Connectivity Based Node Clustering in Decentralized Peer-to-Peer Networks

Lakshmish Ramaswamy
College of Computing
Georgia Institute of Technology
Altanta, GA 30324
Email: laks@cc.gatech.edu

Bugra Gedik
College of Computing
Georgia Institute of Technology
Altanta, GA 30324
Email: bgedik@cc.gatech.edu

Ling Liu
College of Computing
Georgia Institute of Technology
Altanta, GA 30324
Email: lingliu@cc.gatech.edu

***Abstract*— Connectivity based node clustering has wide ranging applications in decentralized Peer-to-Peer (P2P) networks such as P2P file sharing systems, mobile ad-hoc networks, P2P sensor networks and so forth. This paper describes a Connectivity-based Distributed Node Clustering scheme (CDC). This scheme presents a scalable and an efficient solution for discovering connectivity based clusters in peer networks. In contrast to centralized graph clustering algorithms, the CDC scheme is completely decentralized and it only assumes the knowledge of neighbor nodes, instead of requiring a global knowledge of the network (graph) to be available. An important feature of the CDC scheme is its ability to cluster the entire network automatically or to discover clusters around a given set of nodes. We provide detailed experimental evaluations of the CDC scheme, addressing its effectiveness in discovering good quality clusters. Our experiments show that utilizing message-based connectivity structure can considerably reduce the messaging cost, and provide better utilization of resources, which in turn improves the quality of service of the applications executing over decentralized peer-to-peer networks.**

I. Introduction

In recent years the field of distributed data management systems has witnessed a paradigm shift from the traditional Client-Server model to the Peer-to-Peer (P2P) computing model. While file sharing applications like Gnutella [4] and Kazaa [6] were the harbingers of this change, various other systems like mobile ad-hoc networks, P2P sensor networks etc. have adopted this model of computation and communication.

Although these systems appear to be diverse and disparate, all of them share some distinct characteristics:

- Network and data management are completely decentralized
- Individual nodes have limited knowledge about the structure of the network
- Networks are dynamic with frequent entry and exit of nodes

In this paper we use the term Peer-to-Peer systems in a generic sense to refer to any system that possesses these characteristics. Although, the P2P distributed computing paradigm alleviates the scalability problem that has dogged client-server systems and enables a lot of interesting and useful applications, it also raises key research challenges that need to be addressed by the community including:

1) Scalable techniques for data discovery and peer look-up
2) Efficient mechanisms for communication among nodes in the network

The strategies adopted by most of the present systems to address these challenges are costly in terms of the number of messages required.

It is our contention that the absence of any knowledge of the network structure is proving to be a stumbling block in utilizing the full capabilities of P2P networks. We believe that every such network exhibits some unique structural properties and discovering these network structures is crucial to efficient data discovery, node look-up and communication.

Connectivity-based node clustering is one such interesting and important network structure that can be utilized in various ways to improve the quality of service of applications running on these networks. Informally, a connectivity-based node clustering (hereafter referred to as node clustering) can be defined as a partition of network nodes into one or more groups based on their connectivity. We provide a formal definition of a node-cluster in Section II. For now we shall assume that two nodes that are highly connected are placed in the same cluster.

To illustrate the utility of node clustering, consider a P2P file sharing system like Gnutella [4] or Freenet [3]. The predominant file discovery mechanism in such systems has been broadcast-based breadth first search. Though, this is a simple solution its major drawback immediately comes to the fore: The number of messages in the network is very high. To overcome this drawback some researchers have proposed techniques to limit the breadth first search by sending the query message to a few neighbors that are either random or selected based on some criteria. The tradeoff here is between the number of messages and the quality of search results.

Now let us see how we can utilize the cluster structure for limiting the number of messages, while ensuring minimal degradation of search results. Suppose the nodes in the network are clustered and each node in the network knows which of its neighbors belong to its own cluster. Now a node V_i in the system, on receiving a query message forwards it to all its neighbors that do not belong the same cluster as V_i. From the neighbors that belong to its cluster, V_i selects a very few nodes and forwards the message to them. The logic behind this scheme is that nodes in same clusters are highly connected and hence it is not necessary to send query message to each one

of them.

A second example of the utility of node cluster information is the design of an intelligent file replication scheme for P2P file sharing systems. The problem here is to reduce the number of replicas of files, while ensuring that average file download latency does not increase significantly. Some systems replicate a file at each node the file passes through. Other systems replicate a file only at those nodes which downloaded the file. A simple scheme that uses cluster information would limit the number of replicas in each cluster to some small value. In addition to the above two examples, researchers in the past have applied clustering information to address certain key problems in various decentralized P2P networks [1], [8], [7], [5].

Although node cluster information has been utilized to improve performance and scalability in P2P networks, very few researchers have studied the problem of discovering and maintaining node clusters in P2P systems. There has been considerable research in the algorithm community addressing the problem of clustering nodes in directed and undirected graphs. Although the P2P systems are essentially undirected graphs, most existing graph clustering algorithms assume that the entire graph information is available in one central location. Unfortunately, none of the P2P networks maintain their complete and up-to-date connectivity information. Therefore the need is to design schemes that can cluster the nodes of a network in a completely distributed and decentralized manner.

