

SDC: A Distributed Clustering Protocol for Peer-to-Peer Networks

Yan Li, Li Lao, Jun-Hong Cui

yan.li@uconn.edu, llao@cs.ucla.edu, jcui@cse.uconn.edu

UCONN CSE Technical Report: UbiNet-TR06-02

Last Update: February 2006

Abstract

Network clustering can help to facilitate data discovery and peer-lookup in peer-to-peer systems. In this paper, we design a distributed network clustering protocol, called SCM-based Distributed Clustering (SDC), for peer-to-peer networks. In this protocol, clustering is dynamically adjusted based on Scaled Coverage Measure (SCM), a practical clustering accuracy measure. By exchanging messages with neighbors, peers can dynamically join or leave a cluster so that the clustering accuracy of the whole network is improved. SDC is a fully distributed protocol which requires only neighbor information, and it can handle node dynamics locally with very small message overhead while keeping good quality of clustering. Through extensive simulations, we demonstrate that SDC is able to discover good quality clusters very efficiently.

I. INTRODUCTION

Clustering is a very important method for data analysis. It is widely studied in various areas, such as biology, chemistry, linguistics, physics, and sociology. The basic goal of clustering is to group data into classes, with data in each class sharing certain similarity. In this paper, we study one interesting type of clustering: *graph clustering* or *network clustering*. It provides a way to partition a network topology into clusters such as nodes in the same cluster are highly connected and between clusters they are sparsely connected.

Network clustering has become an important technique in networking research. For instance, by taking advantage of network clustering features, we can design scalable and efficient routing protocols [10] [11] [15]. Another good example is network modelling. Since the Internet itself has clustering structure [5] [8], how to well characterize its clustering features is one critical step in topology modelling.

Network clustering can be performed in both centralized and distributed ways. Centralized network clustering is an off-line procedure, in which complete network topology information is needed before clustering. In our work, we focus on the latter one. We are interested in the network clustering of large-scale peer-to-peer systems.

In peer-to-peer networks, there are usually large number of peers and the knowledge of each peer about the network topology is mostly limited to its immediate neighbors. Due to the large scale and the lack of knowledge about the complete network structure in each peer, a main challenge in peer-to-peer system design is to perform

data discovery and peer look-up in a scalable and efficient way. The network clustering technique can significantly facilitate these operations. We give two examples as follows.

Let us first consider the Gnutella file sharing system [1]. In Gnutella, peers form an unstructured overlay network in which queries are routed through flooding. Obviously, routing in Gnutella is not scalable as queries are broadcast to every peer, leading to a huge amount of traffic. If flooding is curtailed, the message overhead can be reduced, but the system may fail to locate a file which is actually being shared. By utilizing network clustering, we can reduce message overhead while ensuring accurate searching results. Suppose a peer-to-peer network is clustered. Then for each peer, besides neighbor information, it can also maintain its cluster information (e.g., some information about other peers in the same cluster). When a peer issues a search query, a small-scale flooding can be first performed inside its cluster. Then only a small portion of peers (which are connected to other clusters) need to continue forwarding the query. In this way, the amount of exchanged messages can be reduced significantly.

Network clustering is also very useful in hierarchical peer-to-peer system design. In a hierarchical P2P system (such as KaZaa [2]), peers are classified into normal peers and super peers. Normal peers together with super peers form various clusters. And only super peers are responsible for the communication between clusters. Clearly, small scale clusters introduces much lower management overhead compared with the whole network.

Besides the above examples, network clustering has been widely utilized to address certain key issues in various P2P systems [12] [3] [14] [7]. Then, the challenge is how to partition a peer-to-peer network in an efficient and distributed manner, i.e., to design an effective distributed clustering protocol. This is the central problem we want to solve in this paper.

There are several characteristics of a good distributed clustering protocol. First of all, as a natural requirement of network clustering, nodes in the same clusters should be highly connected, and less connected between clusters. Secondly, the protocol should well control the cluster size (or cluster diameter). Thirdly, the protocol should result in a minimum number of “orphan” nodes. Lastly, a good clustering protocol should take node dynamics into account, since the target networks (peer-to-peer networks) are highly dynamic with frequent entry and exit of nodes. We will have a detailed discussion on the criteria of clustering in Section II.

In the literature, there have been considerable research efforts addressing the problem of network clustering, but very few of them studied the problem of clustering in peer-to-peer networks. Among the existing approaches, MCL [18] is well accepted as an efficient and accurate network clustering algorithm. However, this approach assumes that the complete network topology is available at one central point, which is not realistic in peer-to-peer systems. CDC [17], on the other hand, is a distributed algorithm. It forms clusters based on node connectivity. The main issue with this algorithm is that it can not handle node dynamics in a decent way, as limits its utility in peer-to-peer networks.

With these problems in mind, we design a novel network clustering protocol called **SCM-based Distributed Clustering (SDC)**, which satisfies all the design criteria discussed above. In this protocol, clustering is dynamically adjusted based on Scaled Coverage Measure (SCM) [19], a practical clustering accuracy measure. By exchanging

messages with neighbors, peers can dynamically join or leave a cluster so that the clustering accuracy of the whole network is improved. To control the cluster size, TTL (Time-To-Live) is piggybacked in exchange messages to guarantee the cluster diameter will never exceed a predefined threshold. SDC is a fully distributed protocol which requires only neighbor information, and it can handle node dynamics locally with very small message overhead while keeping good quality of clustering. Through extensive simulations, we demonstrate that our proposed protocol, SDC, is able to discover good quality clusters in a very efficient way.

The rest of this paper is structured as follows. In Section II, we review some background information and related work. In Section III, we discuss the important concept, SCM. Then in Section IV, we present our clustering protocol SDC. Following that, we show the performance of SDC by extensive simulation results in Section V. Finally, we conclude the paper in Section VI.

II. BACKGROUND AND RELATED WORK

In this section, we first present the network model (used in this paper) and the criteria of network clustering. Then we review some related techniques.

