iteratively and the super-peer hierarchy is built bottom-up, eventually at the top-level super-peer there will exist $N_C{=}k$ clusters that describe the data of the entire network. We assume that $k < N_{sp}$.

In Fig. 3(a), the zones created during the iterative zone creation are depicted. Fig. 3(b) shows the 2-dimensional data space of our example with two clusters per super-peer. In the figure we use the notation: $C^i_{A,j}$ to represent the j-th cluster of SP_A at level-i. Initially, at level-1 there exists a super-peer topology as depicted in Fig. 4. At level-2 three zones are created: SP_A-SP_B,SP_C-SP_D,SP_E-SP_F. At level-3 one zone is created with initiator super-peer SP_B.

An interesting observation is that the information about which clusters of the previous level form a new cluster C_i is maintained at super-peer SP_i, and it is not necessary to broadcast it to the rest of the network. Each cluster C_i is represented by a tuple $C_i : \{(K_i, r_i), score_i, SP_i\}$, where K_i is the d-dimensional cluster centroid, r_i is the cluster radius, $score_i$ is a score indicating the cluster's quality and SP_i is the super-peer that generated C_i. Regarding cluster quality, obviously different quality indices can be employed. However, as progressive clustering of clusters may result in higher level clusters that contain large areas of empty space in the d-dimensional space, we define

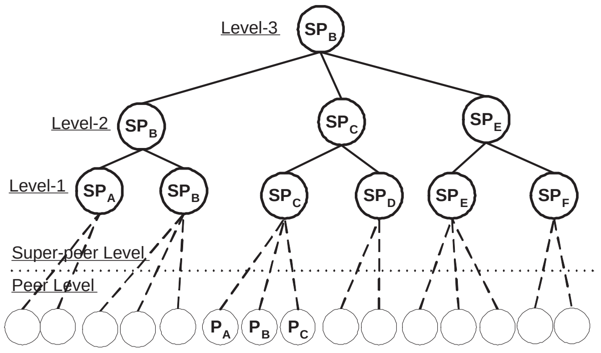


Fig. 4 Super-peer hierarchy of DESENT.

a quality index named *compactness* that captures this effect. Given a cluster $C_i(K_i, r_i)$ which contains clusters $C_j(K_j, r_j)$, $0 \leq j \leq k$, the compactness of C_i is defined as:

$$compactness = \frac{\sum_{j=1}^{j=k} Volume(C_j)}{Volume(C_i)} \qquad (1)$$

A value of compactness larger than 1 means that there is high overlap of the clusters of the lower clustering level. A small value of compactness (much smaller than 1) means that there is a large empty space in the cluster C. Notice that the compactness of a cluster C_i is computed at the super-peer that generates C_i, as it also maintains the necessary information to compute the volume of C_i's subclusters.

4.2 Phase 2: Cluster assignment to groups

In the next phase, the aim is to decompose the k global clusters, so that at least one cluster is assigned to each of the N_{sp} groups. Later each group can be assigned to a

Algorithm 1 Assignment of k clusters to N_{sp} groups.

```
1: Input: $k$ top-level clusters $\{C_i, ..., C_k\}$
2: Output: $N_{sp}$ groups
3: $count \leftarrow 0$
4: $groups \leftarrow \{\emptyset\}$
5: $heap \leftarrow \{C_1, ..., C_k\}$ // heap initialized with top-level clusters
6: while ($count < N_{sp}$) do
7:   if ($heap = \{\emptyset\}$) then
8:     $C_{min} \leftarrow groups.minCompClu()$
9:     $groups.remove(C_{min})$
10:    $count \leftarrow count - 1$
11:    $heap \leftarrow \{C'_1, ..., C'_k\}$ // $C'_i$ denotes a subcluster of $C_{min}$
12:  end if
13:  $groups.add(heap.pop())$
14:  $count \leftarrow count + 1$
15: end while
16: if ($heap \neq \{\emptyset\}$) then
17:   $groups.add(heap)$
18: end if
```

super-peer. Obviously, each group should consist of similar clusters, in order to have super-peers with clustered data. Based on the hierarchical clustering, clusters that are assigned to a group aggregate more than one similar peers' clusters. The process of cluster assignment to groups is performed at the top level initiator in the hierarchy. Notice that the lack of global knowledge of lower level cluster descriptions makes the task of cluster assignment particularly challenging.

