A Distributed Approach to Node Clustering in Decentralized Peer-to-Peer Networks

Lakshmish Ramaswamy, **Buğra Gedik** and **Ling Liu**

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{laks, bgedik, 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 **C**onnectivity-based **D**istributed Node **C**lustering 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. To cope with the typical dynamics of P2P networks, we provide mechanisms to allow new nodes to be incorporated into appropriate existing clusters and to gracefully handle the departure of nodes in the clusters. These mechanisms enable the CDC scheme to be extensible and adaptable in the sense that the clustering structure of the network adjusts automatically as nodes join or leave the system. We provide detailed experimental evaluations of the CDC scheme, addressing its effectiveness in discovering good quality clusters and handling the node dynamics. We further study the types of topologies that can benefit best from the connectivity based distributed clustering algorithms like CDC. 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.

Index Terms: Distributed node clustering, Connectivity based graph clustering, Peer-to-peer networks, Decentralized network management.

1 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 [17] and Kazaa [21] were the harbingers of this change, various other systems like mobile ad-hoc networks and P2P sensor networks have adopted this model of computation and communication.

Although these systems appear to be 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 highly 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, such as: (1) Scalable techniques for data discovery and peer look-up; and (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 2. 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 [17] or Freenet [15]. Let us examine how node cluster information can be utilized to design intelligent file replication schemes in 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. A discussion on the various other applications of node clusters in different P2P systems is provided in Section 6.4.

Although some researchers have applied clustering information to address certain key problems [8, 22] in decentralized P2P networks, very few of them 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 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. Further, as P2P networks are highly dynamic, efficient schemes are needed to incorporate newly entering nodes into the cluster structure and gracefully handle the exit of existing nodes in the network.

With these problems in mind, this paper presents a *C*onnectivity-based *D*ecentralized Node *C*lustering scheme (CDC), a scalable and an efficient solution for discovering connectivity based clusters in peer networks. Among the research questions addressed are:

1. Can we develop accurate and efficient algorithms and protocols to cluster the network of nodes in a completely distributed fashion? In other words, can we discover node clusters without ever constructing a complete view of the network?
2. How accurate can such distributed node clustering schemes be, when compared with centralized clustering algorithms?
3. If the nodes are allowed to arbitrarily enter and leave the system, what mechanisms can be adopted to handle such node dynamics, without having to re-cluster the whole network every time a node exits or enters the system?
4. How accurate and beneficial are these mechanisms when compared with the option of re-clustering the network on the entrance and the exit of each node?

In contrast to centralized graph clustering algorithms, the CDC scheme only requires 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. Another strength of the CDC scheme is the mechanisms to handle dynamics of nodes without resorting to re-clustering at each entry and exit. Our experimental results indicate that these schemes are not only efficient but also maintain high quality clusters. These mechanisms enable the CDC scheme to be extensible and adaptable in the sense that the clustering structure of the network may alter according to nodes joining or departing the system. A detailed experimental evaluation of the CDC scheme is provided, showing the effectiveness of the CDC scheme in discovering good quality clusters and handling the node dynamics. We have further studied the topologies that can benefit best from the connectivity-based distributed clustering algorithms like CDC.

2 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 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. The number of edges incident upon a node in graph G is termed as its *degree*. Two nodes V_i and V_j are termed *neighbors* if there is an edge $E_l = (V_i, V_j)$ connecting them in the graph. 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 graph G is said to be a *connected graph* if and only if (iff) for any two vertices V_i and V_j in G, there exists at least one path connecting them.

A *dissimilarity 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 $d(v, u) = 0$ iff $v = u$. Equivalently, a *similarity function* s can be defined on the vertices of a graph as a symmetric function from $V \times V$ to $\mathbb{R}_{\geq 0}$, such that $s(v, u) = \infty$ iff $u = v$. The similarity and dissimilarity functions may be appropriately defined according to the semantics of the graph under consideration and the application at hand.

The general problem of clustering on a set of data points is to partition the data point set into *natural groups* [20]. Intuitively, a natural grouping of data points can be thought of as a grouping based on the similarity measure between points in a group. Two points d_1 and d_2 belong to the same cluster if the similarity between them is high. Otherwise, the two are assigned to different clusters.

The Euclidean distance is the predominantly used similarity function for clustering data points in Euclidean spaces. However, similarity functions can be defined in a host of meaningful ways for clustering the nodes in a graph. 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 a collection of 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 dissimilar. Each 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. That is $Cl_l \cap Cl_m = \emptyset, \forall (Cl_l, Cl_m) \in Cl$.

For the graph clustering problem, the similarity can be defined in host of meaningful ways. Each of these definitions leads 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 a large number of paths connecting them.
- A random walk [24] 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 a 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** [35]. Like most existing graph clustering algorithms, both of them assume that 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. Unfortunately, most P2P systems promote decentralized network management, where each node knows only its neighbors and has no explicit knowledge of other nodes in the system. The challenge here is not only how we can find clusters under decentralized management, but also how accurate and efficient such algorithms are in finding good clusters.

3 CDC Scheme for Graph Clustering

In contrast to centralized graph clustering algorithms, the problem of distributed clustering assumes that each node has a 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 a node ever wants to reach another 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. For example, in the Gnutella network [17], each peer 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 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. The nodes and edges that are not in the dense structure would have relatively few people in them.
- Nodes that lie inside the cluster would have accumulated a higher weight when compared with the nodes that are remotely approachable from the originator.

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. However if a node did not receive enough weight from any node, then it decides to be an outlier (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.