A. Network Model

We assume each peer-to-peer network is represented by a connected, undirected graph $G = (V, E)$, where V is the set of nodes corresponding to the set of peers in the system and E is the set of links, which are the logical connections between peers. We denote $|V| = n$ and $|E| = m$. Then the partition $\mathcal{C} = \{C_1, C_2, \dots, C_l\}$ of V is called a *clustering* \mathcal{C} of graph G , and C_i s are called *clusters*. Each cluster should be a non-empty subset of V . Obviously, $\bigcup_{i=1}^l C_i = V$. The *diameter* of a cluster C_i is defined as the maximum length of the shortest paths among all pairs of nodes in C_i . Then if a cluster has only one node, it has a diameter of 0. We call the clusters with diameter 0 as *orphan nodes*. In this paper, we also define *cluster size* as the number of nodes in a cluster to represent the cluster scope. Clearly, cluster size and cluster diameter are closely related. In most context, “control cluster size” and “control cluster diameter” have the same meaning of “control cluster granularity”. We only differentiate these two concepts in the protocol description.

B. Criteria of Clustering

The network clustering problem is essentially to discover a “good” clustering \mathcal{C} of graph G with the goal of correctly identifying the existing clustering substructures. More specifically, in a “good” partition \mathcal{C} , the intra-cluster node connectivity should be maximized and the inter-cluster node connectivity should be minimized. Therefore, node connectivity is a basic criterion to be considered in network clustering. In addition to node connectivity, cluster size is taken as another important metric. In large-scale distributed networks, due to the lack of knowledge about the network structure, it is expensive to maintain expanded clusters, In other words, the cluster diameters should

be carefully controlled. A good network clustering algorithm should also take orphan nodes into consideration. In most scenarios, orphan nodes are not preferred as they violate the goal of clustering and should be eliminated.

As discussed above, node connectivity, cluster diameter, and orphan nodes are important criteria for good network clustering algorithms. However, more issues need to be addressed when we consider the clustering in peer-to-peer networks. In such networks, each node only has the knowledge about its neighbors and can join or leave the network at any time. To obtain a complete view of the network structure, a huge number of messages need to be exchanged to collect the topology information. Moreover, the obtained topology may expire very soon due to the node dynamics. Recollecting topology information on each node-entry and node-exit will overload the network with a huge amount of exchanged messages. Therefore, it is impossible to maintain a complete and up-to-date network topology in such networks. Given these concerns, a good clustering protocol for peer-to-peer networks should be able to discover clusters in a fully distributed fashion, i.e., the peers themselves should form clusters automatically without the knowledge of the complete network topology, and it should be able to recover from node dynamics with small overhead in term of the amount of messages exchanged between peers.

C. Related Work

Researchers have done significant work on network clustering and its applications [4] [9] [11] [13] [18] [17]. However, most of the existing network clustering algorithms assume that the complete network topology is available at a central point. Among these algorithms, MCL [18] is considered as a scalable and efficient network clustering scheme. The basic idea behind this algorithm is flow simulation. In this algorithm, an input graph G is mapped in a generic way onto a Markov matrix. Then the set of transition probabilities are iteratively recomputed via matrix expansion and inflation. An infinite sequence consisting of repeated alternation of expansion and inflation constitutes a new algebraic process called the Markov Cluster (MCL) process. The heuristic behind this algorithm is that a flow between sparsely connected dense regions evaporates after MCL process. Therefore, it is easy to detect dense regions in the original graph, and the dense regions are returned as clusters.

MCL algorithm can achieve high clustering accuracy. However, due to its centralized feature, it can not be used in peer-to-peer networks.

Another interesting network clustering technique is CDC [17]. To the best of our knowledge, CDC is the only existing clustering algorithm for discovering connectivity based clusters in peer-to-peer networks. In CDC, a set of peers are selected as “originators” and clusters are discovered around these peers by controlled message flooding (TTL is used to control the flooding range). If the “originators” are well distributed in the network, clusters with good quality can be formed.

The CDC scheme is a distributed approach and the cluster size can be effectively controlled. One main issue in this scheme is the selection of “originators”. So far, there is no good solution which can guarantee good “originators”. Thus, the clustering accuracy may be affected significantly. Another issue in CDC is it can not efficiently handle node dynamics. To maintain good clustering quality, the whole network has to be re-clustered at each node join or

leave, which introduces a huge amount of message overhead.

In summary, there is no existing clustering algorithm which can satisfy all the criteria for network clustering in peer-to-peer systems. In this paper, we propose a novel network clustering protocol, SDC. This protocol is fully distributed and can form high quality clusters with very small message overhead.

III. SCALED COVERAGE MEASURE

Before introducing our protocol, we first discuss **Scaled Coverage Measure (SCM)**, a practical measure to evaluate the accuracy of connectivity based clustering proposed by S.Van Dagon [18].

We assume $\mathcal{C} = \{C_1, C_2, \dots, C_l\}$ is a clustering on network $G = (V, E)$. Given a node $v_i \in V$, we have the following notations:

- **Nbr**(v_i) is the set of neighbors of node v_i ;
- **Clust**(v_i) is the set of nodes in the same cluster as node v_i (excluding v_i);
- **FalsePos**(v_i, \mathcal{C}) is the set of nodes in the same cluster as v_i but not neighbors of v_i ;
- **FalseNeg**(v_i, \mathcal{C}) is the set of neighbors of v_i but not in the same cluster as v_i .

Then the *Scaled Coverage Measure* of node v_i , $SCM(v_i)$, is defined as:

$$1 - \frac{|FalsePos(v_i, \mathcal{C})| + |FalseNeg(v_i, \mathcal{C})|}{|Nbr(v_i) \cup Clust(v_i)|}. \quad (1)$$

For graph G , the SCM value, $SCM(G)$, is defined as the average of the SCM values of all the nodes, that is, $SCM(G) = (\sum_{v_i} SCM(v_i))/n$, which lies in $[0, 1]$.

