DESENT: Decentralized and Distributed Semantic Overlay Generation in P2P Networks

Christos Doulkeridis, Kjetil Nørvåg, and Michalis Vazirgiannis

***Abstract*— The current approach in web searching, i.e., centralized search engines, rises issues that question their future applicability: 1) coverage and scalability, 2) freshness, and 3) information monopoly. Performing web search using a P2P architecture that consists of the actual web servers has the potential to tackle those issues above. In order to achieve the desired performance and scalability, as well as enhancing search quality relative to centralized search engines, semantic overlay networks (SONs) connecting peers storing semantically related information can be employed. The lack of global content/topology knowledge in a P2P system is the key challenge in forming SONs, and this paper describes an unsupervised approach for decentralized and distributed generation of SONs (DESENT). Through simulations and analytical cost modeling we verify our claims regarding performance, scalability, and quality.**

I. Introduction

The current approach in web searching, i.e., centralized search engines, rises issues that question their future applicability: 1) coverage and scalability, 2) freshness, and 3) information monopoly. Performing web search using a P2P architecture that consists of the actual web servers has the potential to tackle those issues above.

P2P architectures can be classified into *structured*, like Chord and CAN, and *unstructured* systems, like Gnutella and Kazaa. Although structured P2P systems have recently received a lot of attention because they can guarantee retrieval of existing documents and provide upper bound on retrieval cost (in a better way than unstructured systems), they have a number of limitations that make them less suitable for the task of Internet-scale web searching. For example, 1) peers indexing the most popular search term will easily become bottlenecks, 2) when a peer joins the network each term that should be indexed has to be sent to the appropriate peer, 3) when a peer leaves the terms it stores have to be reindexed, and 4) lack of support for efficient partial-match queries. These limitations do not occur in unstructured P2P systems. However, in order to make Internet-scale searching feasible, alternatives to the pure flooding-based search strategy have to be employed. Recently, the concept of *Semantic Overlay Networks* (SONs) [1] has been proposed as a solution to delve with this problem. The aim is to have peers storing similar documents in the same SON. If SONs have been created queries can be forwarded to only those sites containing documents that satisfy the constraints of the query context, thus reducing the communication cost of the query.

C. Doulkeridis is with the Dept. of Informatics, AUEB, Athens, Greece.
K. Nørvåg is with the Dept. of Computer Science, NTNU, Trondheim, Norway.
M. Vazirgiannis is with the Dept. of Informatics, AUEB, Athens, Greece.

One of the problems of SONs is the actual construction of these overlays, assuming the lack of knowledge of both global content and network topology. This is the main topic of our paper. In a P2P architecture each peer is initially aware only of its neighbors and their content. Thus finding other peers with similar contents, to form a SON, becomes a tedious problem. The contribution of this paper is a *decentralized and distributed* method for semantic overlay network construction (DESENT), that provides an efficient mechanism for web search in unstructured P2P networks. To the best of our knowledge, this is the first paper to deal with P2P web search using unstructured P2P networks. Our strategy for creating SONs is based on clustering peers based on their content similarity (henceforth the word cluster will be used to refer to a SON and vice-versa). This is achieved by a recursive process that starts on the individual web sites. By applying clustering on the documents stored at each site, one or more feature vectors are created for each web site, i.e., one for each topic a site covers. Then representative peers, each responsible for a number of peers in a *zone* are selected. These peers, henceforth called *initiators*, will collect the feature vectors from the members of the zone and use these as basis for the next level of clustering. This process is applied recursively, until we have a number of feature vectors covering all available documents.

The organization of the rest of this paper is as follows. In Section II we give an overview of related work. In Section III we present our method for creating SONs that can be used in the search process described in Section IV. In Section V we use analytical cost models to study the cost of creating overlays, while in Section VI we present results from simulations of a P2P network using SONs created by our algorithms. Finally, in Section VII, we conclude the paper.

II. Related Work

Semantic Overlay Networks (SONs) (similar to associative overlays [2]) have been proposed as an approach for semantically organizing peers, so that queries can be forwarded to only those peers containing documents within specific topics. In [1], SONs are presented as thematic focused groups of peers, which share common interests. A different notion of SONs [3] is related to schema mappings and peers that are logically interconnected through schema mappings.

Although several papers describe how to use SON-like structures, little work exists on the issue of to actually create SONs in an unsupervised, decentralized and distributed way in unstructured networks. Cholvi *et al.* [4] propose the use

of *acquaintances* as an extension to Gnutella-like networks to improve searching. A similar approach has been described in [5]. Other relevant approaches include gossiping algorithms [6]. The difference between our approach and gossiping approaches is that the connections in our approach ensure that relevant nodes will be clustered together. For large-scale P2P networks, gossiping approaches cannot guarantee that a remote peer that may contain relevant results will eventually be found.

Another approach to improve on some of the problems of unstructured P2P systems, is to use a super-peer architecture where a number of peers/clients are connected to a super-peer. An interesting study of super-peer networks is presented by Yang and Garcia-Molina [7]. Edutella [8] is another super-peer approach, where searching is achieved through routing at super-peer level. A super-peer architecture can also be used to realize a hierarchical summary index as described in [9].

Most approaches for P2P web search rely on the use of structured networks. In [10], the authors present MINERVA∞, a P2P web search engine that aims at providing scalability and efficiency. Previous approaches have focused on building global inverted indices, as for example the approach of Reynolds and Vahdat [11]. Most of these approaches are in general not applicable to very large networks due to the well-known problems related to building and maintaining a global index for terms in unreliable peers [12], intensified by the fact that construction and update costs are usually not taken into account.

In [13], a P2P architecture where nodes are logically organized into a fixed number of clusters is presented. The main focus is on fairness with respect to the load of individual nodes. In contrast to our approach, the creation of clusters/allocation of documents to clusters is done by classification, is not unsupervised, and clusters are not hierarchical.