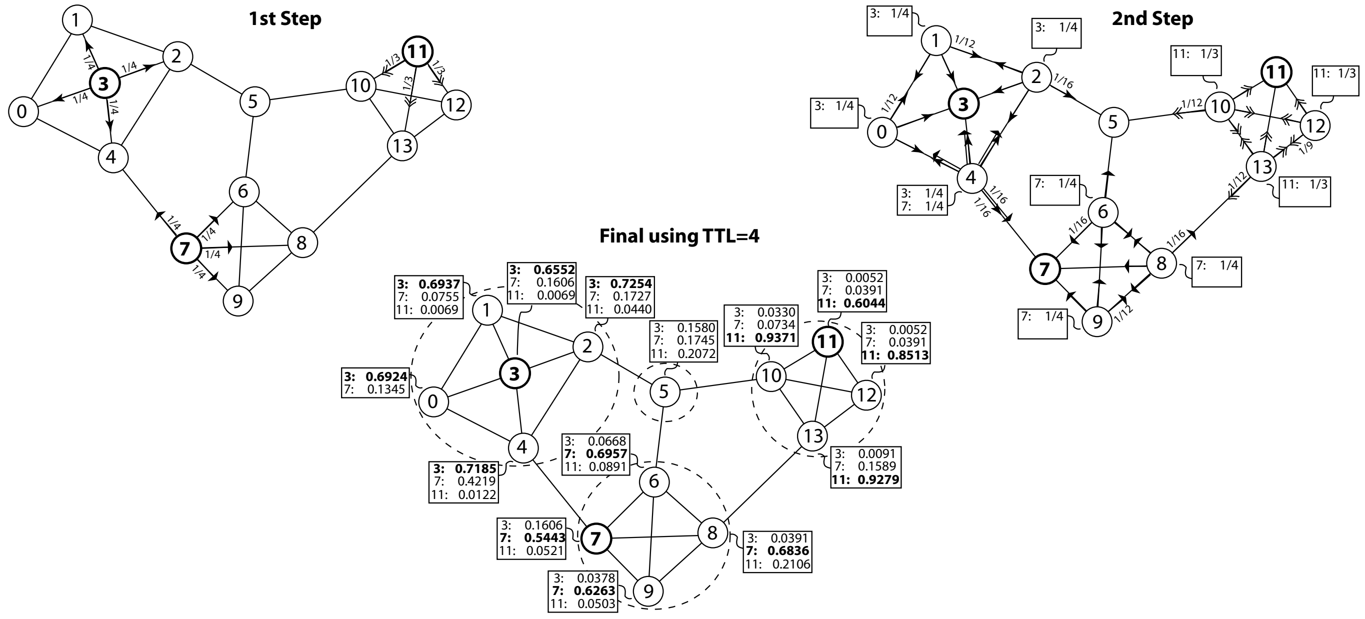

Figure 1: CDC Illustration

3.1 CDC Algorithms

In this section we concretize the algorithms of the CDC scheme and provide their pseudo-code. The 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 $O = \{O_1, O_2, ..., O_Q\}$. These originator nodes initiate the process of message circulation by sending out messages to all 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 3.2. 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. 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 $Msg.MWeight = \frac{1}{Degree(O_l)}$. The TTL field may be set to a small integer value. The only constraint is that all originator nodes should use the same TTL value.

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. Node V_i halts the message circulation if TTL is 0 or $MWeight$ becomes insignificantly low.

Each node may receive multiple messages from several different nodes. A node calculates the $TotalWeight$ function for each of the originators from which it received messages. After a node has received the last message, it waits for some time to ensure that there are no more messages to be processed. Then the node joins the cluster led by the originator, for which the value of $TotalWeight$ is the maximum. To join a particular cluster, the node sends a message to the originator informing the originator of its decision and its node-ID. If all $TotalWeight$ values lie below a predefined threshold, then the node remains an outlier. We note that a node that has already joined a cluster or has decided to remain an outlier may discover at a later point in time that it has accumulated higher weight from the messages from a different originator. In this case, if the node already belongs to a cluster, it informs the current cluster's originator about its decision to quit its cluster. It sends a message to the new cluster's originator notifying its decision to join its cluster.

```
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
```

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. A casual glance at the graph indicates that there are three different clusters in the graph. We illustrate how the algorithm works when the process is initiated by three originator nodes $\{3, 7, 11\}$. 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 $TotalWeights$ received from all originator nodes are less than the threshold and hence it becomes an outlier.

In this discussion we have omitted some subtle issues which are necessary for the correct execution of the scheme. We provide the pseudo-code for the CDC scheme in Algorithms 1 and 2. These algorithms 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, $\sum_{Msg.TTL=h \wedge Msg.OID=O_l}(Msg.MWeight)$ 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.

Though we have used a particular weight function in our scheme, a class of different graph clustering algorithms can be obtained by altering the ways in which the message weight is initialized by the originators and how they are modified by the message's recipients. For example if the scheme adopts a constant weight function wherein each message weight is always set to 1, we obtain the **distributed K-path algorithm**.

3.2 THP Originator Determination Scheme

The 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. We consider the same graph, which we used to illustrate the CDC algorithm. The diagrams in Figure 2 show four different scenarios, indicating the clusters we obtain when we start 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 $LastMsgID(O_l) == Msg.MID$ then
        $TotalWeight(V_i, O_l) \leftarrow TotalWeight(V_i, O_l) + Msg.MWeight$
    else
        $TotalWeight(V_i, O_l) \leftarrow Msg.MWeight$
        $LastMsgID(O_l) \leftarrow Msg.MID$
    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. In this scenario the single outlier in the scenario joins the cluster led by node 10, leaving three clusters and no outliers. The clusters we obtain in the other two scenarios are unintuitive and are in no way close to the ideal clustering in Scenario 1. 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 selecting good originators.

We briefly discuss the properties a good originator set should possess.

Property 1: 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. This is exactly what is happening in Scenario 3 of Figure 2. Here we see all three originators $\{2, 5, 6\}$ are neighbors and are concentrated in one single region. The effect of this is that the nodes $\{10, 11, 12, 13\}$ remain as outliers.

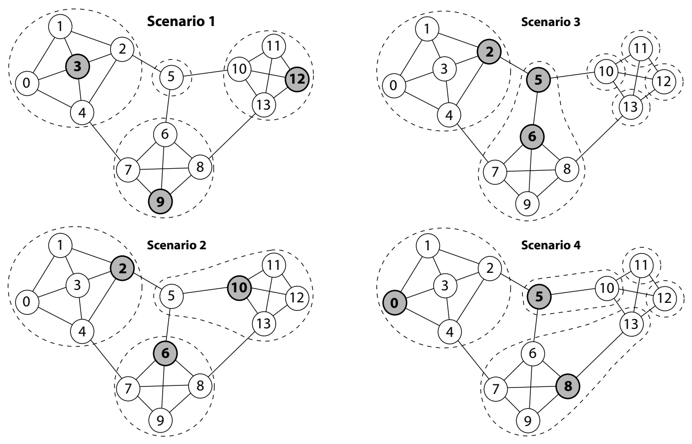

Figure 2: Importance of Good Originators

Although, the above condition is necessary for good clusters, it is not sufficient. Scenario 4 in Figure 2 is a testimony to this fact. Here we see the originators $\{0, 5, 8\}$ are spread out in all the regions of the graph. But still the clusters obtained are not good. Clearly the set of originators needs to possess some more important properties.

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. This observation leads us to the second essential property of good originators:

Property 2: 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. $TotalWeight(V_l, V_l) \geq TotalWeight(V_l, V_i), \forall V_i \in V$.

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 pair of vertices. Further, the $TotalWeight$ function cannot be determined until 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. In Figure 2, originator sets $\{2, 6, 10\}$ and $\{3, 9, 12\}$ satisfy both criteria and lead to good quality clusters.

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, and hence, the node opts not to become an originator. Otherwise, the node 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 then the node chooses to be an originator. Otherwise, the node will not become an originator. The pseudo-code for the THP scheme is provided in Algorithm 3. The two configurable parameters for this algorithm are the $Vicinity$ factor and the $TwoHopThreshold$ factor. 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 its neighbors. Hence this scheme is very efficient in terms of messaging cost.

```
Algorithm 3: Determine whether node $V_l$ is an Originator
  Wait for random time in anticipation of cluster messages from other originators
  Check if $V_l$ received any cluster message with $MSG.TTL > InitialTTL - Vicinity$
  if $V_i$ received message with $MSG.TTL \geq InitialTTL - Vicinity$ then
    {There is an originator in the vicinity}
    $V_l$ not an originator. Do not initialize messages
  else
    {No originators in the vicinity, Compute TwoHopProb}
    $TwoHopProb(V_l) \leftarrow \sum_{V_i \in Nbr(V_l)} (\frac{1}{\|Degree(V_l)\| \times \|Degree(V_i)\|})$
    if $TwoHopProb(V_l) < TwoHopThreshold$ then
      {$TwoHopReturnProbability$ too low}
      $V_l$ not a originator. Do not initialize messages
    else
      $V_l$ is an originator. Initialize Messages
    end if
  end if
```

The configurable parameters of the CDC algorithm and the THP originator determination mechanism can be used to tune the algorithm such that the resulting clusters obey certain constraints that may be required by some applications. For example, the performance of an application may be affected by the number of outlier-nodes in the network, in which case the parameter $WeightThreshold$ of the CDC algorithm can be tuned for optimal performance of the application. Increasing the $WeightThreshold$ parameter leads to an increase in the number of outliers, and vice-versa. Similarly, the parameters $Vicinity$ and $TwoHopThreshold$ of the THP mechanism can be used to control the number of clusters produced by our scheme. Increasing the values of these parameters reduces the number of discovered clusters, and vice-versa.