SCM well reflects the significance of clustering features in a given network. First of all, it is easy to see that the higher the SCM, the smaller the connectivity between clusters and the higher the connectivity within clusters. For graphs containing only isolated clusters/subgraphs that are themselves fully connected, the SCM value is 1. Secondly, for any graph, there exists a highest SCM value which is determined solely by the network structure. If the network does not contain significant clustering substructures, this highest “available” SCM value can be very small. However, if we evaluate two clustering techniques on the same network, the one which results in a higher SCM value discovers more accurate clustering substructures than the one with smaller SCM value, although both resultant SCM values could be very small. Lastly, the SCM value of an orphan node is 0, which matches our goal of minimizing the number of orphan nodes.

Based on the definition of SCM, the network clustering problem can be simplified as partitioning a network topology so that its SCM is maximized. Our proposed SDC protocol exactly follows this idea, adaptively forming clusters in an aggressive manner.

Simplified Notations To simplify the computation in SDC, we can re-express SCM at node v_i as follows:

$$SCM(v_i) = 1 - \frac{b_{v_i}}{a_{v_i}}, \quad (2)$$

Thus,

$$b_{v_i} = |FalsePos(v_i, \mathcal{C})| + |FalseNeg(v_i, \mathcal{C})|,$$

and

$$a_{v_i} = | Nbr(v_i) \cup Clust(v_i) | .$$

If node v_i is an orphan node, $b_{v_i} = a_{v_i} = deg(v_i)$, where $deg(v_i)$ is the degree of node v_i . These two parameters b_{v_i} and a_{v_i} can be easily updated based on neighbor information upon node joining and leaving the cluster.

In the next section, we show how SCM is utilized in the SDC protocol to form clusters in a distributed fashion.

IV. THE SDC PROTOCOL

A. Protocol Description

The SDC protocol performs in a fully distributed way. Each node v_i only needs to maintain some basic information about its neighbors and the cluster it belongs to, such as the cluster id $clust_id$, the cluster size $clust_size$ (which is the total number of nodes in the cluster), and the two parameters for SCM computation b_{v_i} and a_{v_i} .

Given a network, each node v_i is initialized as an orphan node with its own $clust_id$ (any unique id is sufficient) and $clust_size$ (1 in this case). And the two parameters for SCM computation b_{v_i} and a_{v_i} are initialized as $deg(v_i)$. Then all nodes start to exchange messages with their neighbors, conduct some simple computation, and form clusters in a greedy manner. After a number of rounds of communication, the clustering procedure becomes stable without further message exchange and the network is finally clustered.

In SDC, we define a set of **Clust_** type of messages. Suppose node v_i wants to be clustered. The following clustering messages may be involved.

- **Clust_Probe.** Node v_i first sends the message *Clust_Probe* to every node $v_j \in Nbr(v_i)$ to find out other clusters in the neighborhood. Each node which receives *Clust_Probe* will send its $clust_id$ and $clust_size$ back to v_i .
- **Clust_Request.** Once receiving the $clust_ids$ from its neighbors, node v_i can determine its “neighbor clusters”. Suppose v_i discovers that a cluster Cl is connected with it, it issues a *Clust_Request* message which is flooded in Cl and v_i 's current cluster $Clust(v_i)$. This is a well-controlled flooding, since upon receiving *Clust_Request*, a node can forward this message to others only if it is in Cl or $Clust(v_i)$. For any node v_j in cluster Cl , upon receiving *Clust_Request*, a very simple computation is performed to obtain $\Delta SCM(v_j)$, the gain in $SCM(v_j)$ assuming node v_i joins Cl . This computation only requires the information of whether v_i is v_j 's neighbor or not. Specifically, if node v_j is a neighbor of node v_i ,

$$\Delta SCM(v_j) = \frac{1}{a_{v_j}}. \quad (3)$$

On the other hand, if v_j is a non-neighbor node of v_i , the $\Delta SCM(v_j)$ is given by:

$$\Delta SCM(v_j) = \frac{b_{v_j}}{a_{v_j}} - \frac{b_{v_j} + 1}{a_{v_j} + 1}. \quad (4)$$

Similarly, for any node $v_k \in Clust(v_i)$, it needs to compute $\Delta SCM(v_k)$ as if v_i leaves its current cluster.

To control the number of exchanged messages, a *TTL* is carried in *Clust_Request*. Once receiving *Clust_Request*, any node should check the *TTL* value first and will discard the message without forwarding to others if *TTL* expires. *TTL* is also used to control the cluster diameter.

- **Clust_Reply.** Upon receiving *Clust_Request* from v_i ($TTL \neq 0$), node v_j sends back a *Clust_Reply* message carrying $\Delta SCM(v_j)$ and v_j 's *clust_id* back to node v_i .
- **Clust_Reject.** Based on the *TTL* in *Clust_Request*, node $v_j \in Cl$ can determine whether or not the cluster diameter will exceed the predefined threshold due to the joining of node v_i . If this is the case ($TTL = 0$), v_j simply stops forwarding *Clust_Request* to other nodes and a *Clust_Reject* message will be sent back to v_i . Once receiving *Clust_Reject*, node v_i will not join *Cl*.
- **Clust_Update.** After node v_i receives *Clust_Reply* messages from all the nodes in its current cluster and the neighbor cluster *Cl* (in the case that no *Clust_Reject* is received from *Cl*), it computes the overall gain $\Delta SCM(G)$ based on the received information, assuming it leaves its original cluster and joins *Cl*. If we use Δ_{leave} as the overall gain in SCM as if node v_i leaves its original cluster and Δ_{join} as the gain in SCM assuming node v_i joins *Cl*, the overall gain can be computed as:

$$\Delta SCM(G) = \Delta_{join} + \Delta_{leave} \quad (5)$$

If $\Delta SCM > 0$, v_i should join *Cl*. There might be multiple clusters for which ΔSCM are positive, v_i should join the one with the maximum SCM gain. Once v_i determines which cluster to join, a *Clust_Update* message containing v_i 's node id and its original *clust_id* is flooded in its original cluster and the new cluster it will join. Then, v_i and any node receiving this message will update the *clust_size* and their own SCM.