In the following, we sketch our algorithm for decomposing the k clusters to a predefined set of N_{sp} groups. The pseudocode is presented in Algorithm 1. A heap is used to keep the clusters that will be assigned to groups. Initially the heap maintains the top-level clusters (line 5). Until all groups have been assigned with clusters (line 6), we pick the next cluster from the heap and add it to an empty group (line 13). In case the heap becomes empty (line 7), the cluster with the minimum compactness from those already assigned to groups is selected to be decomposed (line 8). This cluster is removed from the groups (line 9) and its subclusters are inserted in the heap (line 10). Function $minCompClu()$ is used (line 8) to return the cluster C_{min} with the minimum compactness, from those clusters already assigned to groups. In the case that all groups have clusters assigned, but some clusters are left in the heap (line 16), these clusters are assigned to the most suitable group (line 17) based on similarity with already assigned clusters.

Notice that the replacement of a cluster C_{min} by its subclusters $\{C'_1, ..., C'_k\}$ requires some communication between the top level super-peer and the super-peer SP that generated C_{min}. This communication can be efficiently accomplished by using the super-peer hierarchy (Fig. 4). Whenever a cluster C_{min} needs to be decomposed, the top level super-peer contacts the super-peer (SP) responsible for C_{min} using the super-peer hierarchy, in order to retrieve the decomposed cluster descriptions $C'_i : \{(K'_i, r'_i), score'_i, SP'_i\}$. In any case, the cost for such a communication is bounded by a number of $log_{S_Z} N_p$ (equal to the height of the hierarchy) messages, which are required to contact SP.

Another interesting observation is that we decompose the super-peers' top level clusters to clusters from lower levels. This is because we want to have more accurate cluster representations as well as keep as much detail as possible in the cluster descriptions. Consider for example Fig. 3(b). Instead of assigning to a super-peer SP cluster $C^2_{C,2}$, it is preferable to assign clusters $C^1_{D,1}, C^1_{C,2}$, as these clusters are more detailed and contain less empty space. However, both $C^1_{D,1}, C^1_{C,2}$ should be assigned to the same super-peer. The main issue is which level of detail of clusters to assign to each super-peer, while at the same time assigning similar clusters to the same super-peer, and this is addressed by Algorithm 1.

4.3 Discussion

Although the problem of changing the data distribution from uniform to clustered is challenging and hard to solve under the assumption of lack of global knowledge, our approach described above for content organization has several attractive features.

The top-level super-peer aggregates only information about the super-peer hierarchy. The cluster information that it aggregates consists only of cluster descriptions about the clusters one level below. Thus the total clustering information is distributed over the super-peers of the hierarchy. The desired requirement accomplished is that the cluster descriptions of all clusters do not have to be sent to one central location.

Similarly, a level-i super-peer knows how many super-peers exist in its subtree. Furthermore, each super-peer knows the super-peers that can be reached in its subtree. It should be stressed that this information is maintained only until the content organization is completed. Afterwards, at query time, the initial super-peer topology is used, therefore each super-peer routes any query through its neighboring super-peers.

5 Fault-tolerance and resilience

An inherent feature of P2P systems is that they support fault-tolerance and resilience in an effective way. In this section, we focus on the most important types of failures that can occur in our system, and we describe how they are efficiently handled. Such failures include peer and super-peer failures, as well as maintenance of the clustering information.

The number of failures inevitably increases with the number of peers being involved. In a P2P network, peer failures are relatively frequent. However, the responsible super-peer easily detects a peer failure, by sending periodic *ping* messages. Then, the super-peer marks the failing peer and its data clusters as absent, and maintains this information until the next time that the peer reconnects. This technique ensures incremental maintenance of peer data, as the next time the peer joins the P2P system, only changes to its data clusters need to be propagated to the super-peer.

Other types of faults are those related to availability during the construction of the super-peer hierarchy. In order to ensure that no super-peer becomes a single point of failure or a bottleneck, a load-balancing mechanism is required. In our approach, this is accomplished by the mechanism of zones (described in Section 3.1), which essentially assigns partitions of peers to super-peers. Furthermore, by controlling the zone size, our approach provides a mechanism to avoid having super-peers assigned with heavy processing burden.