This work is an extension of the work presented in [14]. It extends the previous paper by a more in-depth feasibility analysis, mature fault-tolerance algorithms, and more extensive experiments.

III. Overlay Network Creation

In this section we describe the SON generation process, assuming peers (for example web sites) storing documents and being connected in an unstructured P2P network. We refer to a *zone* as a set of peers in the same topological neighborhood. The *initiator* of a zone is the peer responsible for creating the zone and managing the zone's peers. A *cluster* is a set of peers that contain documents in the same topic(s). A *cluster representative* is a peer responsible for maintaining information about its cluster. Our approach is based on creating local zones of peers, forming semantic clusters based on local documents, and then merging zones and clusters recursively until global zones and clusters are obtained.

A. Decentralized and Distributed Overlay Creation

The peer clustering process is divided into 5 phases: 1) local clustering, 2) zone initiator selection, 3) zone creation, 4) intra-zone clustering, and 5) inter-zone clustering. The result is the creation of a hierarchy of zones and initiators (see Figure 1) and semantic clusters consisting of one or more peers.

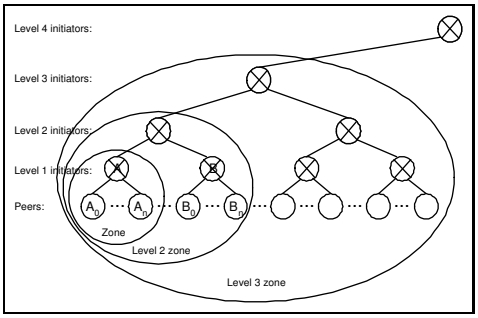


Fig. 1. Hierarchy of zones and initiators.

Local clustering on each peer is performed asynchronously in relation to other peers and is terminated before the global clustering process starts. Because the aim of the clustering process is to achieve a global result, it is beneficial to perform the subsequent phases at the same time at the different peers in the network. Achieving (or even assuming) a common global time for temporal synchronization is not feasible in a large P2P network, and fortunately not necessary. Our technique to cope with this problem is to use a partial synchronization technique, making only the assumption that each peer has a clock that is accurate within a certain amount of time t_a.

The global overlay network construction, henceforth also called *DESENT creation*, is assumed to start simultaneously at regular temporal intervals, at all peers. The length of the intervals (time between each DESENT creation) is part of the protocol, but can be as low as a few hours. Several other operations, like quasi-flooding, are also performed stepwise, one step at each synchronization point. The time interval between each synchronization point should be large enough to 1) ensure that the operation (in this case forwarding the message) can complete and 2) smooth the inaccuracies at the peers' clocks.

Phase 1: Local Clustering: Feature vectors are used instead of the actual documents because of the large amounts of data involved. A feature vector F_i is a vector of tuples, each tuple containing a feature (word) f_i and a weight w_i. However, even though a feature vector will be smaller than a document, having one feature vector for each document is still too much. This problem is solved by performing clustering of the document collection at each site. The result is a set of document clusters, and a feature vector representing each of the clusters. The feature vectors are created using a feature extraction process (see Section VI).

Phase 2: Initiator Selection: Assuming Z_i is the set of all peers in zone i, the zone consists of $|Z_i|$ peers, and one of these peers is given the role of *initiator*, which subsequently initiates and controls the clustering process within the zone. The process of choosing initiators is completely distributed and is performed at all peers concurrently (within the synchronization constraints as described above).

Because of load-balancing, the aim is to have as uniform zone sizes as possible, of approximately S_Z peers per zone. Assuming the IP of a peer P_i is IP_{P_i} and the time is T

(rounded to nearest t_a), a peer will discover that it is an initiator if $(IP_{P_i} + T)MODS_Z = 0$. The aim of the function is to select initiators that are uniformly spread out in the network (at approximately equal distance from each other) and an appropriate number of initiators relative to the total number of peers in the network (this is achieved by using the MOD, i.e., rest of division operator). By including time in the function we ensure that we obtain different initiators each time the clustering algorithm is run. This tackles the problem of being stuck with faulty initiators as well as reducing the problem of permanent cheaters.

It might happen that no initiator is selected by using the strategy described above, but this will be discovered from the fact that no *PROBE* is received within a certain time. In this case, a fallback mechanism is used: a universal decrease of the modulo-parameter is performed (by dividing by an appropriate prime number) in order to increase the chance of selecting (at least) one peer at the next iteration. This might happen more than once, until at least one peer starts zone creation. However, the maximum number of times is bounded because of the reduction of the modulo value.

Phase 3: Zone Creation: When initiators have been selected, the next step is to establish the zones around the initiators. At the end of Phase 2, the zone membership state of all initiator candidates is set to OCCUPIED, while the zone membership state for all other peers is initialized to NOT_OCCUPIED.

After a peer P_i has discovered that it is an initiator, it sends out a *PROBE* message to its immediate neighbors. Upon receiving a probe message, a peer P_j performs the following actions:

- If its state is NOT_OCCUPIED: 1) changes its state to OCCUPIED, 2) sends back to the initiator its identifier P_j, and 3) then forwards the probe to all neighboring peers except the peer from which it was received.
- If the peer's state is OCCUPIED: 1) the peer sends a *OCCUPIED* message to the *PROBE*'s initiator, as well as 2) a message to its own initiator in order to inform both initiators about each other's identities. In this way, each initiator will be able to become aware of its neighboring zones as well as the initiators in these zones. Note that this is only performed once by the peer for each new neighbor zone it detects.

The algorithm terminates when all peers have become members of a zone, i.e., all peers are in the OCCUPIED state. Obtaining global knowledge of termination would be very resource consuming. In order to avoid this, we use the knowledge that the radius of a zone is relatively small and assume the algorithm has terminated after rt_a time. The value of r should be large enough to cover non-uniform network topologies (considering the topology of real-world networks and zone sizes, this value can be relatively low). Unlike the flooding algorithms used for searching in P2P systems, this zone creation algorithm has a much lower cost, because a message will soon meet a neighbor zone and stop. Thus, the high cost of flooding involving a large number of peers is avoided. In the case of too large zones, the initiator can decide to partition its zone (i.e., a zone is split into two or more zones, so that each resulting zone has an appropriate size).