- **Clust_Wait.** In the case that multiple nodes try to join the same cluster simultaneously, only one node can be served and all the other nodes should be “locked” until this node is done with its clustering. Whenever a node receives a *Clust_Request* from node v_i , while processing another request, it will send *Clust_Wait* message to node v_i . And v_i has to wait a predefined period of time before the next clustering attempt.

After node v_i joins the new cluster, its neighbors in the original cluster are affected and should check whether they should join other clusters, in the same way as node v_i does. The whole procedure will end if no node can join any cluster based on $\Delta SCM(G)$ and the cluster diameter control.

B. An Example

A simple example is shown in Fig. 1 to illustrate the SDC clustering procedure. In this example, *TTL* is set to 2, so the diameter of any cluster should not exceed 2. In Fig. 1a, node 0 wants to be clustered with other nodes. It first sends *Clust_Probe* messages to all of its neighbors. Each neighbor node upon receiving the *Clust_Probe* message sends its *clust_id* and *clust_size* back to node 0 as shown in Fig. 1b. After finding two clusters connected with it, clusters *A* and *B*, node 0 sends *Clust_Request* to these two clusters, which is shown in Fig. 1c. At the same time, node 7 also wants to be clustered and can send a *Clust_Request* to its neighbor 2. Since node 0 is being served,

node 7 should be “locked” and it has to wait a period of time before the next clustering attempt. In clusters *A* and *B*, every node which receives *Clust_Request* computes its SCM gain and then sends *Clust_Reply* back to node 0 (Fig. 1d). Node 0 then computes $\Delta SCM(G)$ based on the received information and joins Cluster *A* (Fig. 1e). Since node 4 is affected by node 0’s joining action, it starts a new clustering procedure in a similar way as node 0 does (Fig. 1f).

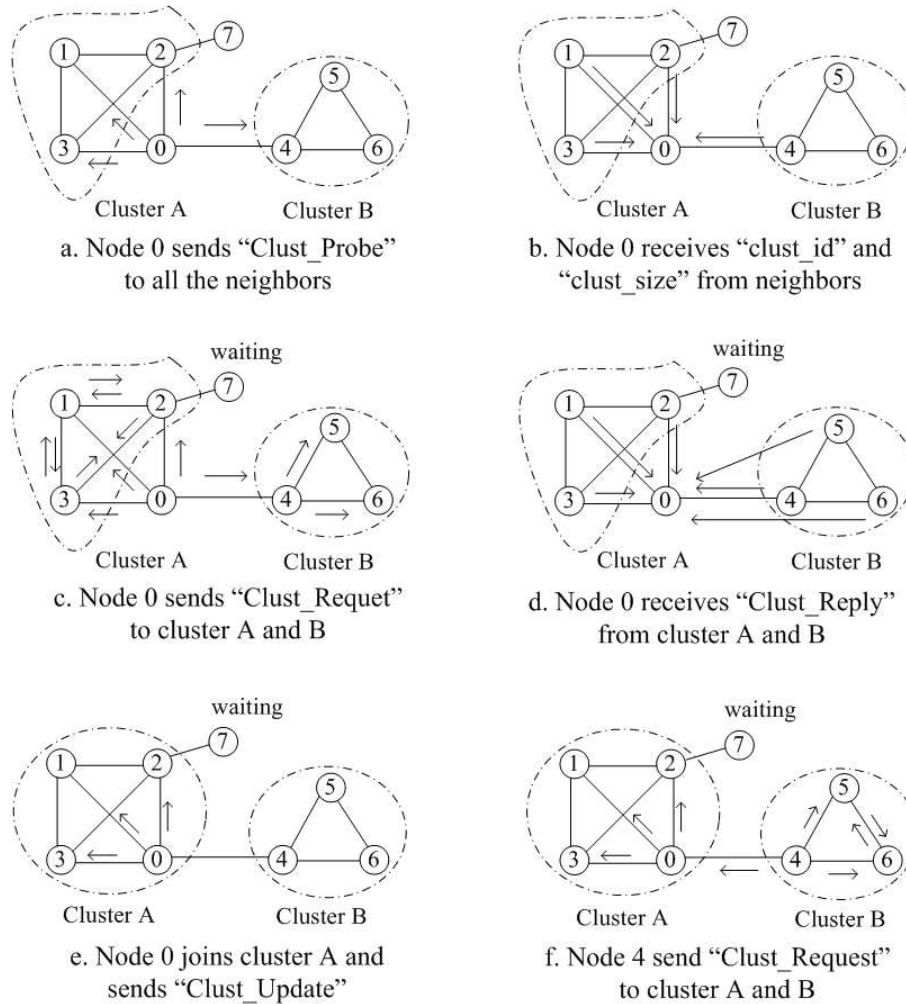


Fig. 1. A simple example of SDC protocol ($TTL = 2$).

C. Handling Node Dynamics

Peer-to-peer networks are dynamic systems. With node entry and exit at arbitrary points, the network structure is changed and the existing clusters are affected. Re-do the whole clustering procedure may keep good clustering accuracy. However, it is very inefficient and the procedure may never stabilize if node entry and exit happens frequently. Therefore, designing an effective and efficient scheme to handle node dynamics is a critical issue in the clustering of peer-to-peer networks.

Our SDC protocol can naturally handle node dynamics. Whenever a new node v_i joins the system, it is first initialized as an orphan node and gets its own *clust_id* and *clust_size* (which is 1). Since the network structure

between node v_i and its neighbors is changed, a **Join** message carrying v_i 's *clust_id* is issued by v_i to all of the neighbors so that they can update their SCM. As v_i 's joining changes its neighbors' connectivity, the affected neighbor nodes should perform a new round of clustering procedure. When a node wants to leave, it sends a **Leave** message to each of its neighbors as well as every other node in its cluster through flooding so that the *clust_size* and SCM values of the affected nodes can be updated. This will also activate a new round of clustering procedures at these affected nodes. The logic behind this scheme comes from the fact that node entry and exit are localized events and only a few nodes are affected and need to be re-clustered.