With these problems in mind, this paper presents a Connectivity based Decentralized node Clustering scheme (CDC), a scalable and an efficient solution for discovering connectivity based clusters in peer networks. In contrast to centralized graph clustering algorithms, the CDC scheme only requires the local knowledge about neighboring nodes. An important strength of the CDC scheme is its ability to cluster the entire network automatically or to discover clusters around a given set of nodes in the network. Our experimental results indicate that these schemes yield high quality clusters. An initial experimental evaluation of the CDC scheme is provided, showing the effectiveness of the CDC scheme in discovering good quality clusters.

II. Definitions and Terminologies

The connectivity structure of every P2P network can be represented by an undirected graph with nodes of the P2P network forming the vertices of the graph and connections between nodes being the edges of the graph. Henceforth we use the terms graph and network interchangeably. Similarly, the terms node and vertex are used equivalently, and so are the terms edges and connections.

Let $G = (V, E)$ be an undirected graph, where $V = \{V_1, V_2, V_3, ..., V_N\}$ is the set of nodes and $E = \{E_1, E_2, E_3, ..., E_M\}$ is the set of edges in the graph G. Two nodes V_i and V_j in the graph are said to be *Connected* if there exists a series of consecutive edges $\{E_1, E_2, E_3, ..., E_M\}$ such that E_1 is incident upon the vertex V_i and E_p is incident upon the vertex V_j. The series of edges leading from V_i to V_j is called a *Path* from vertex V_i to V_j. The length of a path P is the number of edges in the path.

A *Similarity function* d on the vertices of a graph G is a symmetric function mapping $V \times V$ to $\mathbb{R}_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ is the set of positive real numbers and $d(u, v) = d(v, u)$. Further the function satisfies the condition that $d(v, u) = \infty$ iff $v = u$. The similarity and dissimilarity function may be appropriately defined according to the semantics of the particular graph under consideration and the particular application at hand.Two of the most popular similarity functions have been the number of *K-Paths* between the vertices and the *Reach Probability* from one vertex to another in the graph.

A *Clustering* Cl of a graph $G = (V, E)$ is collection of the sets $\{Cl_1, Cl_2, ..., Cl_Q\}$, satisfying the following three conditions: (1) each Cl_l is a non-empty subset of vertices ($\forall Cl_l, Cl_l \subseteq V$) and $\bigcup_{l=1}^{l=Q} Cl_l = V$. (2) Any two nodes in Cl_l, $1 \leq l \leq Q$, are similar. (3) Any two nodes belonging to two different sets, say Cl_l and Cl_m, are not similar. Each of the Cl_l in Cl is termed as a cluster. A clustering P is termed as a *Disjoint Clustering* if the clusters are pair-wise disjoint.

For graph clustering problem, the similarity can be defined in host of meaningful ways. Each of these definitions lead to different natural clusters and have different applications. However there are some properties that most useful clustering schemes share.

- In a graph G, for any two nodes V_i and V_j which belong to the same cluster Cl_l, there exists at least one path in G such that all intermediate vertices along that path lie in Cl_l.
- For a graph G, any two vertices lying in the same cluster tend to have large number of paths connecting them.
- A random walk on the graph G tends to visit most of the nodes in a cluster multiple times before it leaves the cluster.

Based on these properties of good clustering, a number of graph clustering algorithms have been proposed. The two most popular schemes are: the **K-Path Clustering Algorithm** and the **MCL Algorithm** [10]. Like most existing graph clustering algorithms, both of them assume that the global information about the entire graph (i.e., the number of vertices, the number of edges, and their connectivity) is available in one central location.

III. CDC Scheme for Graph Clustering

In contrast to centralized graph clustering algorithms, the problem of distributed clustering assumes that each node has limited view of the entire network. In this section we present our scheme for distributed node clustering, termed as the Connectivity based Decentralized node Clustering scheme (CDC). First, we first formalize the distributed node clustering problem and then discuss the CDC approach.

Let $G = (V, E)$ be an un-weighted, undirected graph. Each node V_i in this graph is mapped to an autonomous and independent computing element CE_i. Further each of these computing elements knows only its neighbors. In other words, the node CE_i has the knowledge of the existence of another

node CE_j iff CE_i and CE_j are neighbors in the graph G. This condition has important connotations. First, it implies that we do not have a centralized global view of the graph G. Second, it also means that each node can communicate only with its immediate neighbors. If it ever wants to reach a node that is not its neighbor, then it would have to route the message through one of its neighbors. The problem is how to discover reasonable node clusters in the graph G in a completely distributed fashion, i.e. without ever constructing a global view of the graph.

The problem setting described above reflects the scenario in real-world systems like P2P file sharing systems, P2P sensor networks and ad-hoc mobile networks. As an example let us consider the Gnutella network [4]. Each peer in the network maintains a live TCP connection with a few other peers, which are called its neighbors. The knowledge of each peer about the network is limited only to its immediate neighbors.