When this algorithm terminates, 1) each initiator has assembled a set of peers Z_i and their capabilities, in terms of resources they possess, 2) each peer knows the initiator responsible for its zone and 3) each initiator knows the identities of its neighboring initiators. An interesting characteristic of this algorithm is that it ensures that all peers in the network will be contacted, as long as they are connected to the network. This is essential for a P2P web search approach, otherwise there may exist peers whose content will never be retrieved.

Phase 4: Intra-zone Clustering: After the zones and their initiators have been determined, global clustering starts by collecting feature vectors from the peers and creating clusters based on these feature vectors:

1) The initiator of each zone i sends probe messages *FVecProbe* to all peers in Z_i.
2) When a peer P_i receives a *FVecProbe*, it sends its set of feature vectors $\{F\}$ to the initiator of the zone.
3) The initiator performs clustering on the received feature vectors. The result is a set of clusters $\{C_i\}$ represented by a new set of feature vectors $\{F_i\}$. A feature vector F_i consists of the top-k features of cluster C_i. Note that a peer can belong to more than one cluster.
4) The initiator selects a representative peer R_i for each cluster, based on resource information provided during Phase 3, like peer bandwidth, connectivity, etc.
5) The result kept at the initiator is a set of cluster descriptions (CDs), one for each cluster C_i. A CD consists of the cluster identifier C_i, a feature vector F_i, the set of peers $\{P\}$ belonging to the cluster, and the representative R_i of the cluster, i.e., $\text{CD}_i = (C_i, F_i, \{P\}, R_i)$.
6) Each of the representative peers are informed by the initiator about the assignment and receive a copy of the CDs (of *all* clusters in the zone). The representatives then inform peers on their cluster membership by sending them messages of the type (C_i, F_i, R_i).

Phase 5: Inter-zone Clustering: At this point, each initiator has identified the clusters in its zone. These clusters can be employed to reduce the cost and increase the quality of answers to queries involving the peers in one zone. However, in many cases peers in other zones will be able to provide more relevant responses to queries. Thus, we need to create an overlay that routes queries to clusters in remote zones. In order to achieve this, we recursively apply merging of zones to larger and larger super-zones, and at the same time merge clusters that are sufficiently similar into super-clusters: first a set of neighboring zones are combined to a super-zone, then neighboring super-zones are combined to a larger super-zone, etc. The result is illustrated in Fig. 1 as a hierarchy of zones and initiators. Note that level-i initiators are a subset of the level-$(i-1)$ initiators.

This creation of the inter-zone cluster overlay is performed as follows:

1) From the previous level of zone creation, each initiator maintains knowledge about its neighboring zones (and their initiators). Thus, the zones essentially form a zone-

to-zone network resembling the P2P network that was the starting point.

2) A level-i zone consists of a number of neighboring level-$(i-1)$ zones, on average $|SZ|$ in each (where SZ denotes a set of zones, and $|SZ|$ the number of zones in the set). This implies that $\frac{1}{|SZ|}$ of the level-$(i-1)$ initiators should be level-i initiators. This is achieved by using the same technique for initiator selection as described in Phase 2, except that in this case only initiators at level-$(i-1)$ in the previous phase are eligible for this role.
3) The level-i initiators create super-zones using the same algorithm as used in Phase 3. In the same way, these level-i initiators will become aware of their neighboring super-zones.
4) In a similar way to how feature vectors were collected during the basic clustering, the approximately $N_C^{i-1}|SZ|$ CDs created at the previous level are collected by the level-i initiator (where N_C^{i-1} denotes the number of clusters per initiator at the previous level). Clustering is performed again and a set of super-clusters is generated. Each of the newly formed super-clusters are represented by k features. A peer inside the super-cluster (not necessarily one of the representatives of the cluster) is chosen as representative for the super-cluster. The result is a new set of CDs, $\text{CD}_i = (C_i, F_i, \{P\}, R_i)$, where $\{\text{P}\}$ contains the representatives of the clusters forming the base of the new super-cluster.
5) The CDs are communicated to the appropriate representatives. The representatives of the merged clusters (the peers in $\{P\}$ in the new CDs) are informed about the merging by the super-cluster representative, so that all cluster representatives know about both their representatives *below* as well as the representative *above* in the hierarchy. Note that although the same information could be obtained by traversing the initiator/super-initiator hierarchy, the creation of super-cluster distributes the load more evenly and facilitates efficient searching.

This algorithm terminates when only one initiator is left, i.e., when an initiator has no neighbors. Unlike the initiators at the previous levels that performed clustering operations, the only purpose of the final initiator is to decide the level of the final hierarchy. The aim is to have at the top level a number of initiators that is large enough to provide load-balancing and resilience to failures, but at the same time low enough to keep the cost of exchanging clustering information between them during the overlay creation to a manageable level. The top-level peer probes level-wise down the tree in order to find the number of peers at each level until it reaches level j which has an appropriate number of peers. The level-j initiators are then informed about the decision and they are given the identifiers of the other initiators at that level, in order to send their CDs to them. Finally, all level-j initiators have knowledge about the clusters in zones covered by the other level-j initiators.

We emphasize that even though parts of this process resemble a centralized approach, this is not the case: initiators are chosen at random and perform their tasks completely independent of each other. Also, the role of the final peer in the hierarchy is only to determine that the global process is finished. As can be noted, initiators have similarities with super-peers, but one important difference is that their role is not constant.

B. Final Organization

To summarize, the result of the zone- and cluster-creation process described above are two hierarchies.