Some overhead is introduced when SDC handles node dynamics. Nevertheless, this overhead is very small since only neighbors and/or the nodes in the same cluster are directly affected. In next section, we will show that SDC can achieve very good clustering accuracy with low overhead in the presence of node dynamics. In contrast, CDC has to re-do the complete clustering procedure for any node join or leave in order to maintain good clustering accuracy, which introduces a lot of overhead.

V. SIMULATION EVALUATIONS

In this section, we conduct simulations to evaluate the performance of SDC, comparing it with the decentralized clustering scheme, CDC.

A. Experiment Settings

To test the applicability of our clustering protocol to different network structures, we use two types of topologies: random topologies from GT-ITM topology generator [6] and power-law topologies from the BRITE generator [16].

We implement both the SDC and CDC algorithms and run them on different topologies. There are several configurable parameters in the CDC scheme: *Vicinity*, *TwoHopThreshold*, *WeightThreshold* and *MinWeight*. These parameters are carefully tuned so that we can get the best clustering accuracy for CDC. Specifically, we set the values for *Vicinity*, *TwoHopThreshold*, *WeightThreshold*, *MinWeight* as: 1, 0.0005, 0.001, 0.00001 respectively. In both algorithms, TTL is used to control the cluster size. For all the results shown in this section, the initial TTL values are set to 3. The reason we select this value is due to the fact that the clustering accuracy of CDC is improved with the increase of TTL. When TTL is configured to 3, CDC achieves the comparable clustering accuracy as it does for higher TTL values while the message overhead is much smaller than the resultant overhead from higher TTLs.

The metrics used in our experiments are: *clustering accuracy*, *message overhead* and *cluster size*. We compute the clustering accuracy using SCM, and we measure the overhead in term of the number of exchanged messages between peers. These two metrics are tested in both static and dynamic systems. The cluster size is measured as the number of nodes in a cluster. Using this metric, we want to test the performance of SDC in controlling the cluster granularity. Although TTL can guarantee the cluster diameter not exceeding a predefined threshold, it can

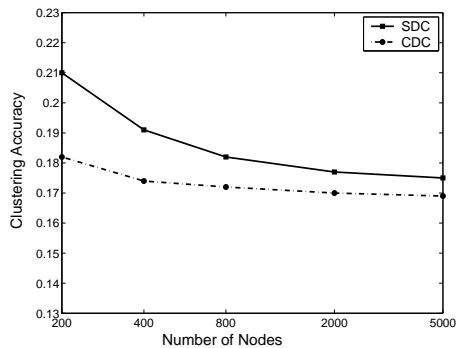


Fig. 2. Clustering accuracy in pure random topologies

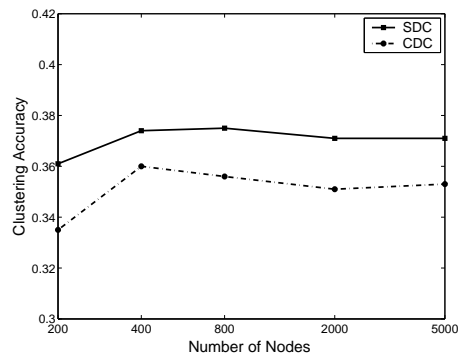


Fig. 3. Clustering accuracy in power-law topologies

not control the number of nodes in each cluster as there is no strict relationship between these two metrics. We also study the influence of node degree and TTL on the performance of SDC.

B. Performance of SDC in Static Systems

In this set of experiments, we evaluate the performance of SDC in static systems, i.e., there are no node dynamics. We want to test the effectiveness of SDC for different network structures.

We first conduct experiments on pure random topologies, which are generated by the GT-ITM topology generator. Nodes in this type of topologies have the same probability of being connected with other nodes. We fix the average degree as 10, and vary the topology size (i.e., the number of nodes) from 200 to 5000. We run each experiment more than 100 times so that all the results have a standard deviation of less than 0.1% of the average value. We plot the clustering accuracy of SDC and CDC in Fig. 2. This figure shows that the accuracy value of both algorithms slightly decreases as the network size is lifted. The rationale behind this is following: in pure random topologies, when the average node degree is fixed, as the network becomes bigger, nodes tend to evenly distributed, which results in less significant clustering features. When networks get very big, the influence of network size becomes minor. If we compare the performance of two algorithms, SDC is slightly better than CDC. This indicates that for a topology with uniformly distributed node degrees, the capacities of both algorithms in detecting accurate clusters are comparable.

Now let us examine the clustering accuracy of SDC for topologies with skewed degree distribution (i.e., power-law topologies). We fix the average node degree of each topology as 4, while the highest node degree is around 22. We plot the results in Fig. 3. It shows that SDC performs much better than CDC in power-law topologies. Moreover, the clustering accuracy of both algorithms are stable as the network size increases. This result tells us that the clustering features in power law-topologies is more dominated by the node degree distribution.

We also measure the number of exchange messages to evaluate the clustering overhead for both sets of topologies. We plot the results for each type of topologies in Fig. 4 and Fig. 5 respectively. For random topologies, Fig. 4 tells that with the increase of the network size, the message overhead in both algorithms is increasing significantly.

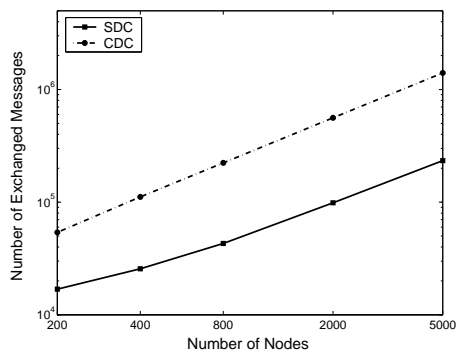


Fig. 4. Message overhead in pure random topologies

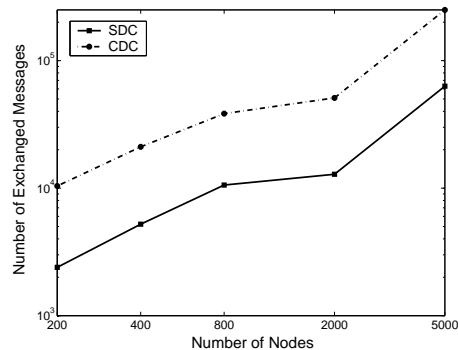


Fig. 5. Message overhead in power-law topologies

However, the increasing rate in CDC is much faster than in SDC.