After the clustered topology has been formed, maintenance of the super-peer network itself is performed similarly to traditional super-peer networks [22]. A failure in such systems is usually detected by lack of response. Finally, in order to avoid complete recalculation of clusters and assignment during failure of super-peers, we use k-replication of important data. This means that information on clusters and assignment is replicated over k super-peers.

6 Experimental study

In this section we study the efficiency of the proposed approach using large-scale simulations, with a simulator prototype implemented in Java. The simulations run on 2.8GHz Intel processors with 512MB RAM. In order to be able to test the algorithms with realistic network sizes, we ran multiple instances of the peers on the same machine and simulated the network interconnection.

In our experiments, we used the GT-ITM topology generator[3] to create well-connected random graphs of N_{sp} super-peers with average connectivity DEG_{sp}=4. We vary the following values: network size N_p=2000-20000 peers, number of super-peers N_{sp}=$\{200, 400, 600\}$ and DEG_p = 10-50. K-means is executed with k=5. We also tried other setups, for example with varying k, not shown since they produce similar comparative results.

[3] Available at: http://www.cc.gatech.edu/projects/gtitm/

In order to evaluate the scalability of our approach, we experimented with synthetic clustered data collections, which are horizontally partitioned evenly among the peers. For the clustered dataset generation, we randomly create N_{sp} d-dimensional cluster centroids and all associated peers obtain k centroids from them selected at random. Thereafter, the peers' objects are generated by following a Gaussian distribution on each axis with variance 0.025, and a mean equal to the corresponding coordinate of the centroid.

We conduct experiments varying the dimensionality (3-7d) and the cardinality (1M-10M) of the dataset. We keep the number of objects per peer (n/N_p) constant and equal to 500. In all cases, we generate 20 queries uniformly distributed and we show the average values. A peer initiator is randomly selected for each query. In our experiments we vary the query selectivity (res) indicating the number of results that are returned to the user. A higher number of results means that probably more super-peers have to be contacted, in order to retrieve all relevant data.

We study the performance of query processing and the benefits of our approach in comparison to a typical super-peer network where peers join randomly selected super-peers. The random super-peer assignment is denoted as *super-peer* in all charts, while our proposed approach appears with the name *SON-based*[4]. In our simulations, we have as starting point a super-peer network with random assignment of peers to super-peers. Then, the approach described in Section 4 takes place resulting in a new super-peer network topology. We execute range queries on the initial random and the new super-peer network topology using SIMPEER, and we report the results. We measure the comparative performance of both approaches in terms of: 1) number of messages for searching, 2) percentage of contacted peers with results for the query, and 3) number of contacted peers.

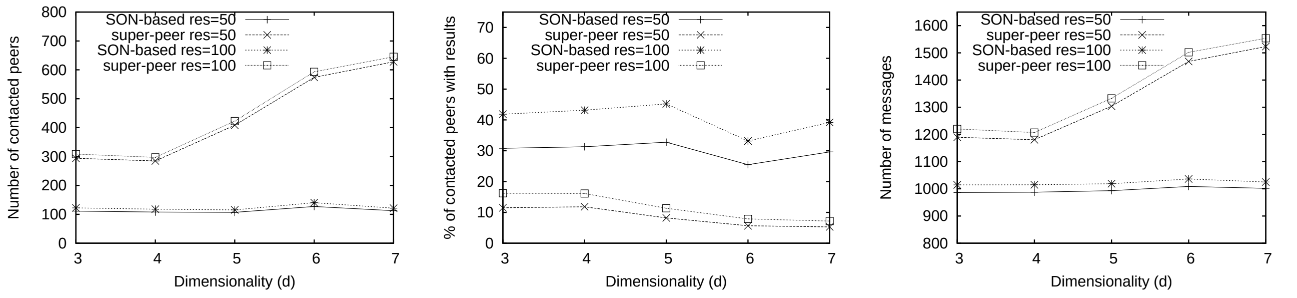
6.1 Experimental results

Our experimental evaluation consists of a scalability study of the proposed approach with respect to: 1) the dataset dimensionality d, 2) the network size N_p, 3) the average peer to super-peer connectivity DEG_p, and 4) the number of super-peers N_{sp}.