A couple of subtle issues have to be addressed in order to complete the discussion of the distributed clustering algorithms. The *first* issue is when and how does the clustering process start? As we discuss in the next section, the graph has to be re-clustered when a substantial number of nodes enter or exit the system, in order to maintain high quality clusters. The clustering process may be initiated by any one of several ways. The first method would be to re-cluster the graph at regular intervals of time. When a node enters the network, the bootstrapping process informs the node of the time interval between successive re-clusterings. At the end of an interval, each node waits for a small, random time to see whether it receives any clustering messages from its neighboring node. If at the end of this small, random wait it has not still received any clustering messages, it initiates the THP mechanism. In the second method, any of the originators, on observing that a substantial number of nodes belonging to its cluster exit the system or a considerable number of new nodes enter its cluster, may flood the network with a clustering-initiation message. In order to avoid very frequent re-clustering of the network, the system specifies a minimum re-clustering interval. In the third method, the system can have a special bootstrapping node that initiates the clustering process. In this case, the originators which want network re-clustering send their requests to the bootstrapping node. When the fraction of nodes requesting re-clustering exceeds a preset threshold, the bootstrap node floods the network with cluster-initiation message. Each of the methods has its pros and cons. The first method is a good choice since it is easy to implement and has minimal message overhead. We think that this method in conjunction with mechanisms for handling node dynamics (discussed in the next section) would be sufficient to maintain good quality clusters between successive re-clusterings.

The *second* issue that needs to be addressed is regarding where the clustering information is maintained and how it is used in message-routing schemes. The originator node of every cluster maintains a list of its neighboring clusters that contains information such as the neighboring cluster's originator-ID, shortest path to that cluster, etc. This information can be obtained by border nodes (nodes in a cluster that have a direct link to neighboring clusters). The originator nodes use this to devise efficient communication strategies to improve performance of the system. The exact manner in which this information is employed in designing efficient communication mechanisms is dependent upon the P2P system at hand. For example, the super peers in Kazaa or the new Gnutella protocol maintain similar information about a small subset of ordinary peers. As a result, the CDC originator determination mechanism can be used to guide the selection of super peers in super-peer-based P2P networks such as Kazaa and the new Gnutella. Furthermore, in CDC an originator replicates all the information it maintains regarding the neighboring clusters and the nodes belonging to its cluster at its immediate neighbor nodes. This information replication provides fault tolerance in case the originator exits the network or fails.

4 Handling Dynamics of Nodes

Nodes in most P2P networks are dynamic and may enter and exit the system at arbitrary times. One of the challenges we had listed was whether we could design efficient and effective algorithms to handle this dynamism without re-clustering the graph on the entry and exit of each node. The need for efficient schemes for this problem is evident when we consider the messaging cost of the CDC scheme and the highly dynamic nature of some P2P networks. In this section we provide effective solutions to this problem. The solutions are based on heuristics, which are easy to compute and which make use of only local information.

4.1 Node Entry Mechanism

Problem Statement: Suppose we have a graph $G = (V, E)$, which is already clustered. Let the clusters be represented as $Cl = \{Cl_1, Cl_2, ..., Cl_Q\}$. Now a new incoming node V_{N+1} has to be incorporated into this cluster structure. The information available in the incoming node V_{N+1} is limited to its neighbors (represented as $Nbr(V_{N+1})$) in G. This node has to be added on to one of the existing clusters or has to be made an outlier.

The solution we provide is based on obtaining an approximate value for the $TotalWeight$ function used in the CDC algorithm. Our scheme calculates how much the incoming node V_{N+1} is "attracted" to each of the exiting clusters. The node joins the cluster which attracts it the most. If there is no such cluster, then it chooses to be an outlier. The attraction of V_{N+1} to various clusters is calculated as follows. For each node $V_j \in Nbr(V_{N+1})$ the *Neighbor-Attraction* function is defined as $NbrAttraction_{V_{N+1}}(V_j) = \frac{1}{Degree(V_j)}$. This indicates how much V_{N+1} towards V_j. Now the attraction of V_{N+1} towards a cluster Cl_l is the total of the Neighbor-Attraction values of all its neighbors which belong to Cl_l, which is represented as $ClustAttraction(Cl_l) = \sum_{V_i \in Cl_l \wedge V_i \in Nbr(V_{N+1})} NbrAttraction_{V_{N+1}}(V_i)$.

The algorithm works as follows. The node V_{N+1} obtains two sets of information from each of its neighbors, namely the degree of the node and the cluster to which it belongs. Then the node V_{N+1} calculates the $ClustAttraction$ value for all clusters which contain at least one neighbor of V_{N+1}. The node joins that cluster which attracts it the most, provided its $ClustAttraction$ is higher than a preset threshold. Otherwise V_{N+1} remains as an outlier.

We note that one or more neighbors of the node V_{N+1} might themselves be in the process of entering the network, in which case they would not be associated with any cluster, or, they might be in the process of exiting the network. In these scenarios, such neighbors do not return a valid cluster-ID when probed by V_{N+1} (for example they might return -1 as their cluster-ID). The node V_{N+1} ignores all those neighbors which returned invalid cluster-IDs when it calculates the $ClustAttraction$ values for various clusters.

4.2 Node Exit Mechanism

Problem Statement: Suppose we have a graph $G = (V, E)$ with N nodes and M edges, which is already clustered. Let the clusters be represented as $Cl = \{Cl_1, Cl_2, ..., Cl_Q\}$. Now a node V_k in this cluster exits the network, which is noticed by only the neighbors of V_k. The problem is to update the cluster structure to reflect this node's exit.

Our scheme to handle node exit is again completely distributed and localized. As the exit of any node is recognized only by its neighbors, they execute the algorithm to update the cluster. The crux of the algorithm can be captured as follows. Suppose V_k and V_{k+1} are neighbors in the graph G, and the node the node V_k

exits the graph. If V_{k+1} belongs to a different cluster than V_k, then it performs no action other than updating its connectivity information. On the other hand, if V_{k+1} and V_k belong to the same cluster, then we adopt the same strategy as adding a new node. The node V_{k+1} acts as though it has just entered the system and calculates the $ClustAttraction$ parameter between itself and various existing clusters. It joins that cluster to which it has the maximum attraction. In other words, on calculating the new $ClustAttraction$ values, if the node V_{k+1} discovers that it is attracted to another cluster more than the one it currently belongs to, then it "defects" to that cluster. The pseudo-code of the scheme is provided in Algorithm 4.