The central idea in the CDC scheme is to simulate flow in the network in a distributed and a scalable fashion. Clustering a graph through flow simulation is based on the following intuition. Let us think of the graph as a network of mutually intersecting roads. The roads are the edges of the graph and the intersection of two or more roads are the nodes of the graph. Suppose a large number of people who do not know the structure of the roads start out from a node V_i in the road graph, which we call the **Originator node**. Let each person carry a weight W_i along with him. As these persons are not aware of the structure of the roads, they choose any of the roads starting out from the node V_i and travel along the road to reach another intersection. Whenever they reach an intersection, they drop some of the weight they are carrying at the intersection. Then they choose another road at random and continue to travel along that road to execute the same cycle till they are tired of walking or the weight they carry becomes negligible. Now if one were to aerially observe the roads and the intersections, he would observe two facts:

- If the graph structure has a densely connected graph structure around the originator node, then a high percentage of people can be observed in the nodes (intersections) and edges (roads) that lie in the dense region (i.e. cluster) around the originator.
- Nodes that lie inside the cluster would have accumulated a higher weight from all the people who passed through the node and dropped part of their weight. In comparison, the nodes that are remotely approachable from the originator would have accumulated very small weight.

These observations lead us to the central idea of the algorithm. If there are a few originators in the graph from where people would start their random walk, the nodes would acquire weight from different originators. The idea then is that each node would join a cluster from whose originator, it received the maximum weight. If some node did not receive enough weight from any node, then it decides to be an outlier node (a node that does not belong to any cluster).

In a distributed P2P network, peer nodes are analogous to intersections, and connections between peers represent roads. People moving about are simulated by messages that are circulated in the network. Each message has a predefined *Time to Live*(TTL for short). Each message executes only TTL hops, after which it expires and is discarded.

A. CDC Algorithms

The CDC algorithm starts out by initiating messages from a set of nodes that are the originators of the clustering algorithm. The set of originators is represented by $O = \{O_1, O_2, ..., O_Q\}$, which initiate the process of message circulation by sending out messages to all its neighbor nodes.

Each cluster message is a tuple consisting of the following five fields:

- **Originator ID (OID):** A field uniquely identifying the Originator node
- **Message ID (MID):** A field distinguishing each message from all other messages from the same originator.
- **Message Weight (MWeight):** The weight carried by the message
- **Source ID (SourceID):** A field indicating the most recent node the message visited
- **Time to Live (TTL):** The maximum number of hops this message can be re-circulated

The algorithm for the originator selection itself is completely distributed and is explained in detail in Section III-B. For now we assume that we have been provided with a set of originators.

The $SourceID$, the MID and the OID fields in the message tuple are self explanatory and straightforward to initialize. The weight function that we use estimates the probability of reaching any node from originator nodes. An originator node O_l initializes message weight as $Msg.MWeight = \frac{1}{Degree(O_l)}$.

Each node V_i maintains a set of values, represented as $TotalWeight(V_i, O_l)$. This value indicates the sum of the weights from all the messages that originated at O_l and reached V_i. On receiving a message Msg, the recipient V_i updates the $TotalWeight$ function corresponding to the message originator. Then the node V_i checks whether the TTL of the message is greater than 0. If so, V_i forwards the message to all its neighbors. Before the re-circulation, the recipient updates the $MWeight$ and the TTL. The message weight is divided by the degree of V_i and the TTL is decremented by 1.

Each node may receive multiple messages from several different nodes. It calculates the $TotalWeight$ function for each of the originators from which it received messages. Each node joins the cluster led by the Originator, for which the value of $TotalWeight$ is the maximum. If all $TotalWeight$ values lie below a predefined threshold, then the node remains an outlier. The psuedo-code for the CDC scheme is provided in Algorithm 1 and 2.

Now we provide an example to illustrate the functioning of the CDC algorithm. In Figure 1 we have a graph of 14 nodes, labeled from 0 to 13. We illustrate how the algorithm works when the process is initiated by three Originator nodes