Hierarchy of peers: Starting with individual peers at the bottom layer, forming zones around the initiating peer which acts as a zone controller. Recursively neighboring zones form super-zones (see Fig. 1), finally ending up in a level where the top of the hierarchies have replicated the cluster information of the other initiators at that level. This is a forest of trees. The peers maintain the following information about the rest of the overlay network: 1) Each peer knows its initiator. 2) A level-1 initiator knows the peers in its zone as well as the level-2 initiator of the super-zone it is covered by. 3) A level-i initiator (for $i > 1$) knows the identifiers of the level-$(i-1)$ initiators of the zones that constitute the super-zone as well as the level-$(i+1)$ initiator of the super-zone it is covered by. 4) Each initiator knows all cluster representatives in its zone.

Hierarchy of clusters: Each peer is member of one or more clusters at the bottom level. Each cluster has one of its peers as representative. One or more cluster constitute a super-cluster, which again recursively form new super-clusters. At the top level a number of clusters exist. The peers store the following information about the cluster hierarchy: 1) Each peer knows the cluster(s) it is part of, and the representative peers of these clusters. 2) A representative also knows the identifiers of the peers in its cluster, as well as the identifier of the representative of the super cluster it belongs to. 3) A representative for a super-cluster knows the identifier of the representative at the layer above as well as the representatives of the layer below.

C. Fault-tolerance and Resilience

The number of failures inevitably increases with the number of peers being involved. In a P2P network peer failures can be relatively frequent, and in order to ensure that no peer in the hierarchy becomes a single point of failure or a bottleneck this issue has to be handled efficiently. Our main approach is to use k-replication of important overlay network data, i.e., hierarchy and cluster information. The replicated data is distributed on peers in a way that also distributes the tasks of the initiators over more peers.

In the DESENT overlay network it suffices to replicate the overlay-related information stored at the initiators. This data is replicated at $k-1$ other peers in the same zone. This replication is performed after the clustering process at level-i and before the creation of the level-$(i+1)$ zone. During creation of the level-$(i+1)$ zone the level-$(i+1)$ is informed about the replica peers.

In order to detect failures, the peers regularly send *alive* messages to the peers containing their replicas. If an *alive* message from a peer P_F is not received within a specified amount of time, the repair process is performed as follows:

1) The replica managers do a voting process in order to choose who is going to be the repair manager P_M that will perform the repair. There is also a possibility that missing *alive* messages only imply network problems rather than peer fault and have only been lost for one or a few of the peers. If this is discovered the repair process is interrupted.
2) A replacement peer P_R has to be found for P_F, and P_R is chosen from one of the other peers in the same zone. The identities of the candidate peers, i.e., the other peers in the zone, are known by P_M because they were part of the replica.
3) The replica data is sent to P_R, and P_R is now promoted to level-i initiator of a level-i super-zone. The other level-$(i-1)$ initiators in this zone are notified about the new initiator, the same is the case with the level-$(i+1)$ initiator. This notification will update the replicated data on these initiators, and the result is that their replicas have to be updated as well.

If a cluster representative that has no initiator role fails, this will be discovered when queries forwarded to it fail. Repair is in this case performed by the initiator, which simply selects a new representative from the peers in the cluster.

If a peer with no other responsibilities (i.e., a peer at the bottom level of the hierarchy) disappears this will be discovered by the respective cluster representative during query forwarding. When this occurs, the CD_i will be updated to reflect that P_F is not part of the cluster anymore. The zone initiator is also notified so that it updates the copy of the CD_i that it has stored.

From what is described above, we see that as long as fault-tolerance is handled for the initiators by k-replication, other repairs can be performed with no additional replication or monitoring.

Unfortunately, faults can also occur before the termination of the overlay network formation in a number of ways. Failure of a non-initiator peer can simply be ignored. If an initiator fails *before* the probe messages have been sent, the failure will not be detected by other peers, and the peers that should have been part of this initiator's zone are taken by other initiators. If an initiator fails *after* the probe messages have been received by at least one peer, the failure will be detected by timeout from the peers in the zone. They then select a new initiator. Failures in later stages of the overlay network creation process can be handled from the replicas as described above.

In addition to peer failures, faulty operation can also create a problem. One important case is timing failures, where peers either start the zone creation process too early, or try to flood more than one step at each synchronization point. If this is not detected, the result would be very large zones seized by a possibly faulty initiator. Luckily, this type of failure is easy to detect, because other peers will immediately discover peers performing operations asynchronously with the rest of the network.

In P2P systems cheaters are a possible problem. In DESENT a cheater is a danger mainly when having a controlling role in the zone- or cluster hierarchy. However, the fact that roles are not fixed means that a peer is only occasionally able to cheat, and is not able to do it synchronously with other cheaters (thus reducing the impact). Thus the incentive of even being able to function as a cheater will be small.

D. Peer Join

A peer P_J that joins the P2P network first establishes connection to one or more P2P neighbors as part of the basic P2P bootstrapping protocol (the actual protocol depend on the variant of unstructured P2P network, possible techniques include use of "known peers" as well as multicasting). These neighbors provide P_J with their zone initiators. Through one of these zone initiators P_J is able to reach one of the top-level nodes in the zone hierarchy and through a search downwards find the most appropriate lowest-level cluster which is will then subsequently join.

Note that no reclustering will be performed, so after a while a cluster description might no be accurate. However, the global clustering process is performed at regular intervals and will then create a new clustering that reflects also the contents of new nodes (as well as new documents that have changed individual peer's feature vectors). This strategy considerably reduces the maintenance cost in terms of communication bandwidth compared with incremental reclustering, and also avoid the significant computational cost that could be the result of continuous reclustering.

A peer can leave the network in two ways: 1) graceful departure where it notifies other relevant peers in the overlay network, or 2) leaving without notice, i.e., similar to a peer failure. In our system, both cases are treated similar to peer failure as described in detail in Section III-C. The only difference between the two is that in the case of a graceful departure a *takeover* message is sent to one of the peers containing the replica of its overlay network data, while in the latter case this process does not start until the failure is detected.