Algorithm 4: Algorithm executed by the node V_{k+1}, a neighbor of a node V_k that exited the system

```
if $ClusterID(V_k) \neq ClusterID(V_{k+1})$ then
  Nothing needs to be done.
else
  $CurrClustID \leftarrow ClusterID(V_{k+1})$
  Obtain $ClusterID$ and $Degree$ of each of its neighbors
  $CurrClustAttraction \leftarrow$
        $\sum_{V_i \in CurrClustID \wedge V_i \in Nbr(V_{k+1})} \frac{1}{Degree(V_i)}$
  $NbrClusterId \leftarrow \bigcup_{V_i \in Nbr(V_{k+1})} ClusterID(V_i)$
  $MaxAttraction \leftarrow 0$
  for Each $Cl_j \in NbrClusterId$ do
    $ClustAttraction(Cl_j) \leftarrow \sum_{V_i \in Cl_j \wedge V_i \in Nbr(V_{N+1})} \frac{1}{Degree(V_i)}$
    if $ClustAttraction(Cl_j) \geq MaxAttraction$ then
      $MaxAttraction \leftarrow ClustAttraction(Cl_j)$
      $MaxClustID \leftarrow Cl_j$
    end if
  end for
  if $MaxAttraction \geq CurrClusterAttraction$ then
    Defect to $MaxClustID$
  else
    Remain in the same cluster.
  end if
end if
```

We observe special cases might occur when a node exits the network while its neighbor is still in the process of entering the network, or when two nodes exit the network simultaneously. In the first case, suppose the node V_{k+1} is still in the process of determining its cluster, when it detects the exit of V_k. Then the entering node V_{k+1} re-computes the $ClustAttraction$ values towards various clusters ignoring V_k, and joins the cluster with the maximum $ClustAttraction$. In case of simultaneous node exits, suppose the node V_{k+1} detects that two or more of its neighbors exit the network simultaneously. Then V_{k+1} performs the cluster-attraction test if at least one of the exiting nodes belonged to its cluster.

The exit of an originator needs special handling as it contains the cluster and the routing information. As any other node, the originator either informs its immediate neighbors before it exits the system or just fails in which case the failure is again detected by its neighbors. The neighbors of the originator adopt a leader election strategy to elect a new originator. A simple yet effective strategy would be to elect the neighbor with the highest number of in-cluster edges as the new originator. This node stays in the cluster and informs all other nodes in the cluster that it would be the new originator of the cluster.

These two algorithms are only approximations to the CDC scheme. As we see in the experiments, the clustering degrades if a large percentage of nodes join or leave the network. Hence the network has to be re-clustered periodically. These schemes are designed to handle node-dynamics between re-clusterings. Frequency of re-clustering is specific to the needs of the P2P network and the node dynamics. The tradeoff is between the accuracy of clustering and the clustering message load on the network.

5 Extending the CDC Scheme for Weighted Graphs

The algorithms we have discussed above for distributed node clustering and handling entry and exit of nodes assumed that the graph was un-weighted. Therefore the clustering was purely based on the connectivity of the nodes in the P2P network. However, performance of some P2P applications may be optimized by considering the weights of the edges when clustering the nodes of the underlying network. In this section we briefly explain how the CDC scheme can be extended for weighted networks.

Before explaining extensions to the algorithms, we discuss an important issue regarding the edge-weights in graphs, as they can have a profound impact on the resulting clusters. The weights associated with the edges in the network may represent a wide variety of parameters such as the distance between the nodes, the bandwidth of the link, the latency of the link, etc. While in some cases the weight quantifies the *similarity* between the nodes, in others the edge-weight represents the *dissimilarity* of the nodes. For example, while the distance or the link-latency quantifies the dissimilarity, the link-bandwidth can be construed as an approximate measure of the node similarity. Without loss of generality, in this discussion we assume that the weight of each edge quantifies the similarity between its end-vertices. If the edge-weight W_i corresponds to a dissimilarity measure, then $\frac{1}{W_i}$ can be used as the corresponding similarity measure. We now have $G = (V, E, W)$ as a weighted graph, where W is a symmetric similarity function from $V \times V$ to $\mathbb{R}_{\geq 0}$ such that $W(V_i, V_j) > 0$ if the edge (V_i, V_j) exists in the graph G, and $W(V_i, V_j) = 0$ otherwise. We assume that the weight $W(V_i, V_j)$, of an edge (V_i, V_j), can be estimated by the end vertices and is available at V_i and V_j.

We now discuss how each algorithm of the CDC scheme can be extended to weighted graphs. The structural outline of these algorithms remains essentially the same, even when extended to weighted graphs. However, the Message-weight ($Msg.MWeight$), the Two hop return probability ($TwoHopProb$) and the Neighbor attraction ($NbrAttraction$) functions have to be adopted for weighted graphs.

Extending the Basic CDC Algorithm

Recall that in the CDC algorithm, the originator O_l initializes the message weight of the message as $Msg.MWeight = \frac{1}{Degree(O_l)}$ and sends the message to each of it neighboring nodes. For weighted graphs, this is modified to reflect the edge weight as follows: The message weight carried by a message sent to a neighboring node V_i depends on the weight of the link between V_i and O_l and is initialized as $Msg_{V_i}.MWeight = \frac{W(O_l,V_i)}{\sum_{V_j \in Nbr(O_l)} W(O_l,V_j)}$. Similarly the recipient of a message Msg_{V_i}, say node V_i, takes into account the edge weights when initializing new messages. The message-weight of the new message being sent to a neighboring node V_k is initialized as $NewMsg_{V_k}.MWeight = Msg_{V_i}.MWeight \times \frac{W(V_i,V_k)}{\sum_{V_j \in Nbr(V_i)} W(V_i,V_j)}$. We note here that the above suggested modification scales the message weight in proportion to the weight of the link between the sender and the recipient of the message.

Extending the THP Originator Selection Scheme

In the case of the THP originator determination scheme, the two hop return probability should be scaled to reflect the weights of the links in the network. Accordingly we propose the following formula to calculate the two hop return probability: $TwoHopProb(V_l) = \sum_{V_i \in Nbr(V_l)} (\frac{W(V_l,V_i)}{\sum_{V_j \in Nbr(V_l)} W(V_l,V_j)} \times \frac{W(V_i,V_l)}{\sum_{V_h \in Nbr(V_i)} W(V_i,V_h)}) = \sum_{V_i \in Nbr(V_l)} (\frac{W(V_l,V_i)^2}{\sum_{V_j \in Nbr(V_l)} W(V_l,V_j) \times \sum_{V_h \in Nbr(V_i)} W(V_i,V_h)})$. The THP scheme itself remains unchanged except using the formula to compute the two hop return probability.

Extending the Mechanisms to Handle Node Entry and Exit

As we discussed in section 4, in our scheme, a node entering the network, say V_{N+1}, calculates the affinity towards various clusters ($ClustAttraction$) by computing the sum of the attractions towards its neighbors ($NbrAttraction$) belonging to those clusters. In order to adopt the node entry mechanism to the weighted networks, we modify the neighbor-attraction function so that the link-weights are taken into account. The neighbor-attraction function between the entering node V_{N+1} and its neighboring node V_j is now calculated as $NbrAttraction_{V_{N+1}}(V_j) = \frac{W(V_{N+1},V_j)}{\sum_{V_k \in Nbr(V_{N+1})} W(V_{N+1},V_k)}$. The entering node V_{N+1} computes the $ClustAttraction$ values towards the existing clusters by summing the $NbrAttraction$ values for the neighbor nodes belonging to those clusters and joins the cluster with the maximum $ClustAttraction$ value.

When a node exits the system, the neighbors detect its exit and each node belonging to the same cluster as the exiting node acts as though it just entered the system. It computes $ClustAttraction$ parameter between itself and various clusters using the new function for $NbrAttraction$ and defects to another cluster if it discovers that it is attracted to that cluster more than its current cluster.