6.1.1 Scalability with dimensionality

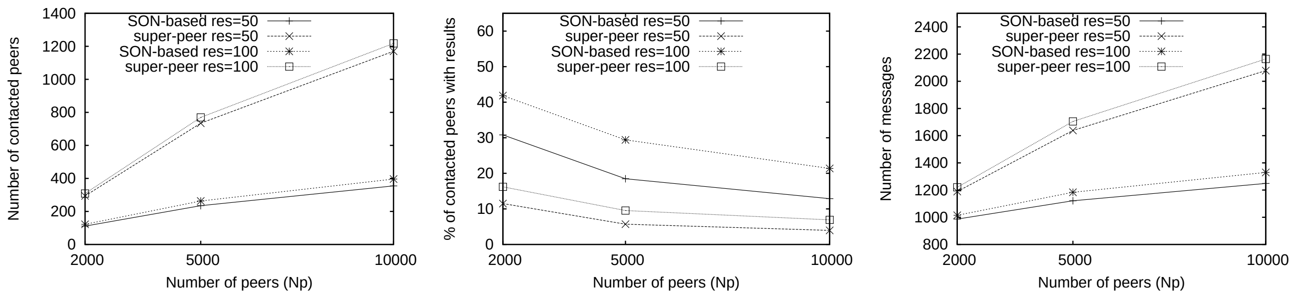
In Fig. 5, we study the scalability of our approach while increasing the dataset dimensionality from d=3 to d=7.

[4] SON stands for semantic overlay networks, which are peer groups with similar contents and they are generated by DESENT.



(a) Number of contacted peers. (b) % of contacted peers with results. (c) Number of messages.

Fig. 5 Scalability with dimensionality (d), N_{sp}=200, N_p=2000, n=1M, k=5, S_Z=15, n/N_p=500.



(a) Number of contacted peers. (b) % of contacted peers with results. (c) Number of messages.

Fig. 6 Scalability with network size (N_p), N_{sp}=200, d=3, k=5, S_Z=20, n/N_p=500.

Our setup consists of a network of 2000 peers, 200 super-peers, 1M data objects, and we set the k-means parameter to k=5. In this series of experiments, we depict the results of each approach using different values of query selectivity (res), namely 50 and 100 results. As shown in the charts, the gain of our proposed method compared to the initial random super-peer assignment increases, when the number of query results increases too.

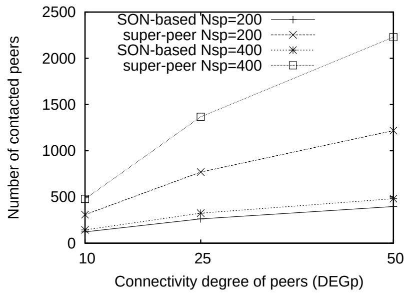
In Fig. 5(a), the SON-based approach manages to contact fewer peers than the initial random super-peer assignment, thus saving network costs and reducing response time. Moreover, in our approach, the number of contacted peers remains stable for higher dimensions, proving the scalability of our approach.

In Fig. 5(b), we measure the percentage of contacted peers that return results relevant to the query. The results show that in the SON-based approach, more than 30% of the contacted peers return relevant results, whereas the random assignment performs worse and its performance drops with increased dimensionality.

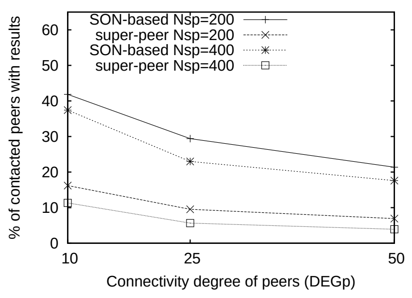
Furthermore, we also measure the number of exchanged messages in Fig. 5(c). Again our approach consumes less bandwidth, as fewer messages are required for searching. The SON-based approach achieves to reduce the number of messages and the savings range from 200-500 messages according to the dimensionality. Therefore, our approach improves the performance of similarity search on a super-peer network, and the improvement increases with the dimensionality, compared to a random peer to super-peer assignment.