Fig. 5 shows the clustering overhead of SDC comparing with CDC in power law topologies. It is clear that SDC yields much smaller message overhead than CDC. Another observation (by comparing Fig. 4 and Fig. 5) is that for the same network size, power-law topologies have smaller overhead than pure random topologies. This is because these two type of topologies have very different network structures. In power-law topologies, only a few nodes can have large degrees while the degrees of other nodes' are very small. Therefore, the message overhead in such networks is mainly caused by the clustering of a small number of nodes that have high degrees.

C. Performance of SDC in Dynamic Systems

We now evaluate the performance of SDC in systems with node dynamics. In this set of experiments, we use power-law topologies, with the same degree parameters as before. We measure the message overhead and clustering accuracy for arbitrary node join (and leave). We run such experiment more than 100 times, and take the average performance values.

We first study node leaving. In SDC, as discussed in Section IV-C, when a node leaves the network, the affected nodes (its neighbors and the nodes in the same cluster) need to “re-cluster” in order to maintain good clustering accuracy. In CDC, upon a node entry or exit, the whole network has to be re-clustered. For comparison, we also run “SDC Reclustering”, in which the whole topology redoes SDC clustering after each node exits the network. The results are presented in Fig. 6 and Fig. 7. It can be seen that SDC can maintain a higher clustering accuracy than CDC while only much smaller overhead is introduced. Moreover, compared with “SDC Reclustering”, SDC yields almost same accuracy values, which further demonstrates the effectiveness of SDC for node leaving.

We conduct similar experiments for node joining. The performance is illustrated in Fig. 8 and Fig. 9. These figures show very similar results to node leaving. We can conclude that SDC can handle node dynamics very effectively.

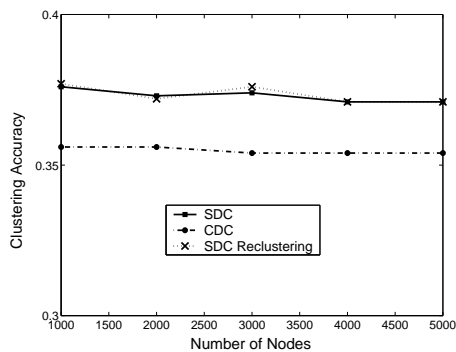


Fig. 6. Clustering accuracy on node exit

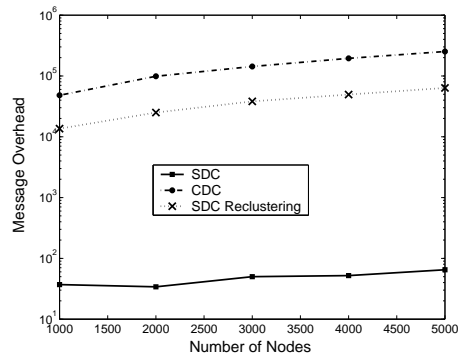


Fig. 7. Message overhead on node exit

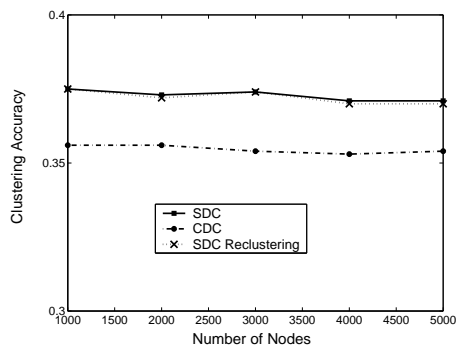


Fig. 8. Clustering accuracy on node joining

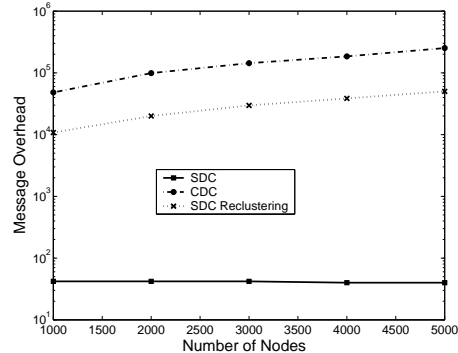


Fig. 9. Message overhead on node joining

D. Cluster Size Control

The major goal of SDC protocol is to detect accurate clustering in an efficient way. The previous sections have demonstrated the promising performance of SDC in terms of clustering accuracy and efficiency. To apply this clustering protocol to real network scenarios, more considerations have to be counted. One concern is the resultant cluster size. Too expanded clusters may bring in huge maintenance overhead and therefore are not preferred. SDC and CDC can effectively control the cluster diameter, but neither of them has an explicit control of the number of nodes in a cluster. This set of experiments aims to test the performance of SDC in controlling cluster size (number of nodes in a cluster) under the condition that high accurate clusters are formed.

Again, pure random topologies and power law topologies are used in our simulations. For each type of topologies, we vary the total number of nodes from 1000 to 5000 but fix the average node degree as 10 for the random topologies and 4 for the power law topologies (for which we also set the other parameters unchanged in order to maintain the topology structure).