This section discussed how the CDC scheme can be easily extended to cluster nodes in weighted graphs. In the rest of the paper we restrict our discussion to un-weighted networks.

6 Experiments and Results

This section reports the experimental evaluation of the proposed schemes. We begin by discussing the metric used for measuring graph clustering accuracy.

6.1 Accuracy Measure for Graph Clustering

Measuring the accuracy of a given clustering on a 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/dissimilarity measures. In this paper we use an intuitive performance measure proposed by [36] termed as the *Scaled Coverage Measure*.

The motivating idea behind this performance measure is that an optimal clustering of a given graph minimizes both the number of inter-cluster edges and the number of non-neighbor vertices in each cluster.

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. $Clust(V_i) = Cl_l$ iff $V_i \in Cl_l$.
- **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:

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

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 Coverage Measure of all of its nodes.

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

We use this measure in our experiments to quantify the clustering quality and compare different clusters.

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.

6.2 Experimental Data Sets

For our experimental evaluation, we have used three datasets, which we now briefly describe.

Live Gnutella Data: This dataset was captured by taking a snapshot of the Gnutella P2P network on Jan 06 2003. The total number of nodes in the snapshot was 1043, which we estimate to form 10% of the complete network. However, in Gnutella there are two types of nodes, namely *normal nodes* and *super nodes*. A normal node can be connected only to super nodes. On the other hand a super node can have one or more neighbors, which may be either normal nodes or other super nodes. It is these super nodes that play a crucial role in search and message broadcasts. So we would like to discover cluster information on these super nodes. Therefore, we eliminate all the single-neighbor nodes and retain only the super nodes.

Power Law Topology: We use power law distribution to generate graphs that resemble P2P data sharing networks in their topological structure. Studies in the past few years on the topology of the Internet [14], and more recently on P2P data sharing networks [30] have revealed that the topology of such networks closely follows 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, 5000 and 10,000 nodes.

Range Topology: The range topology models the connectivity relationship in wireless/sensor networks. In wireless networks two computing units have knowledge of the existence of the other, only if they fall in each other's radio range. Range topology graphs model this phenomenon. A range graph is generated as follows. We consider a square area of unit dimension. Nodes are randomly thrown at this unit square. Each node represents a wireless unit, and has a range-distance associated with it. Two nodes are connected by an edge if they fall in each other's range-distance.

Parameter	Gnutella Data	Power Graphs	Range Graphs
Total Nodes	184	5000	5000
Total Edges	307	11446	27083
Average Degree	3.34	4.57	10.83
Maximum Degree	38	623	25
Minimum Degree	2	1	1
Variance in Degree	18.89	212.53	11.08

Table 1: Parameter Values for Various Topology Graphs

Table 1 lists some of the important properties of all the three kinds of datasets.

6.3 Experimental Results

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

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 Dongen [35] 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 various values from 1.2 to 5. We measure the accuracy of the clusters obtained, and use the highest value as our benchmark.

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.

Figures 3 and 4 indicate the clustering accuracy of CDC scheme, the centralized clustering scheme and the Distributed K-path clustering scheme on range and power topology graphs respectively. The CDC scheme performs better than the K-path clustering scheme for all graph sizes. For a range graph with 500 nodes, the CDC scheme yields an accuracy value of 0.462, whereas distributed K-path gives 0.413, which is an improvement of almost 12%. For a graph with 10,000 nodes the improvement is around 11%.

However the centralized MCL scheme performs better than the CDC scheme. The centralized MCL scheme beats the CDC scheme by almost 14% and 10.5% for range graphs with 500 and 10,000 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

In this set of experiments we demonstrate the effectiveness of THP originator selection mechanism. Figures 5 and 6 indicate the clustering accuracy of the Two-Hop Probability originator determination scheme and compare it with the clustering accuracy of the CDC scheme with random originator selection and centralized MCL scheme on range and power topology graphs respectively. The results show that THP originator determination scheme improves the clustering accuracy of the CDC scheme considerably. For example, on a range graph of 1000 nodes, the THP originator mechanism yields a mean accuracy value 0.574 as against 0.457 given by the CDC scheme with random originators, which amounts to an improvement of over 25%.

However, the CDC scheme coupled with the THP originator selection mechanism does not always perform better than the centralized clustering scheme. For example on a random graph of 200 nodes and 4800 edges, our scheme yields a clustering accuracy value of 0.1422 whereas the MCL scheme gives an