6.1.2 Scalability with network size

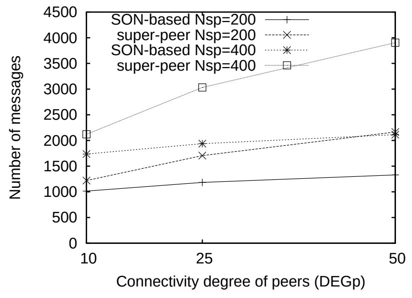
In the next set of experiments, we examine the proposed method's scaling features with regards to network size. In Fig. 6, we study the scalability of our approach with increasing number of peers in the network, while we keep the number of super-peers constant end equal to 200. We generate a clustered dataset of 3 dimensions, and we increase the dataset cardinality from n=1M to n=5M, so that n/N_p=500. Again, we vary also the query selectivity and we depict the results for number of results equal to 50 and 100. Fig. 6(a) clearly depicts that the number of contacted peers of the SON-based approach increase only slightly with the network size, while for the random super-peer assignment the number of contacted peers increases rapidly. Although our approach contacts fewer peers, Fig. 6(b) shows that a higher percentage of them are useful, i.e. they return results relevant to the query. The plot in Fig. 6(c) illustrates the number of message exchanged for processing the query, and it is obvious that the performance of our approach is more stable than the random super-peer assignment when the number of peers increases.



(a) Number of contacted peers.

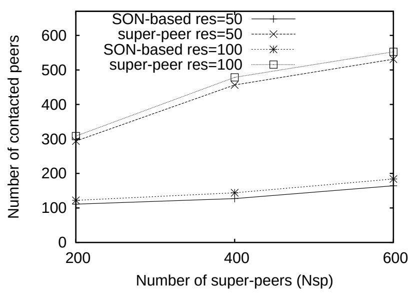


(b) % of contacted peers with results.

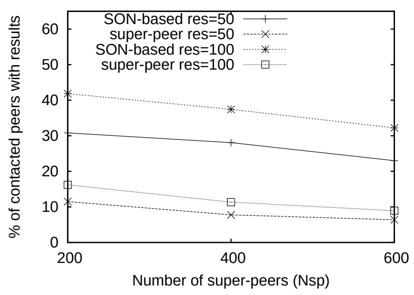


(c) Number of messages.

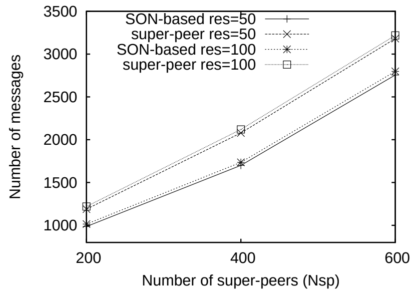
Fig. 7 Scalability with number of peers per super-peer (DEG_p), d=3, k=5, S_Z=15, n/N_p=500.



(a) Number of contacted peers.



(b) % of contacted peers with results.



(c) Number of messages.

Fig. 8 Scalability with number super-peers (N_{sp}), d=3, DEG_p=10, k=5, S_Z=15, n/N_p=500.

Summarizing, the results show that our approach consistently outperforms the random super-peer assignment, and moreover we achieve higher gains with increasing network size N_p. This is a strong fact in favor of the scalability of our approach, indicating that it works better than random super-peer assignment for larger P2P networks. In particular, for the largest setup that consists of 10000 peers, our approach contacts less than one third of the peers contacted by the random super-peer assignment.

6.1.3 Scalability with peer connectivity degree

Subsequently, we study the effect of varying the number of peers per super-peer. We fix res=100 and we show the results using two super-peer topologies of size N_{sp}=200 and N_{sp}=400, in Fig. 7. In terms of number of contacted peers, our approach is always better than the random super-peer assignment and presents a more stable performance as DEG_p increases (Fig. 7(a)). By inspecting the percentage of contacted peers with results, in Fig. 7(b), we see that the advantage of our approach is sustained for increased number of peers per super-peer, for both super-peer networks. Similar conclusions are drawn from Fig. 7(c), concerning the number of messages required for query processing.

6.1.4 Scalability with number of super-peers

Finally, in Fig. 8, we increase the number of super-peers available from 200 to 600, while keeping the number of peers per super-peer constant and equal to DEG_p=10. In all cases, our approach performs much better than the random super-peer assignment, showing that our approach scales gracefully with the number of super-peers. In particular, notice that in Fig. 8(a), the number of contacted peers for our approach essentially remains stable even when the number of super-peers increases. Summarizing, our approach always outperforms the random super-peer assignment and moreover it is a viable solution when the network size scales, where the performance of the random assignment degrades significantly.