IV. Searching

When web search is performed, it is common that more than one documents match the query. In our context, the aim is to direct a query Q to the cluster(s) that are most relevant for the query with respect to query terms Q_T. A query originates from one peer P, and it is continually expanded until satisfactory results, in terms of number and quality, have been generated. All results that are found as the query is propagated are returned to the query originator P. Query processing can terminate at any of the steps below, if the result is satisfactory. A query is distributed as described below:

Q is sent to one of the top-level initiators (remember that each of the top-level initiators knows all top-level clusters). The most similar top-level cluster is determined, and Q is forwarded to its representative. Next, Q is routed down the cluster hierarchy, until the query is actually evaluated at the peers in a lowest-level cluster. The path is chosen based on highest similarity ($sim(Q, C_i)$) of the actual sub-clusters of a level-i cluster. If the number of results is insufficient, then backtracking is performed, in order to extend the query to more clusters. In the experiments reported later in this paper

the aim is to get as high recall as possible, and in this case the backtracking results in searching all forests that have sufficient similarity with the query. It should be noted that in practice a web search is satisfied by only finding the most relevant results, thus having a much lower cost.

V. Feasibility Analysis

In this section we will use cost models to study the feasibility of applying DESENT in a real-world P2P system. We will concentrate on two issues: 1) communication cost of DESENT creation, and 2) time needed to create DESENT. The parameters and default values used in the cost models are summarized in Table I. These values have been chosen based on appropriateness (typically size and performance as verified by simulations) or based on observed values from our simulations that will be described in mroe detail in Section VI.

A. Cost of Overlay Network Creation

A very important concern is the burden the DESENT creation imposes on participating nodes. We assume that communication cost will be the possible bottleneck and hence the most relevant metric, and we consider the cost of creating DESENT acceptable if the cost it imposes is relatively small compared to the ordinary document-delivery load on a web server.

In the local clustering and initiator selection phases there will be no communication, so the total cost C_T is essentially the cost of 1) performing zone creation for each level from 1 to the top level L, 2) performing intra-zone clustering recursively for each level from 1 to level $(L-1)$, followed by 3) distribution of clustering information to all level-$(L-1)$ peers (approximated to S_Z peers):
$C_T = \sum_{i=1}^{L} C_{ZC}(i) + \sum_{i=1}^{L-1} C_{IZ}(i) + N_F(N_F - 1)(N_C^i S_{CD} + S_M)$
where $C_{ZC}(i)$ denotes the cost of performing level-i zone creation and $C_{IZ}(i)$ denotes the cost of level-i intra/inter-zone clustering. Note that if the number of peers at level L is smaller than a certain threshold min_F, the peers at the level below are used as the forest of trees instead. In our study we set $min_F = S_Z/4$. Also note that the additional overhead incurred by network packet fragmentation for large messages is small compared to the actual payload, so this detail is omitted from the model.

In estimating the cost of zone creation, the most significant cost is the forwarding of the *PROBE* message from peers to its neighbors (all peers during this phase will receive one or more probes, but will only forward once, and not in the direction where the probe came from). Note that although this amounts to a relatively large number of messages during creation of the level 1 zones, most of the messages will be local and each peer will forward at most once. Other costs during zone creation include probe reply, and *OCCUPIED* messages from the border peers. The exact number of *OCCUPIED* messages is difficult to predict, but based on the experimental results we have found the number to be in the order of $N_{(i-1)}$, where $N_{(i-1)}$ denotes number of peers per zones at level $(i-1)$. Thus, the total cost of this phase is:
$C_{ZC}(i) = N_i D_{(i-1)} S_M + (N_{(i-1)} - N_i)(D_{(i-1)} - 1)S_M + (N_{(i-1)} - N_i)S_M + N_{(i-1)} S_M$

The cost of intra-zone clustering involves sending *FVecProbe* to all non-initiator peers, returning feature vectors to initiators, distributing the resulting CDs to the representatives, who resend them to the individual peers. We assume that each peer participates in approximately the same number of clusters as it originally provided, and the total cost of this phase is:
$C_{IZ}(i) = (N_{(i-1)} - N_i)S_M + (N_{(i-1)} - N_i)(N_C^0 S_{CD} + S_M) + N_i N_C^i (N_C^i S_{CD} + S_M)$
$+ (N_{(i-1)} - N_i)(N_C^0 S_{CD} + S_M)$

In studying the feasibility of DESENT, it is important that both the *average* communication cost for each peer is acceptable, as well as the *maximum* cost that can be incurred for a peer, i.e., the cost for the initiators on the top level of the hierarchy. The average communication cost can be calculated as $C_A = C_T/N_P$. In order to study the maximum cost for a particular peer to participate in the creation of the overlay network, both received and sent data should be counted because both pose a burden on the peer, i.e., $C_M = C_R + C_S$. Sent data include *PROBE*s in the zone creation phase, *FVecProbe*s and distributing the resulting CDs in the intra-zone clustering phase, and participation in the final exchange phase when being a root in the top-level forest:
$C_S = \sum_{i=1}^{L} D_{i-1} S_M + \sum_{i=1}^{L-1} \big((S_Z - 1)S_M + N_C^i (N_C^i S_{CD} + S_M)\big) + (N_F - 1)(N_C^i S_{CD} + S_M)$
Received data include probe replies and *OCCUPIED* messages in the zone creation phase, received CDs during intra-zone clustering, and participation in the final exchange phase when being a root in the top-level forest :
$C_R = \sum_{i=1}^{L} \big((S_Z - 1)S_M + S_Z S_M\big) + \sum_{i=1}^{L-1} \big((S_Z - 1)N_C^0 (S_{CD} + S_M)\big) + (N_F - 1)(N_C^i S_{CD} + S_M)$

Figure 2 illustrates C_A and C_M for different values of network size N_P and zone size S_Z. We see that in both cases a large zone size gives higher cost, but with very high variance. The situations in which this happens, is when the number of top-level peers is just below the min_F threshold so that the level below will be used as top level instead. With a large zone size this level will contain a large number of peers, and the final exchange of clusters information between the roots of this forest will be expensive. However, in practice this could be solved by merging of zones at this level. If we consider a zone size of $S_Z = 100$, we see that the maximum cost is just above 100 MB. If we compare this value with the load of a typical web server which is some GB of delivered documents per day [1], this is acceptable even in the case of daily reclustering. However, considering the fact that the role of the upper-level initiators changes every time the overlay network is created, it could even be feasible to perform this clustering more often.