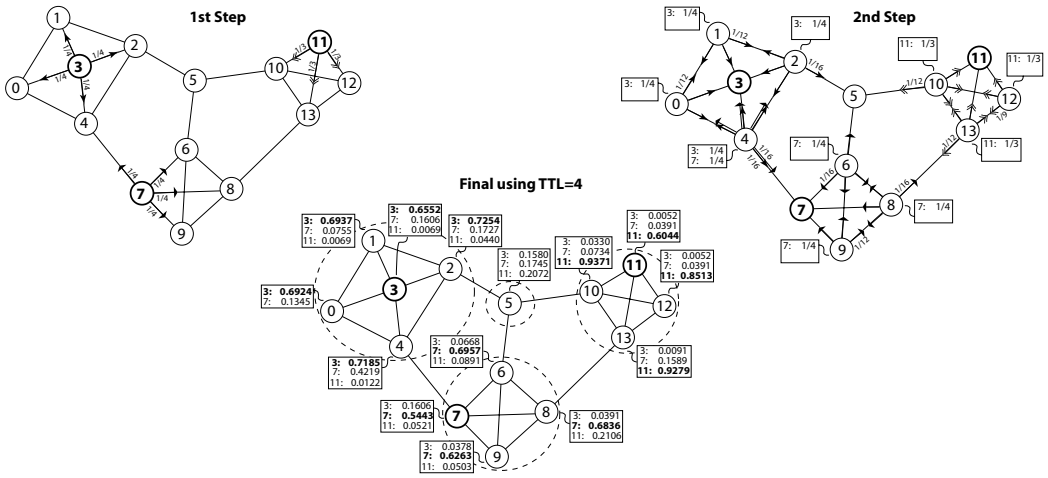

Fig. 1: CDC Illustration

$\{3, 7, 11\}$. The figure illustrates $TotalWeight$ acquired by each node at different steps of the algorithm execution. The arrows in the diagram denote the messages flowing through the system. The weight carried by each message is indicated next to the arrow. The last diagram shows the $TotalWeight$ each node has gained due to the three originators after 4 hops. Each node joins the cluster from which it has gained the maximum $TotalWeight$ leading to three clusters marked in the diagram. However, for node 5, the $TotalWeight$ received from all Originator nodes are less than the threshold and hence it becomes an outlier.

Algorithm 1 Algorithm Executed by Message Originator O_l

Create a New Message Msg
$Msg.OID \leftarrow O_l$, $Msg.MWeight \leftarrow \frac{1}{Degree(O_l)}$
$Msg.SourceID \leftarrow O_l$, $Msg.TTL \leftarrow InitialTTL$
$Msg.MID \leftarrow$ Current System Time {A unique value}
for Each node $V_i \in Nbr(O_l)$ **do**
  Send Msg to V_i
end for

The algorithms described above are self-explanatory. However, we want to discuss an important issue regarding the weight function we are using in the algorithm. If a node V_i in the graph receives q messages whose $TTL = h$ from originator O_l, then the quantity, $\sum_{Msg.TTL=h}(Msg.MWeight)$ has special significance. This quantity indicates the probability of being in node V_i, if one were to start from node O_l and perform a random walk of exactly $(InitialTTL-h+1)$ steps.

This section described algorithms to discover clusters around a given set of Originators. In the next section we address the question of how to select Originator nodes so that the scheme yields good clusters.

B. THP Originator Determination Scheme

Choice of Originators is critical to the performance of the CDC scheme discussed in the previous section. We discuss an example that elucidates the significance of selecting "good" originator nodes. The diagrams in Figure 2 shows four different scenarios, indicating the clusters we obtain when we start out with four different sets of Originators. In each scenario, the originators are indicated by the shaded nodes.

Algorithm 2 Algorithm Executed by Node V_i on Receiving Msg

{Check whether I have received messages from $Msg.OID$}
if I have seen messages from $Msg.OID$ before **then**
  {Check if the $LastMsgId(O_l) == Msg.MID$}
  if $LastMsgId(O_l) == Msg.messageID$ **then**
    $TotalWeight(V_i, O_l) \leftarrow TotalWeight(V_i, O_l) + Msg.MWeight$
  else
    $TotalWeight(V_i, O_l) \leftarrow Msg.MWeight$
    $LastMsgId(O_l) \leftarrow Msg.messageId$
  end if
else
  {This is the first message from O_l}
  $TotalWeight(V_i, O_l) \leftarrow TotalWeight(V_i, O_l) + Msg.MWeight$
  $LastMsgId(O_l) \leftarrow Msg.MID$
end if
if $TotalWeight(V_i, O_l) > MaxWeight$ **then**
  $MaxWeight \leftarrow TotalWeight(V_j, O_l)$
  $MaxWeightId \leftarrow Msg.OID$
end if
if $Msg.TTL > 0$ and $\frac{Msg.MWeight}{Degree(V_i)} > MinWeight$ **then**
  Create a New Message $NewMsg$
  $NewMsg.OID \leftarrow Msg.OID$, $NewMsg.SourceID \leftarrow V_i$
  $NewMsg.MWeight \leftarrow \frac{Msg.MWeight}{Degree(V_i)}$
  $NewMsg.TTL \leftarrow (Msg.TTL - 1)$, $NewMsg.MID \leftarrow Msg.MID$
  for Each node $V_i \in Nbr(O_l)$ **do**
    Send Msg to V_i
  end for
end if
Wait for $WaitTime$ in anticipation of other messages
if $MaxWeight > WeightThreshold$ **then**
  Join the cluster led by $MaxWeightId$
else
  Remain an outlier
end if

Scenario 1 is the best clustering we can obtain for the graph. In this case we have three clusters and a single outlier. The clustering in scenario 2, though not ideal is again a good clustering. The clusters we obtain in other two scenarios are unintuitive and are in no way close to ideal clustering in scenario one. Though we have used the same TTL, the same weight function and the same number of Originators, we obtain different clusters that not only vary in number but also in their quality. This example demonstrates the importance of