7 Conclusions

In this paper, we propose an approach for generating a clustered super-peer overlay network, that improves the performance of query processing, since relevant data are indexed by only few super-peers. More particularly, we address the issue of changing the uniformity of data distribution at super-peer level in a self-organizing manner, so that peers with similar contents intentionally connect to the same super-peer. This improves the overall performance and increases the dependability of the P2P system. Our approach is particularly suitable for large-scale content distribution networks, where any centralized approach is prone to fail sooner or later, given the high rate of content generation. More importantly, the approach is based on unsupervised techniques, such as data clustering, and thus it requires practically no human intervention. The experimental results show that our approach outperforms the common case of random peer to super-peer assignment in all cases.

Acknowledgements This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme that supported Christos Doulkeridis and Akrivi Vlachou.

References

1. A. Avizienis, J.-C. Laprie, B. Randell, and C. E. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Sec. Comput.*, 1(1):11–33, 2004.
2. S. Baset and H. Schulzrinne. An analysis of the Skype peer-to-peer internet telephony protocol. In *Proceedings of INFOCOM*, 2006.
3. Y. Busnel and A.-M. Kermarrec. PROXSEM: Interest-based proximity measure to improve search efficiency in P2P systems. In *Proceedings of ECUMN*, 2007.
4. Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *Proceedings of SIGCOMM*, 2003.
5. V. Cholvi, P. Felber, and E. Biersack. Efficient search in unstructured peer-to-peer networks. In *Proceedings of SPAA*, pages 271–272, 2004.
6. E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer-to-peer networks: Harnessing latent semantics. In *Proceedings of INFOCOM*, 2003.
7. C. Doulkeridis, K. Nørvåg, and M. Vazirgiannis. Schema caching for improved XML query processing in P2P systems. In *Proceedings of IEEE P2P*, 2006.
8. C. Doulkeridis, K. Nørvåg, and M. Vazirgiannis. DESENT: Decentralized and distributed semantic overlay generation in P2P networks. *IEEE Journal on Selected Areas in Communications (J-SAC)*, 25(1):25–34, 2007.
9. C. Doulkeridis, A. Vlachou, Y. Kotidis, and M. Vazirgiannis. Peer-to-peer similarity search in metric spaces. In *Proceedings of VLDB*, pages 986–997, 2007.
10. P. Garbacki, D. H. J. Epema, and M. van Steen. Optimizing peer relationships in a super-peer network. In *Proceedings of ICDCS*, 2007.
11. A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza peer data management system. *IEEE Trans. Knowl. Data Eng.*, 16(7):787–798, 2004.
12. R. Huebsch, B. N. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The architecture of PIER: An internet-scale query processor. In *Proceedings of CIDR*, pages 28–43, 2005.
13. H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. *ACM Trans. Database Syst.*, 30(2):364–397, 2005.
14. G. P. Jesi, A. Montresor, and O. Babaoglu. Proximity-aware super-peer overlay topologies. *IEEE Transactions on Network and Service Management*, 4(2):74–83, 2007.
15. T. Karagiannis, A. Broido, M. Faloutsos, and Kc. claffy. Transport layer identification of P2P traffic. In *Proceedings of IMC*, pages 121–134, 2004.
16. I. Martinovic, C. Leng, F. A. Zdarsky, A. Mauthe, R. Steinmetz, and J. B. Schmitt. Self-protection in P2P networks: Choosing the right neighbourhood. In *Proceedings of IW-SOS/EuroNGI*, 2006.
17. L. Mathy, R. Canonico, S. Simpson, and D. Hutchison. Scalable adaptive hierarchical clustering. In *Proceedings of NETWORKING*, 2002.
18. D. A. Menascé and L. Kanchanapalli. Probabilistic scalable P2P resource location services. *SIGMETRICS Performance Evaluation Review*, 30(2):48–58, 2002.
19. K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *Proceedings of INFOCOM*, 2003.
20. S. Voulgaris, A.-M. Kermarrec, L. Massoulie, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *Proceedings of FTDCS*, pages 238–243, 2004.
21. S. Voulgaris, M. van Steen, and K. Iwanicki. Proactive gossip-based management of semantic overlay networks. *Concurrency and Computation: Practice and Experience*, 19(17):2299–2311, 2007.
22. B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of ICDE*, 2003.