In addition to this cost, there will also be a certain cost for maintaining replicas and peer dynamics in the network. However, this cost will be relatively small compared to the upper-level exchange of CDs.

[1] Using on of the web servers at in our department as example, it delivers in the order of 4 GB per day.

	Parameter	Default Value			Parameter	Default Value
B	Minimum bandwidth available	1 KB/s		N_i	# of peers/zones at level i	$\frac{N_P}{(S_Z)^i}$
D_0	Avg. # of neighbors at level 0	4		N_P	Total # of peers in network	1000000
D_i	Avg. # of neighbors at level i	S_Z		r	Max zone radius	20
L	# of initiator levels	$\lfloor \log_{S_Z} N_P \rfloor$		S_{CD}	Size of a CD	$\approx 1.5 S_F$
min_F	Min. # of trees in top-level forest	$S_Z/4$		S_F	Size of feature vector	200 B
N_C^0	# of clusters per peer	10		S_M	Size of packet overhead	60 B
N_C^i	# of clusters per level-i initiator	100		S_Z	Avg. zone size	100
N_F	# of trees in top-level forest	$> S_Z/4$		t_a	Time between synch. points	60 s

TABLE I

PARAMETERS AND DEFAULT VALUES USED IN THE COST MODELS.

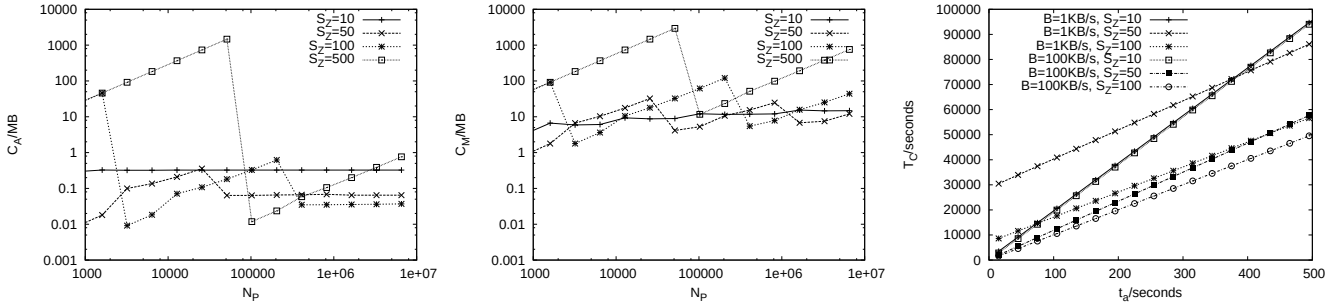


Fig. 2. Average cost (left) and maximum cost (middle) of participation in overlay network creation for different values of network size N_P and zone size S_S. (right) Time to create DESENT as a function of t_a for different zone sizes and bandwidths.

B. Construction Time

In order to provide freshness it is important that the duration of the DESENT creation itself is not too long. We will now show that the time needed to complete a cycle is relatively short.

Local clustering is assumed to be performed asynchronously to the DESENT construction (as noted previously the local clustering does not have to be performed each time DESENT is run), so the total time is then the time it takes from initiator selection and until the cluster exchanging in the last phase has been finished (assuming the time to send a message is insignificant compared to t_a, the time between synchronization points):

- In the case that at least one initiator exists, the initial zone creation phase takes r steps (cf. Section III-A), i.e., $T = rt_a$. However, it is possible that the fallback mechanism of initiator selection (modulo reduction) will be used, which adds a number of synchronization points to the time. This depends on the reduction value v_m (which we assume will be relatively small, for example 2). The maximum steps needed to reach the value 1 (when all IPs will match): $\log_{v_m} S_Z$. Thus, the total time is $T_Z = (r + log_{v_m} S_Z)t_a$.
- During intra-zone clustering, the most time-consuming task will in general be the clustering of feature vectors. Because next phase should not start until it can be guaranteed that all initiators have completed the intra-zone clustering, a certain number of time periods have to be allocated for this task. We denote this number v_c (thus allowing the time $v_c t_a$).
- Inter-zone clustering is a recursive process, performed $(\log_{S_Z} N_P - 1)$ times. With respect to time usage, each step is essentially similar to the zone-creation and intra-zone clustering described above, i.e. the cost is $T_{IZC} = (\log_{S_Z} N_P - 1)(T_Z + v_c t_a)$
- The final step is the exchanging of CDs by the "roots of the forest". The significant time cost here is the sending and receiving of the CDs. The total amount of data each has to send is approximately $C = 2(N_F - 1)(N_C^i(S_{CD} + S_M))$. This is communication between nodes that in general are geographically far away from each other and the bandwidth B (in bytes per second) will be relatively slow. A typical values can be in the order of 100 KB/s (measured between Athens and Tokyo, and between Trondheim and Beijing). The total time of this phase is C/B.