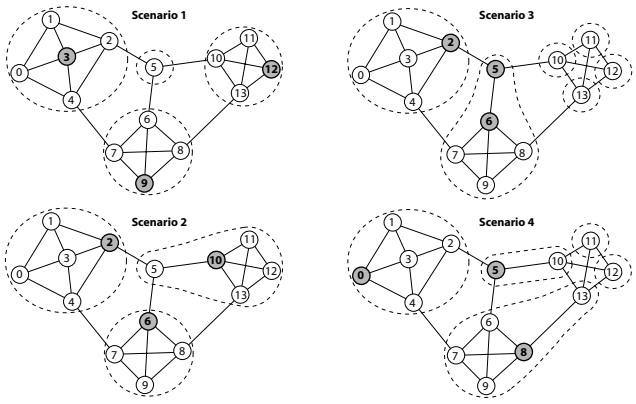

Fig. 2: Importance of Good Originators

selecting good Originators.

Now we briefly discuss the properties a good originator set should possess.

- First, the set of originators should be spread out in all regions of the graph.

If some regions in the graph do not have any originators, then nodes in these unrepresented regions do not receive enough messages and hence do not acquire sufficient weight from any originator. Hence these nodes either get associated with a cluster where they do not really belong or they choose to remain as outliers, both of which result in bad clusters.

- Second, a node V_l is considered to be a good originator if it acquires more weight due to messages initiated by it than the weight acquired by messages initiated by any other originator. i.e. $TotalWeight(V_l, V_l) \geq TotalWeight(V_l, V_i), \forall V_i \in V$.

In any graph, it is not desirable to have originators that accumulate more weight from messages that originated at some other node than the messages initiated by it. If such were to be the case, then the originator itself would "defect" to a cluster initiated by some other originator. This again would result in the formation of bad clusters.

The originators Scenario 1 and Scenario 2 in Figure 2 satisfy both these conditions and yield good clusters. In contrast the originators in Scenario three are concentrated in one single region. The originators in Scenario 4 do not satisfy the second condition. Both of these yield bad clusters.

Although the second property is logical, the crucial question is how do we determine whether a node satisfies this criterion? This property demands that we know $TotalWeight$ for each pairs of vertices, which cannot be determined till we actually execute the CDC scheme.

We adopt an approximation technique to solve this problem, which we call the **Two Hop Return Probability** technique. In this technique we determine the probability of returning to a node V_l in the graph in two hops, if we were to perform a random walk on the graph starting at V_l. This is calculated as $TwoHopProb(V_l) = \sum_{V_i \in Nbr(V_l)} (\frac{1}{Degree(V_l) \times Degree(V_i)})$. A higher $TwoHopProb(V_l)$ indicates that the node V_l has a higher chance of satisfying the condition $TotalWeight(V_l, V_l) \geq TotalWeight(V_l, V_i), \forall V_i \in V)$, and hence has a lesser chance of "defecting" into another cluster. As this scheme relies upon the two hop return probability, we term it as the Two-Hop-Probability scheme or THP scheme for short.

The THP scheme performs two tests. First, it checks whether the node has already received any clustering messages from other nodes in its vicinity. If so, it means that there are other nodes in its vicinity that have already chosen to be Originators. Hence the node opts not to become an originator and does not initialize messages. If the node discovers that there are no Originators in its vicinity, then it obtains the degree of each of its neighbors and computes the Two Hop Return Probability ($TwoHopProb$). If the Two Hop Return Probability is higher than a pre-defined threshold ($TwoHopThreshold$) then the node chooses to be an Originator. Otherwise, the node will not become an Originator.

The two configurable parameters for this algorithm are the $Vicinity$ factor and the $TwoHopThreshold$ factor. If these factors are set to high values, the number of originators would drop and vice-versa.

The THP algorithm is completely distributed. The number of messages circulated in this phase is also very small. Each node, which has not received a message from an originator in its vicinity, has to just get the degree of each of its neighbors. Hence this scheme is very efficient in terms of the messaging cost.

IV. Handling Node Dynamics

Most P2P networks exhibit some degree of dynamism in their structures, although the nature and the scale of dynamism are specific to each network. For example in P2P file sharing systems, the peer nodes enter and exit the system at arbitrary points in time. This dynamics in the structure affects the existing clusters of the nodes in the network. Re-clustering the entire network on each node-entry or exit is impractical for two reasons: 1. Re-clustering on each node-entry and exit causes the network to be loaded with clustering messages. 2. If the entry and exit of the nodes are frequent, then the clusters never stabilize. Even before the clusters are discovered, another node might enter or exit the system, causing the whole process to restart. Therefore efficient and effective schemes are algorithms are needed to handle the node entry and exit.

We have designed algorithms to handle the entry and exit of the nodes in P2P networks. These schemes are based on the fact that node entry and exit are localized phenomena that affect the network structure in the close vicinity of the node that is entering or exiting. The key idea of the node entry mechanism is to calculate the "attraction" between the exiting clusters and the entering node. The node then joins the cluster to which it is most attracted to. Similarly when a node exits the system, its immediate neighbors recalculate their "attraction" to various clusters and "defect " to the cluster to which they are most attracted to. These algorithms are completely distributed and invloves messaging between the entering/exiting node and its immediate neighbors. Due to space limitations, we do

not discuss these schemes in this paper. Interested readers may refer to our technical report [8], where we discuss the algorithms in detail and provide experimental evaluation of the schemes.