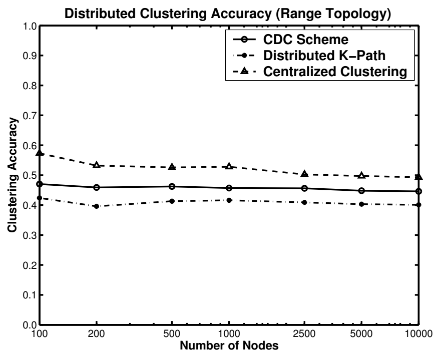

Figure 3: Accuracy of CDC scheme on Range Graphs

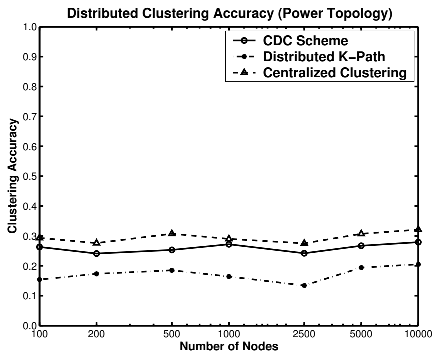

Figure 4: Accuracy of CDC scheme on Power Graphs

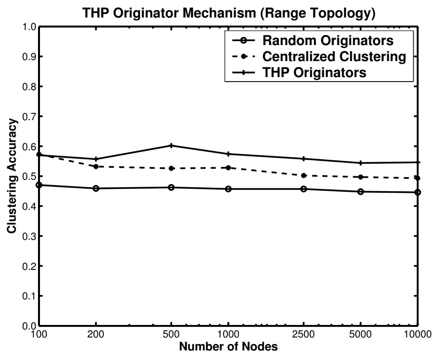

Figure 5: Accuracy of THP scheme on Range Graphs

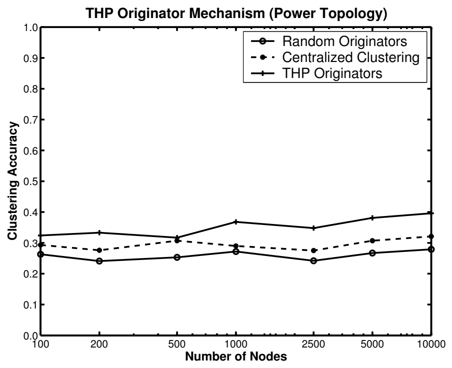

Figure 6: Accuracy of THP scheme on Power Graphs

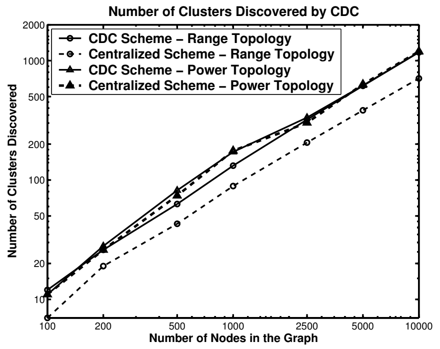

Figure 7: Number of Clusters Discovered in Range and Power Graphs

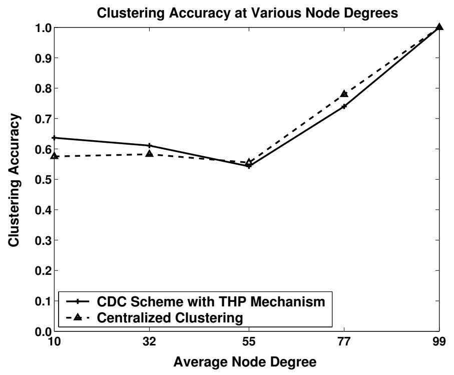

Figure 8: Effect of Node Degree on Accuracy

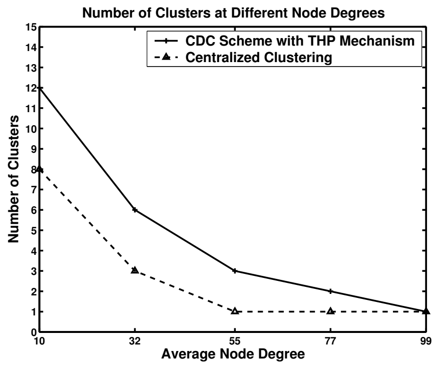

Figure 9: Effect of Node Degree on Number of Clusters

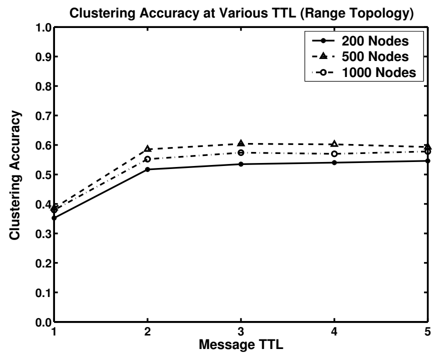

Figure 10: Accuracy at Various TTL for Range Graphs

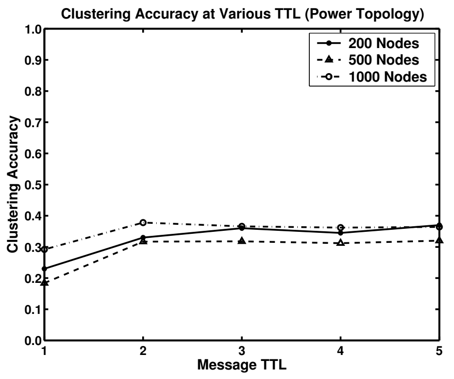

Figure 11: Accuracy at Various TTL for Power Graphs

accuracy value of 0.2355. Another important point to be noted here is that our scheme performs well, irrespective of the number of nodes in the graph. In other words it does not deteriorate for graphs with higher number of nodes.

In the next experiment we demonstrate the stability of the THP originator selection mechanism in choosing approximately same number of originators upon multiple executions. Table 2 shows the statistics on the number of originators selected by the THP mechanism on power, range and random topology graphs of 5000 nodes, when it is executed 500 times. We see that

Network Topology	Mean	Coeff. of Variation	Maximum Value	Minimum Value
Range Topology	616.63	0.009	632	602
Power Toplogy	620.96	0.006	629	604
Random Toplogy	640.79	0.012	669	630

Table 2: Statistics Indicating Stability of THP Mechanism

in all three cases the variation in number of originators is confined to a narrow range. The number of originators discovered in each execution lies within a small range around the mean value. For instance, the maximum width of this range is 39 for random graphs. Also, the coefficient of variation (ratio of standard deviation to the mean) of the number of originators is very low. These results show that the THP mechanism indeed selects a stable number of peers as originators.

Figure 7 shows the number of clusters discovered by the CDC scheme coupled with the THP originator selection mechanism and the MCL scheme for range and power graphs with nodes ranging from 100 to 10,000. Both X and Y axes are on the log scale. We note that for range graphs the number of clusters discovered by the CDC algorithm has been consistently higher than the number of clusters discovered by MCL, whereas for power graphs the numbers of clusters discovered by both schemes are close to each other. As both the CDC scheme and the MCL scheme are aimed at clustering the nodes of the network based on their connectivity, their performance depends upon the density of the connections (edges) in the network. In order to study the relationship between the density of connections and the accuracy of clusters, we evaluated the two schemes for a range graph of 100 nodes. In this set of experiments we varied the number of edges in the network (or in other words the average degree of the nodes in the graph).

Figure 8 indicates the clustering accuracy of the CDC and the MCL schemes when the average degree of the nodes varied from 10 to 99 (for a graph of 100 nodes the maximum node degree is 99, when the graph becomes strongly connected). When the average node degree is 10, the CDC scheme coupled with the THP originator mechanism yields an accuracy value of 0.6367, whereas the MCL scheme gives an accuracy value of 0.5753. As the average node degree increases, we observed a fall in the clustering accuracy of both schemes. The fall in the accuracy value of our scheme is steeper than that of the MCL scheme. When the average node degree is 55, the MCL scheme overtakes the CDC scheme and yields a slightly better accuracy value. From then on this trend continues. When the average node degree reaches 99 (i.e. when the graph becomes fully connected), both schemes group all the nodes into a single cluster giving accuracy value of 1.00. We did not expect the CDC scheme to perform better than the centralized graph clustering approach. We feel that this behavior is due to the approximations employed in the centralized MCL algorithm and the advanced originator selection scheme employed in the THP mechanism.

Figure 9 indicates the number of clusters in the graph detected by the CDC and the MCL algorithms for a graph of 100 nodes, when the average node degree varied from 10 to 99. We see the number of clusters detected by the CDC scheme is always higher than the number of clusters detected by the MCL algorithm. The numbers of detected clusters in both schemes falls with increasing average node degree for

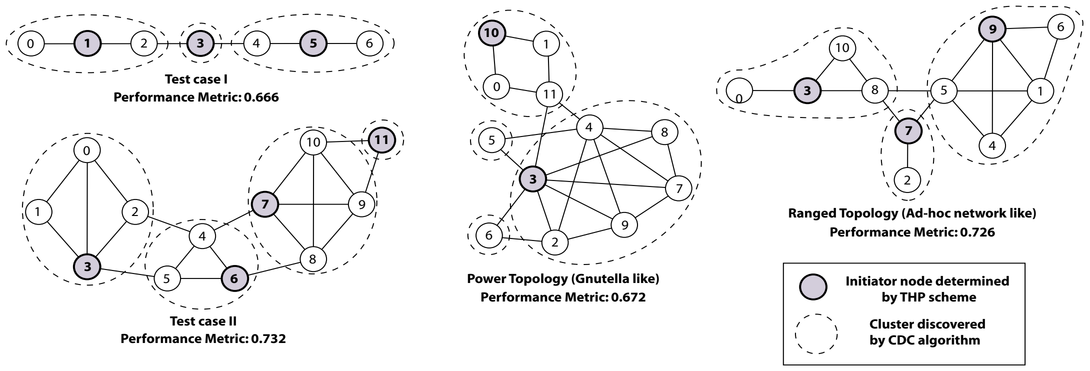
Test case I
Performance Metric: 0.666
Test case II
Performance Metric: 0.732
Power Topology (Gnutella like)
Performance Metric: 0.672
Ranged Topology (Ad-hoc network like)
Performance Metric: 0.726
Initiator node determined by THP scheme
Cluster discovered by CDC algorithm

Figure 12: Illustration of CDC Scheme on Various Example Graphs

both schemes. The MCL algorithm discovers the entire graph as a single cluster when the average node degree is greater than 77. The CDC algorithm, on the other hand, detects 2 clusters on the graph with average node degree 77 and a single cluster for a graph with average node degree 99.