Thus, the total number of time needed for construction of the DESENT overlay network is:

$T_C = (r + log_{v_m} S_Z)t_a + (\log_{S_Z} N_P - 1)(r + log_{v_m} S_Z + v_c)t_a + 2(N_F - 1)(N_C^i(S_{CD} + S_M))/B$

Assuming the value of parameters as summarized in Table I, the time needed to construct the DESENT overlay network is $T_C = 13524$ seconds, i.e., approximately 3.75 hours. This means that the DESENT creation could run more than once a day if desired. An important point is that even if the construction takes a certain time, the average load the construction imposes and peers and communication will be relatively low. Most of the time is used to ensure that events are synchronized without having to use communication for this purpose. Regarding values of parameters, note that the actual *number of peers* has only minimal impact on the construction

time. For example, assuming $N_P = 10000$ instead of $N_P = 1000000$ gives $T_C = 11178$. The important parameters are t, S_Z, and B, and Figure 2 (right) shows the time needed to create DESENT for different parameter values. Essentially, the construction time increases linearly with t_a.

VI. DESENT Simulation Results

We have developed a simulation environment in Java, which covers all intermediate phases of the overlay network generation (see Section III), as well as the searching part (described in Section IV). We run all our experiments on Pentium IV computers with 3GHz processors and 1-2GB of RAM.

At initialization of the P2P network, a topology of N_P interconnected peers is created. We used the GT-ITM topology generator [2] to create random graphs of peers (we also used power-law topologies with the same results, due to the fact that the underlying topology only affects the zone creation phase), and our own synthetic topology (called SQUARE), which is similar to GT-ITM, only the connectivity degree is constant and neighboring peers share 3-5 common neighbors, i.e., the network is more dense than GT-ITM. A collection of N_D documents is distributed to peers, so that each peer retains N_D/N_P distinct documents. Every peer runs a clustering algorithm on its local documents resulting in a set of initial clusters.

In our experiments we used the Reuters-21578 text categorization test collection [3], and we studied two setups: a) 8000 pre-classified documents that belong to 60 distinct categories and b) 20000 documents. We examined two experimental setups: a) 8000 peers and b) 20000 peers. We then performed feature extraction (tokenization, stemming, stop-word removal and finally keeping the top-k features based on their TF/IDF value [15] and kept a feature vector of top-k features for each document as a compact document description). Initiators retrieve the feature vectors of all peers within their zone, in order to execute intra-zone clustering. We used hierarchical agglomerative clustering (HAC) to create clusters of documents. Clustering of documents is based on computing similarities and merging together feature vectors, by taking the union of the clusters' features and keeping the top-k features with higher TF/IDF values. We used the cosine similarity with parameter the similarity threshold T_s for merging. Clusters are created by grouping together documents and each cluster is also represented by a top-k feature vector. Obviously, other clustering algorithms as well as other similarity measures can be used.

A. Zone Creation

At first, we studied the average zone size after the zone creation phase at level 1 (see Figure 3). The network topology consists of $N_P = 20000$ peers, each having 10 neighbors on average and $S_Z = 100$. We run the experiment with and without zone partitioning. The y-value of a point on the chart (or histogram) is the average number of zones having size equal to the x-value of the point. The chart confirms our intuition that zone partitioning keeps all zones smaller than S_Z, while most are of sizes $50 - 100$. However, without zone partitioning, about 30% of the total zones have sizes greater than S_Z, and some are twice larger than S_Z, thus imposing a cumbersome load on several initiators. We also run the same experiment for networks of $N_P = 8000$ peers and $S_Z = 20$ and $S_Z = 100$, and drew the same conclusions.

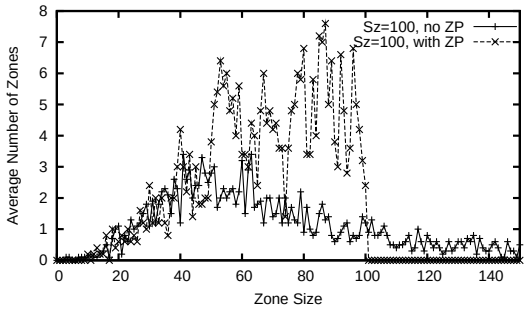


Fig. 3. Effect of zone partitioning during zone creation phase for average zone size $S_Z = 100$.

B. Clustering Results Quality

Measuring the quality of the DESENT clustering results is essential for the value of the approach. As clustering quality in our context, we define the similarity of the results of our clustering algorithm (C_i), with respect to an optimal clustering (K_j). We used in our experiments the F-measure [15] as a cluster quality measure. F-measure ranges between 0 and 1, with higher values corresponding to better clustering.

We compare the clustering quality of our approach to the centralized clustering results. The average values of DESENT F-measure relative to centralized clustering are illustrated in Fig 4(a), and show that DESENT achieves high clustering quality. Also note that the results exhibit a relatively stable behavior as the network size increases. This indicates that DESENT scales well with the number of participating peers. This conveys that the proposed system achieves high quality in forming SONs despite of the lack of global knowledge and the high distribution of the content.

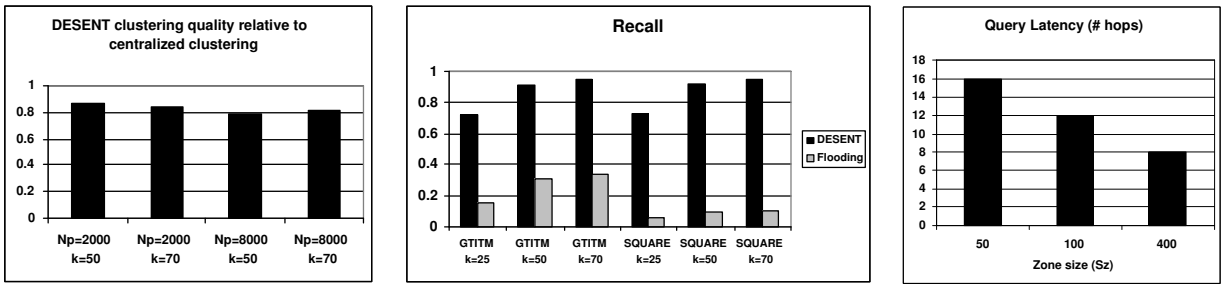
C. Quality and Cost of Searching

In order to study the quality of searching in DESENT, we consider as baseline the search that retrieves all documents that contain all keywords in a query. We measure the searching quality using recall, representing the percentage of the relevant documents found. Note that, for the assumed baseline, precision will always be 100% in our approach, since the returned documents will always be relevant, due to the exact matching of all keywords. We generated a synthetic query workload consisting of queries with term count average 2.0 and standard deviation 1.0. We selected query terms from the documents randomly (ignoring terms with frequency less than 1%). The querying peer was selected randomly.