First, we show the average cluster size obtained from SDC and CDC (Fig. 10 and Fig. 11). The experiment results show very stable average cluster size obtained from both SDC and CDC for topologies with different scales. Intuitively, as long as the overall topology structure is not changed, the cluster size should follow the same pattern even if the topology scale is changed significantly. Based on our simulation results, we declare that both SDC and

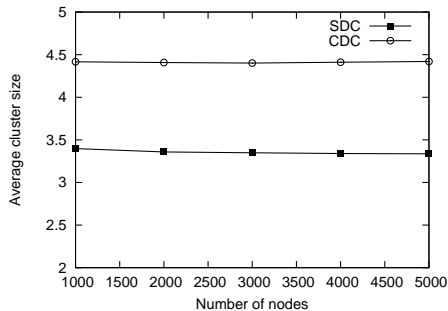


Fig. 10. Average cluster size for random topologies

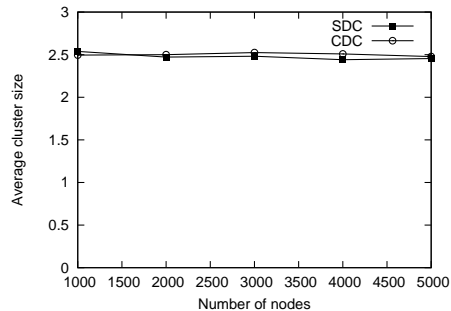


Fig. 11. Average cluster size for power law topologies

CDC have very stable performance in term of controlling the cluster size. For random topologies, SDC results in smaller average cluster sizes compared with CDC, but very similar average cluster sizes are observed for power law topologies. The explanation is easily obtained by looking at the distribution of cluster size of each topology (We compute the complementary cumulative distribution function of cluster size to represent the distribution). As shown in Fig. 12 and Fig. 13, the cluster size from CDC has much higher variability than from SDC for random topologies. More specifically, the maximum cluster size from CDC is 16 while for SDC the maximum cluster size is only 5, which explains the difference of the average cluster size between SDC and CDC for random topologies. However, when we look at the cluster size of power law topologies, the distributions for both algorithms are very close, which results in the similar average cluster sizes.

Here, we don't declare that SDC has better cluster size control over CDC, because in the real peer-to-peer systems, the requirement for cluster size varies with different applications. However, it seems that SDC can maintain more strict bounds for the cluster size than CDC as the cluster size distributions don't spread as much as those in CDC, which is more preferred in the real systems.

Another observation from this set of experiments is that SDC can effectively eliminate the orphan nodes whereas CDC results in a non-trivial amount of orphan nodes. Based on the Fig. 12 and Fig. 13, SDC doesn't leave any orphan nodes in the two types of topologies, but for CDC, more than 10% of nodes are left as orphan nodes. SDC beats CDC again in term of removing orphan nodes.

E. Influence of Node Degree

Both SDC and CDC are aimed at clustering a network based on node connectivity, and their performance may heavily depends on the density of the network (measured by different parameters in different types of topologies). In this set of experiments, we investigate the effect of node degree on the performance of SDC. We use pure random topologies, in which average node degree is a good measure of network density¹. We fix the network size as 100, and the average node degree from 2 to 12.

¹We choose pure random topologies because the average node degree this type of topologies can be easily controlled. Our analysis can also apply to other types of topologies

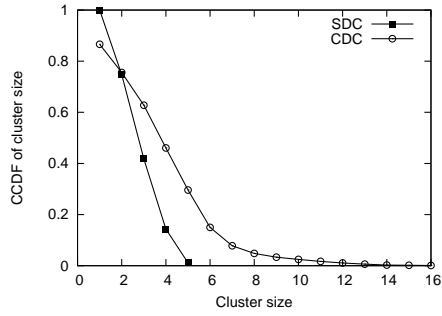


Fig. 12. CCDF of cluster size for random topologies

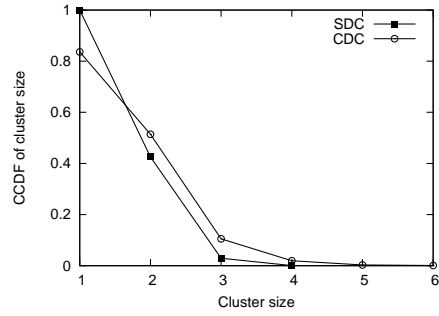


Fig. 13. CCDF of cluster size for power law topologies

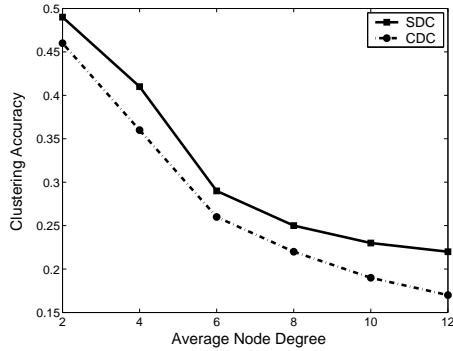


Fig. 14. Effect of node degree on clustering accuracy

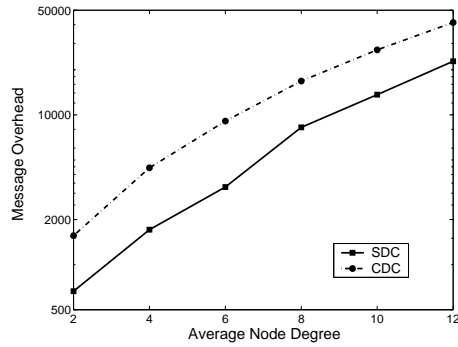


Fig. 15. Effect of node degree on message overhead

Fig. 14 shows the clustering accuracy of the SDC and CDC with the average node degree changed. Clearly, SDC performs better than CDC in all the cases. As the average node degree increases, the clustering accuracy of both schemes decreases. This decreasing trend is reasonable: with the increase of average node degree, pure random topologies have less significant clustering features. We also observe that the improvement percentage of SDC over CDC is enlarged as the average node degree increases. This demonstrates the capacity of SDC on discovering accurate clusters in networks with different densities.

We plot the results for message overhead in Fig. 15. Not surprisingly, the figure shows that SDC always causes less message overhead than CDC. The number of exchange messages arises with the increase of the average node degree in both algorithms, but the message overhead in CDC increases much faster than in SDC (note the y-axis is in log scale). When the topology is very sparse (e.g., the average node degree is 2), CDC generates more 897 messages than SDC, while when the average node degree becomes 12, the difference on message overhead between SDC and CDC reaches 18588. Through this set of simulations, we conclude that SDC achieves better performance than CDC for both sparse and dense networks.