Table 3 shows the results of the CDC scheme on actual Gnutella data. It can be observed that the results are very similar to those of the power topology graphs. The clustering accuracy value is in general low for graphs that are similar to power topology graphs. In this case we see that the CDC scheme yields a slightly higher accuracy value than the centralized scheme.

Parameter	Value
Total Nodes	184
Centralized Clustering Accuracy	0.246
CDC Scheme with Random Originators	0.158
CDC Scheme with THP Originators	0.257
Number of Cluster (CDC Scheme)	21
Number of Outliers (CDC Scheme)	6

Table 3: CDC Scheme performance on Gnutella

Figure 12 provides illustrations of the CDC scheme on various example graphs indicating the originators and the clusters discovered. In short, these experiments demonstrated that a seemingly simple mechanism for originator selection can improve the clustering accuracy of the CDC scheme by large amounts making the scheme perform similar to, or in some cases, better than the centralized MCL algorithm.

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 determine the effect of Initial-TTL on the clustering accuracy, we obtained various clusters by setting the Initial-TTL values from 1 to 5. The Figure 10 and 11 indicate the accuracy values of range topology and power topology graphs with 200, 500 and 1000 nodes.

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.

Message Cost of CDC Scheme

Now we evaluate the CDC scheme from a message cost standpoint. In our experiments, we use the total number of messages as an indicator of the message cost.

The number of messages needed for the CDC scheme is dependent upon the Initial TTL value employed by the scheme. Therefore we have measured the total number of messages generated by the CDC scheme with Initial-TTL set to various values. We compare the CDC algorithm with the scheme where each node floods the network with messages indicating its neighbors. In this scheme each node in the network constructs a complete view of the graph and executes a clustering algorithm for discovering the clusters. We term this as the flooding scheme.

Figure 13 shows the number of messages generated by all the three schemes on range topology graphs. Both X and Y axes are on the log scale. A couple of interesting and important points emerge from this graph. First, the total number of messages generated by the CDC scheme is an order of magnitude less than those generated by the flooding approach. For example for a graph of 500 nodes, the flooding approach needs 22 times the number of messages needed for the CDC scheme with Initial-TTL set to 4. Second, as both axes are on log scale, the total number of messages needed for the CDC scheme increases on a linear scale with respect to the total number of nodes in the network. Third, the rate at which the network load increases for the CDC scheme at various Initial-TTL values decrease as TTL gets higher. It might be observed that the CDC scheme with Initial-TTL set to 4 generates slightly higher number of messages than the CDC with Initial-TTL set to 3. In order to further demonstrate this phenomenon, we tabulate the number of messages generated for range graphs at various TTL values in Table 4. For a range graph of 5000 nodes, the message cost when TTL is set to 7 is just 1.7% more than the message cost for the same graph when TTL is set to 4. For a network of 10,000 nodes, the message cost increases by only around 1.56% when the TTLs of the clustering messages are increased from 4 to 7. These experimental results show that even for larger networks the rate at which message cost of the CDC scheme grows is very small at high TTL values. The reason for this is that as a message executes more hops, the weight of the message keeps reducing rapidly and the scheme drops the message if its weight is insignificant, even if the message's TTL has not expired (in our experiments a message is dropped if its weight drops below 1×10^{-5}). We also note that in the Gnutella P2P file sharing system the TTL of a query is usually set to 7. This experiment shows that not only is the CDC scheme efficient but it also scales well, and hence, can be applied to very large networks.

Number of nodes	TTL = 3	TTL = 4	TTL = 5	TTL = 7
1000	130210	182097	183620	186640
5000	808302	977653	983911	994832
10000	1980340	2554640	2579986	2594492

Table 4: Total number of messages circulated at various TTL values

Node Entry Accuracy

We now evaluate the performance of the mechanisms to handle the dynamics of nodes. First, we present the experimental evaluation of the node entry mechanism.

For this experiment, we consider a pre-clustered graph, to which new nodes are added. Each entering node knows only its neighbors and joins the cluster structure through the mechanism we have discussed in Section 4.1. We measure the accuracy of the clustering at regular intervals in order to evaluate the performance of the scheme. We compare this scheme with the option of re-clustering the graph each time a node enters the pre-clustered graph.