V. Experiments and Results

This section reports the experiments we performed to evaluate the proposed schemes and the results we obtained. We begin by discussing the metric used for measuring graph clustering accuracy.

A. Accuracy Measure for Graph Clustering

Measuring the accuracy of a given clustering on graph is in general a tricky task. This is primarily because, unlike data points in Euclidean spaces, the distance measure for graphs can be defined in many different meaningful ways. Hence it is possible to obtain different accuracy measures based on the different similarity measures. In this paper we use an intuitive performance measure proposed by [10] termed as the *Scaled Coverage Measure*.

Let $G = (V, E)$ be a graph and let $Cl = \{Cl_1, Cl_2, ..., Cl_Q\}$ be a given clustering on the graph. Let us consider any node V_i.

- Let $Nbr(V_i)$ denote the set of neighbors of V_i.
- Let $Clust(V_i)$ denote the set of all nodes that belong to the same cluster as that of vertex V_i, i.e. $Clust(V_i) = Cl_l$ if $V_i \in Cl_l$.
- **False Positive Set** is defined as the set of all nodes, which are not neighbors of V_i, but are included in the same cluster as V_i, i.e. $FalsePositive(V_i, Cl) = \{V_j | V_j \in Clust(V_i) \wedge V_j \notin Nbr(V_i)\}$.
- **False Negative Set** is the set of all neighbors of (V_i) but are excluded from $Clust(V_i)$, i.e. $FalseNegative(V_i, Cl) = \{V_k | V_k \in Nbr(V_i) \wedge V_k \notin Clust(V_i)\}$.

Then define *Scaled Coverage Measure* of the V_i in G with respect to the clustering C as:

$$ScalCov(V_i, Cl) = 1 - \frac{\|FalsePostive(V_i, Cl)\| + \|FalseNegative(V_i, Cl)\|}{\|Nbr(V_i) \cup Clust(V_i)\|} \quad (1)$$

Some of the salient features of the Scaled Coverage Measure are:

- It assumes a value of 1.0 at best and 0.0 at worst.
- It "punishes" a clustering for both false positives and false negatives.
- A node in a sparse region of the graph is penalized more for false positives and false negatives than a node in a dense region.

The accuracy of a clustering C over a graph G is defined as the average of the Scaled Performance Measure of all of its nodes.

$$ClustAccuracy(G, Cl) = \frac{\sum_{V_i \in V} ScalCov(V_i, Cl)}{\|V\|} \quad (2)$$

The highest clustering accuracy achievable for any graph G is called its optimal clustering accuracy. The optimal clustering accuracy for any graph depends upon its structure. It evaluates to 1 for graphs which contain only fully connected components, with no edges across the components. For all other graphs the optimal clustering accuracy is strictly less than 1.

Parameter	Power Graphs	Range Graphs
Total Nodes	5000	5000
Total Edges	11446	27083
Average Degree	4.57	10.83
Maximum Degree	623	25
Minimum Degree	1	1
Variance in Degree	212.53	11.08

TABLE I: Parameter Values for Various Topology Graphs

B. Experimental Data Sets

For our experimental evaluation, we have used two kinds of datasets:

Power Law Topology: We use power law topology to generate graphs that resemble P2P data sharing networks in their topological structure. Studies, in past few years, on topology of the Internet [2] and more recently on P2P data sharing networks [9], have revealed that topology of such networks closely follow what is well known as a *Power Law Distribution*. For our experiments we have used power law topology graphs with 100, 200, 500, 1000, 2500 and 5000 nodes.

Range Topology: Range topology models the connectivity relationship in wireless/sensor networks. In wireless networks two computing units have the knowledge of the existence of the other, only if they fall in each others radio range. Range topology graphs model this phenomenon. In our experiments we have use range topology graphs with 100, 200, 500, 1000, 2500 and 5000 nodes.

Table I lists some of the important properties of both datasets.

C. Experimental Results

In this section we provide a brief description of each experiment and the results obtained. We begin by comparing the accuracy of the CDC algorithm.

Cluster Accuracy of CDC Scheme

Our first experiment is aimed towards demonstrating the effectiveness of the CDC scheme we have proposed. In order to do so, we compare the accuracy of the CDC scheme with the Centralized MCL Clustering and the Distributed K-Path Clustering schemes. We use the public domain software developed by the author of [10] for obtaining clusters in the centralized MCL scheme.

The MCL graph clustering software requires us to set a configurable parameter, which controls the clustering granularity. This parameter can take on values from 1.2 to 5. When the parameter is set to lower values, the algorithm yields fewer number of large clusters and vice versa. We have obtained clusters by setting it to 1.2, 2, 3, 4 and 5. We measure the accuracy of the clusters obtained and use the highest value as our benchmark. We would like to make it clear that a

value, which is in between these values might yield cluster with slightly higher accuracy.