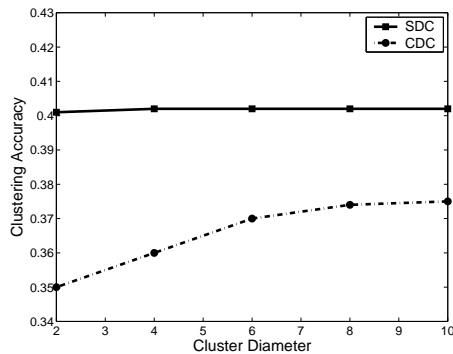


Fig. 16. Effect of TTL on clustering accuracy

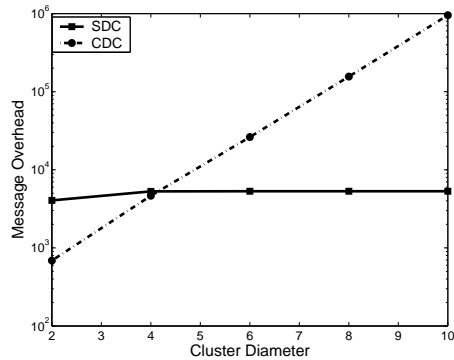


Fig. 17. Effect of TTL on message overhead

F. Influence of TTL

As both SDC and CDC use TTL to control cluster granularity, it is of interest to study the influence of TTL on the performance of the clustering algorithms. In this group of experiments, we evaluate the clustering accuracy and message overhead in SDC and CDC for different TTL values. We use a power-law topology with 500 nodes. We vary the TTL value, and run both of the algorithms for the same setting. Since TTL is implemented in different ways in SDC and CDC, we use a common metric, cluster diameter, to plot the results. In SDC, cluster diameter is equal to TTL plus 1, while in CDC, cluster diameter is twice of TTL.

Fig. 16 shows the results for clustering accuracy. It is observed that as cluster diameter increases, the clustering accuracy of SDC is quite stable, while the performance of CDC is significantly affected by the cluster size control. When cluster diameter reaches 8 (which is equivalent to TTL of 4), CDC becomes stable. If we examine the results for message overhead in Fig. 17, we have the following interesting findings: SDC has higher message overhead than CDC when cluster diameter is very small (2 in the figure). However, CDC has very bad clustering accuracy (referring to Fig. 16) at this point. Then when cluster diameter control is relaxed, the overhead of SDC is again very stable, while CDC involves log-scale increasing overhead. This well justifies the choice of TTL in our earlier experiments: we set TTL as 3 so that CDC can achieve the best performance.

VI. CONCLUSIONS

In this paper, we propose a distributed clustering protocol, SDC, for peer-to-peer networks. We first identify the main challenges in network clustering of peer-to-peer networks, and discuss the criteria for good realistic clustering algorithms. Then we present our distributed clustering protocol SDC, which is essentially a greedy approach based SCM. It can satisfy all the criteria for a good clustering algorithm: it considers node connectivity; it well-controls the cluster size; it minimizes the number of orphan nodes; and it can locally handle node dynamics with small overhead. Through simulations, we demonstrate that SDC can achieve much better performance than CDC in terms of both clustering accuracy and message overhead.

REFERENCES

- [1] Gnutella development page. <http://gnutella.wego.com>.
- [2] Kazaa home page. <http://www.kazaa.com>.
- [3] J. Ahuja and J.-H. Cui. A scalable peer-to-peer file sharing system supporting complex queries. *UCONN CSE Technical Report, UbiNet TR05-01*, January 2005.
- [4] A. Barbu and S.-C. Zhu. Graph partition by swendsen-wang cuts. *Ninth IEEE International Conference on Computer Vision Volume 1, 10 13 - 10 2003, Nice, France*.
- [5] T. Bu and D. Towsley. On distinguishing between internet power law topology generators. *In Proceedings of INFOCOM, 2002*.
- [6] K. Calvert and E. Zegura. Gt-itm: Georgia tech internet network topology models. <http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz>, 1996.
- [7] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems, 2002.
- [8] J.-H. Cui, M. Faloutsos, D. Maggiorini, M. Gerla, and K. Boussetta. Measuring and modelling the group membership in the internet. *In Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC2003)*, pp. 65-77, Miami, Florida, October 27-29, 2003.
- [9] J. S. Deogun, D. Kratsch, and G. Steiner. An approximation algorithm for clustering graphs with dominating diametral path. *Inf. Process. Lett.*, 61(3):121–127, 1997.
- [10] D. Estrin, Y. Rekhter, and S. Hotz. Scalable inter-domain routing architecture. *In SIGCOMM '92: Conference proceedings on Communications architectures & protocols*, pages 40–52, 1992.
- [11] Y. Fernandess and D. Malkhi. K-clustering in wireless ad hoc networks. *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 31–37, 2002.
- [12] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller. Hierarchical p2p systems. *In Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, 2003.
- [13] C. Gkantsidis, M. Mihail, , and E. Zegura. Spectral analysis of internet topologies. *IEEE INFOCOM 2003*.
- [14] G. Kwon and K. D. Ryu. An efficient peer-to-peer file sharing exploiting hierarchy and asymmetry. *In SAINT*, pages 226–233, 2003.
- [15] A. McDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communication*, Vol. 17, No. 8, August 1999.
- [16] A. Medina, A. Lakhina, I. Matta, , and J. Byers. Brite: Universal topology generation from a user's perspective. *In Proceedings of Workshop the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '01)*, October 2001.
- [17] L. Ramaswamy, B. Gedik, and L. Liu. A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 16(9), Sept. 2005.
- [18] S. van Dongen. A new cluster algorithm for graphs. *Technical report INS-R9814, Centrum voor Wiskunde en Informatica (CWI), ISSN 1386-3681*, Dec. 1998.
- [19] S. van Dongen. Performancde criteria for graph clustering and markov cluster experiments. *Technical report, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam*, 2000.