In Fig. 4(b), we show the average recall of our approach compared to normalized flooding using the same number of

[2] http://www.cc.gatech.edu/projects/gtitm/

[3] http://www.daviddlewis.com/resources/testcollections/reuters21578/



(a) Clustering quality compared to centralized clustering for different network sizes and values of k.

(b) Average recall compared to normalized flooding using the same number of messages.

(c) Query latency (number of hops) for different zone sizes ($N_P = 20000$).

Fig. 4. DESENT simulation results measuring clustering quality relative to centralized clustering, achieved recall and query latency.

messages for different values of k, for the GT-ITM topology and the SQUARE topology for 8000 peers. Normalized flooding [16] is a variation of naive flooding that is widely used in practice, in which each peer forwards a query to d neighbors, instead of all neighbors, where d is usually the minimum connectivity degree of any peer in the network. The chart shows that with the same number of messages, our approach improves recall by more than 3-5 times for GT-ITM, and more than 10 for SQUARE, compared to normalized flooding. Furthermore, the absolute recall values increase with k, since more queries can match the enriched (with more features) cluster descriptions. Also notice that our approach presents the same recall independent of the underlying network topology, whereas flooding is sensitive to the topology. For example, when the topology is dense, the cost of flooding increases significantly, in other words recall is reduced using the same number of messages.

Another important measure is query latency, defined as the number of hops necessary to distribute the query and return the first result. Figure 4(c) illustrates the latency for a network of 20000 peers, which serves as an indication of the time a query is issued until the first result is obtained. The chart shows that the hierarchy is very efficient when it comes to latency, i.e., the first results will arrive very fast. Another observation is that increasing zone sizes result in lower latency, due to having smaller number of levels in the hierarchy.

VII. Conclusions and Further Work

In this paper, we have presented algorithms for distributed and decentralized construction of hierarchical SONs in unstructured P2P networks, achieved by distributed clustering of peer contents in a recursive way. Although we use Internet-scale search as the motivation and scalability goal of this effort, it should be noted that this approach is equally applicable at a smaller scale, including enterprise-wide information search. Future work includes performance and quality measurement of the search algorithm using large document collections (TREC, web crawls, etc), studying the use of other clustering algorithms, use of caching techniques to reduce search cost and increase efficiency, as well as integrate ranking into the system.

References

[1] A. Crespo and H. Garcia-Molina, "Semantic Overlay Networks for P2P Systems," Stanford University, Tech. Rep., 2002.

[2] E. Cohen et al., "Associative search in peer to peer networks: Harnessing latent semantics." in *INFOCOM*, 2003.

[3] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt, "Gridvine: Building Internet-Scale Semantic Overlay Networks," in *Proceedings of International Conference on Semantic Web (ISWC'2004)*, 2004.

[4] V. Cholvi, P. Felber, and E. W. Biersack, "Efficient search in unstructured peer-to-peer networks." in *Proceedings of the Sixteenth Annual ACM Symposium on Parallel Algorithms*, 2004.

[5] J. X. Parreira, S. Michel, and G. Weikum, "p2pDating: Real Life Inspired Semantic Overlay Networks for Web Search," in *Proceedings of SIGIR'2005 HDIR Workshop*, 2005.

[6] C. Tempich, S. Staab, and A. Wranik, "REMINDIN': Semantic Query Routing in Peer-to-Peer Networks based on Social Metaphors," in *Proceedings of WWW'2004*, 2004.

[7] B. Yang and H. Garcia-Molina, "Designing a Super-Peer Network," in *Proceedings ICDE'03*, 2003.

[8] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Loeser, "Super-Peer-based Routing and Clustering Strategies for RDF-based P2P Networks," in *Proceedings of WWW'03*, 2003.

[9] H. T. Shen, Y. Shu, and B. Yu, "Efficient semantic-based content search in P2P network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 813–826, 2004.

[10] S. Michel, P. Triantafillou, and G. Weikum, "MINERVA Infinity: A Scalable Efficient Peer-to-Peer Search Engine," in *Proceedings of Middleware'05*, 2005.

[11] P. Reynolds and A. Vahdat, "Efficient Peer-to-Peer Keyword Searching," in *Proceedings of Middleware*, 2003.

[12] J. Li, B. Loo, J.M.Hellerstein, M. Kaashoek, D. Karger, and R. Morris, "On the feasibility of peer-to-peer web indexing and search," in *Proceedings of the 2nd IPTPS'03*, 2003.

[13] P. Triantafillou, C. Xiruhaki, M. Koubarakis, and N. Ntarmos, "Towards High Performance Peer-to-Peer Content and Resource Sharing Systems," in *Proceedings of CIDR'03*, 2003.

[14] C. Doulkeridis, K. Nørvåg, and M. Vazirgiannis, "Scalable semantic overlay generation for P2P-based digital libraries," in *Proceedings of ECDL'06*, 2006.

[15] S. Chakrabarti, *Mining the Web - Discovering Knowledge from Hypertext Data*. Morgan Kaufmann Publishers, 2003.

[16] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes for unstructured peer-to-peer networks," in *Proceedings of INFOCOM'05*, 2005.