In this experiment we want to test the accuracy of the clusters yielded by the bare CDC algorithm. Hence we turn off the THP originator determination mechanism. Instead, we randomly select originators. In the experiments we report we randomly select 15% of the total nodes in the graph. These nodes act as the originators, initiating the messages. As the originator selection is random, we have performed each experiment 100 times and we report the average of the accuracy values obtained.

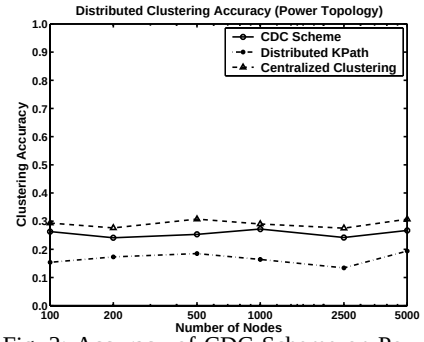

Fig. 3: Accuracy of CDC Scheme on Power Graphs

Figure 3 and 4 indicate the clustering accuracy of CDC scheme, the centralized clustering scheme and the Distributed K-path clustering scheme on power-law topology and range topology graphs respectively. The graphs show that the CDC scheme performs better than the K-path clustering scheme for all graph sizes of both power and range topology. The accuracy value of the CDC scheme is around 27% and 11% higher than the accuracy of the distributed K-path scheme for a power-law graph and range topology graphs of 5000 nodes respectively.

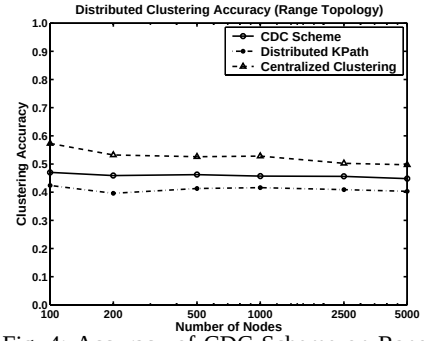

Fig. 4: Accuracy of CDC Scheme on Range Graphs

However the centralized MCL scheme performs better than the CDC scheme for both power-law and range graphs. The centralized MCL scheme beats the CDC scheme by almost 14% and 11% for power graphs and range graphs with 5000 nodes respectively.

These results show that the CDC scheme can yield clustering results comparable to the centralized scheme, thus demonstrating the reasonableness of the CDC approach.

THP Originator Mechanism Accuracy

Having demonstrated that CDC scheme is a reasonable approach for the distributed graph clustering problem, we now demonstrate the effectiveness of THP originator selection mechanism.

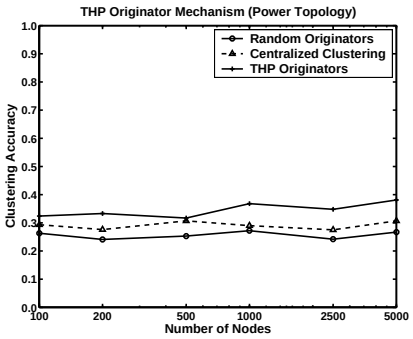

Fig. 5: Accuracy of THP Scheme on Power Graphs

Figure 5 and 6 indicates the clustering accuracy of the Two-Hop Probability originator determination scheme and compares it with the clustering accuracy of the CDC scheme with random originator selection and centralized MCL scheme on power-law and range topology graphs respectively

The results show that THP originator determination scheme improves the clustering accuracy of the CDC scheme considerably for graphs of both topologies. For example, on a power-law graph of 5000 nodes, the THP originator mechanism yields a mean accuracy value 0.381 as against 0.267 given by the CDC scheme with random originators, which amounts to an improvement of over 42%. Similarly for range graphs of 5000 nodes, the improvement is over 21%.

Surprisingly, the CDC scheme coupled with the THP originator determination mechanism yields better accuracy values than the centralized MCL scheme for both power and range graphs with 200, 500, 1000, 2500 and 5000 nodes. We did not expect our scheme to perform better than the centralized scheme. We think that this phenomenon is due to the structural properties of these two topologies. We feel that for graphs of other topologies, the centralized scheme would perform slightly better than the CDC scheme with THP originator mechanism. Hence, though we do not claim that our algorithm performs better than the centralized clustering, we certainly feel that the accuracy of the clusters given by our scheme is very close to the accuracy of the clustering yielded by centralized MCL scheme.

Does CDC Scheme Need High TTL?

One concern which we had regarding the CDC scheme was whether we need to initialize messages with high TTL values in order to obtain good clusters? Using messages with high TTL values has two problems which might be detrimental to the practicality of the scheme. First, it increases the message load on the network. Second, it increases the time required for the clusters to emerge.

In order to figure out the effect of Initial-TTL on the clustering accuracy, we obtained various clusters by setting the Initial-TTL values from 1 to 5. Due to space constraints we

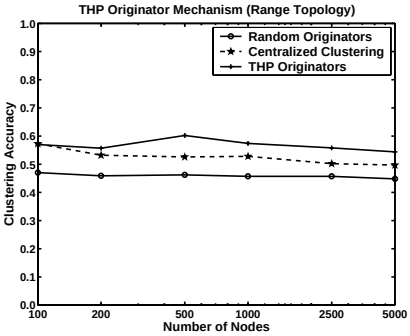

Fig. 6: Accuracy of THP Scheme on Range Graphs

limit our discussion to power-topology graphs. The Figure 7 indicate the accuracy values of range topology graphs with 200, 500 and 1000 nodes.

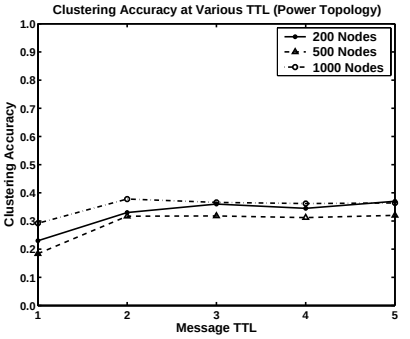

Fig. 7: Accuracy at Various TTL for Power Law Graphs

The results indicate that the algorithm yields good clusters even when the TTL is set to 2. Further the accuracy values for TTL of 5 are almost equal to the accuracy yielded by the scheme when the TTL is 2. This demonstrates that the scheme stabilizes very fast, which is a necessity for any distributed algorithm.

VI. Related Work and Conclusion

In this paper we have proposed the Connectivity based Distributed Node Clustering (CDC) scheme for clustering nodes of a decentralized peer-to-peer network. The scheme can either cluster the entire network automatically or detect clusters around a given set of nodes. To the best of our knowledge, this paper is the first one to report a completely distributed and decentralized scheme for node clustering. This work has been primarily inspired by the MCL algorithm proposed by Dongen [10]. Researchers in algorithmic graph theory have proposed a couple of algorithms for general graph clustering. Out of these algorithms the two most significant ones from a practical view point have been the K-path and the MCL clustering algorithms [10].

The K-path clustering algorithm is based on the observation that in a graph G, any two nodes in the same cluster (if such clusters indeed exist in G) would be connected by large number of paths of lengths k, $k > 1$. The MCL algorithm introduced by [10] views the graph as a **Markov Chain** and operates on the corresponding markov matrix, which contains one-step transition probabilities between all pairs of vertices. The algorithm defines a non-linear operator termed as the **Inflation operator**. Alternate application of the self-multiplication operator and the inflation operator, reveals the clusters in the graph.

The key difference between our work and the ones discussed in these papers is that these are centralized graph algorithms working on the global view of the graph, whereas our scheme is completely distributed and does not need a complete connectivity structure.

The work in the distributed computing community addressing the problem of electing cluster-heads bears some resemblance to our work [1]. These algorithms although distributed, do not attempt to cluster the network based on its connectivity structure. Hence the clusters discovered are not necessarily "good" clusters from a connectivity stand-point. In contrast our scheme is entirely based on connectivity structure of the network and hence leads to high quality clusters.

Several researchers have proposed applying node clustering information to various systems for improving the efficiency and quality of service [7]. However very few of them address the question of discovering good quality clusters. In short, the work reported in this paper is unique and very few researchers have addressed the connectivity based distributed node clustering problem in such detail as we have done in this paper.

We plan to extend this work in a variety of directions. First, we want to experiment with graphs of various other topological structures to study how our scheme performs. The second line of research we want to pursue is to design variants of our scheme to suit specific needs of different networks like P2P networks with low bandwidth or networks with devices which have low battery power etc. We think it is not only feasible but also important to design variants of our scheme to suit specific constraints. Finally, we also plan to study in detail, the application of cluster information in various decentralized P2P systems such as P2P file sharing systems, sensor networks, mobile ad-hoc networks etc.

References

[1] G. Chen, F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic. Connectivity Based k-hop Clustering in Wireless Networks. In *35th Hawaii International Conference on System Sciences*, 2002.

[2] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[3] Freenet Home Page. http://www.freenet.sourceforge.com.

[4] Gnutella development page. http://gnutella.wego.com.

[5] W. Heinzelman, A. Chnadrakasan, and H. Balakrishnan. Energy-Efficient Coomunication Protocols for Wireless Microsensor Networks. In *Hawaii International Conference on System Sciences*, 2000.

[6] Kazaa home page. http://www.kazaa.com.

[7] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan. A Cluster-based Approach for Routing in Dynamic Networks, 1997. ACM SIGCOMM Computer Communication Review, April 1997.

[8] L. Ramaswamy, B. Gedik, and L. Liu. Connectivity Based Node Clustering in Decentralized Peer-to-Peer Networks. Technical report, College of Computing, Georgia Tech, 2003.

[9] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of International Conference on Peer-to-peer Computing*, August 2001.

[10] S. van Dongen. A New Cluster Algorithm for Graphs. In *281*, page 42. Centrum voor Wiskunde en Informatica (CWI), ISSN 1386-3681, 31 1998.