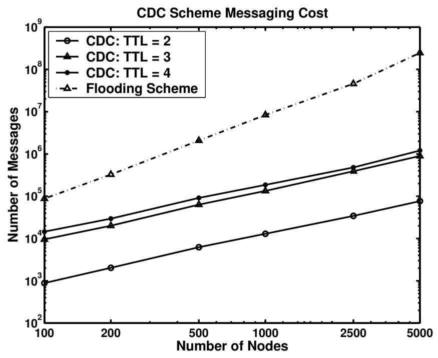

Figure 13: CDC Scheme Message Costs

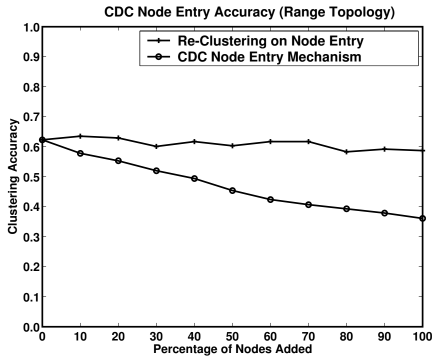

Figure 14: Accuracy of Node Addition Scheme on Range Graphs

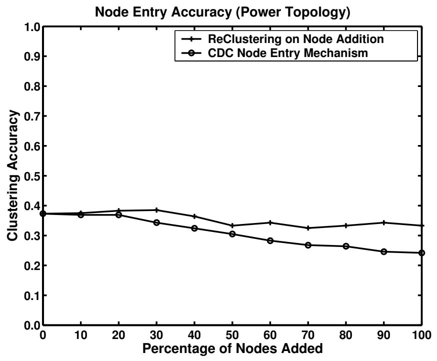

Figure 15: Accuracy of Node Addition Scheme on Power Graphs

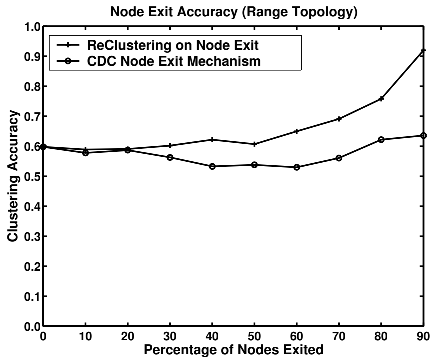

Figure 16: Node Exit Scheme Accuracy on Range Graphs

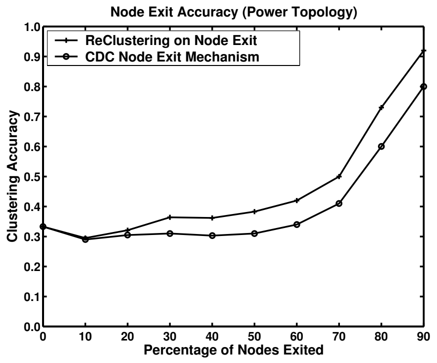

Figure 17: Node Exit Scheme Accuracy on Power Graphs

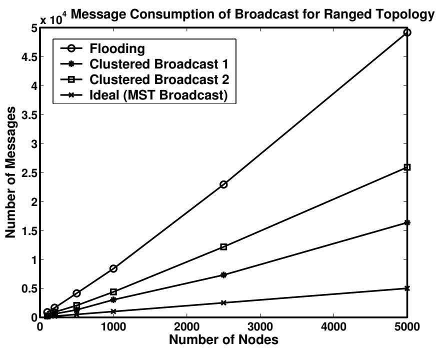

Figure 18: Efficient Broadcast with Distributed Clustering

Figures 14 and 15 show the accuracy of the node addition scheme and the accuracy of re-clustering the graph on node addition for range and power topology graphs respectively. The X-axes in the graphs indicate the percentage of the newly added nodes and the Y axes the clustering accuracy. We see that for a range graph when 10% new nodes join the system the accuracy of our scheme is 0.578. If the graph were to be re-clustered it yields an accuracy value of 0.635. The accuracy value shows a linear deterioration with a small gradient until the number of nodes in the system increases to 1.3 times the number of nodes in the original graph. Thereafter the descent is faster. When the graph doubles in the number of nodes, the accuracy value falls to 0.361 as against a value of 0.586 for the re-clustering scheme. Nevertheless, this scheme for node entry saves messaging costs by large amounts. Therefore this is a very useful and an attractive scheme and yields acceptable performance unless the graph size changes very drastically, at which point the graph has to be re-clustered.

Node Exit Accuracy

Now we evaluate the accuracy of the mechanism we have proposed to handle node exits. In these experiments, we consider a pre-clustered graph. The nodes exit this graph in a random order. When each node exits the graph, its neighbors detect the exit and handle the node exit as described in Section 4.2. As in the previous case we compare our scheme with the option of re-clustering the graph at each node exit.

Figures 16 and 17 indicate the accuracy of the node exit mechanism for range and power graphs with 500 nodes respectively. It can be seen that until the percentage of nodes that have exited is below 20%, our scheme performs almost equally well as that of re-clustering the graph on each node exit, after which our scheme shows minimal degradation in clustering accuracy.

After 60% of nodes exit the system, accuracy values of both schemes start to increase. This phenomenon can be explained as follows. When a large percentage of nodes exit the system, the graph splits up into clear clusters, which are connected by very few edges. Hence clustering algorithms yield higher accuracy values. The increase in accuracy values of re-clustered graph is more pronounced than those of our scheme.

In these experiments we have demonstrated that our mechanisms to handle the entry and exit of nodes maintain the accuracy of the clusters even when considerable percentage of nodes join or leave the system.

6.4 Applications of CDC

Discovering connectivity-based node clusters can improve the efficiency of resource utilization in many decentralized P2P systems. P2P file sharing networks and wireless ad-hoc networks are such potential areas of interest that can benefit from a clustering approach [25, 26, 33, 41].

Network-wide broadcast is a useful mechanism that is frequently used to support ad-hoc routing in wireless networks (as in Ad Hoc On-demand Distance Vector and Zone Routing Protocol), and query lookups in P2P file sharing networks. Simple flooding [19] is one way of performing network-wide broadcast.

Flooding based broadcast results in several duplicate messages being sent which are simply discarded. As a result, it incurs a high messaging cost. Note that an ideal solution to the broadcast problem is to have a spanning tree that can be used to broadcast data without any overhead. However, it is not feasible to maintain such a tree in a dynamic network due to frequent changes required to the tree. A compromise between messaging cost and maintenance cost is to use clustered broadcast. Although using clusters in wireless ad-hoc networks has been proposed before [28, 37, 38], clusters suggested in these approaches are limited when compared to our connectivity-based clusters.

An interesting way of performing network-wide broadcast in wireless networks is to use a version of clustered broadcast, what we call *forest broadcast*. Briefly described, forest broadcast uses per cluster trees for broadcasting data within clusters, and flooding for conveying messages between clusters. Figure 18 compares the message cost for network-wide broadcast using forest broadcast (with two different granulations of clusters) compared to flooding and spanning tree based ideal broadcast.

7 Related Work

The problem of graph clustering has received considerable attention from researchers in algorithmic graph theory. Several algorithms have been proposed for general graph clustering [11, 31, 32, 35]. Out of these algorithms the two most significant ones from a practical view point have been the k-path and the MCL clustering algorithms [13, 35].

The k-path clustering has been one of the earlier algorithms proposed for graph clustering. The similarity function adopted in the k-path clustering is the number of paths of length k between the nodes of the graph. Though the k-path clustering is easy to implement and produces good clusters when the graph has distinct dense regions, its main drawback is that it acts on a global scale and tends to ignore the local variations in the node connectivity.

The MCL algorithm introduced by [35] alleviates the above problem of the k-path clustering. The similarity function used in the MCL algorithm is based on the network flow. The MCL algorithm 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. This algorithm addresses one of the major problems of the k-path clustering of being insensitive to the local variations in the graph connectivity.

The PageRank algorithm [27] applies the principle of random walks on a graph for ranking web documents. This scheme views the entire web as a graph with web pages forming the nodes of the graph and the hyperlinks forming its edges.

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. In addition, we provide mechanisms for addition and deletion of nodes into pre-clustered networks.

Distributed node clustering has been studied in the context of wireless-mobile ad-hoc networks. Most of the works related to ad-hoc networks consider node clustering as a mechanism to regulate communication in such networks. In ad-hoc networks research, node clustering has been employed to decrease the power consumption of nodes through the use of energy-efficient protocols for medium access control (MAC) [23, 42], data collection [3], routing [9], and mobility handling [4, 5, 6].

Most of the prior work on distributed node clustering in ad-hoc networks have focused on constructing one-hop clusters [3, 4, 5, 6, 23]. In a one-hop cluster, each node is at most one-hop away from its clusterhead. A few exceptions to this line of work are [2] and [9]. In [2], a heuristic-based distributed algorithm is introduced for building clusters in which each node is at most d hops away from its clusterhead, where d is a system parameter. The algorithm tends to create clusterings in which clusters have approximately the same size. On the contrary, CDC creates clusters that reflect the connectivity structure of the underlying network, and thus can have arbitrarily sized clusters. In [9], several distributed clustering algorithms are proposed for constructing k-hop clusters. These algorithms differ from the CDC scheme in the fact that they use the node connectivity information only for the purpose of electing clusterheads and not for the clustering process itself. This difference has a direct impact on the clusters discovered by these algorithms. For instance, CDC will partition the nodes within a highly connected region of the network into a small number of large clusters, whereas the algorithms presented in [9] will form large number of small clusters.

An important concept related with decentralized node clustering is *dominating sets*. A dominating set of an undirected graph $G(V, E)$ is a subset of nodes $S \subset V$, such that for every node $v \in V$, either $v \in S$ or $\exists u \in S$, such that $(u, v) \in E$. In other words, every node in the graph that is not included in the dominating set, is one hop reachable from the dominating set. Considering nodes that belong to a dominating set as clusterheads and assigning each non-clusterhead node to its nearest clusterhead, a valid clustering of nodes in the graph can be constructed. In many scenarios, like routing, it is advantageous to have connected clusterheads. Pertaining to this, a subset of nodes $S \in V$ is called a *connected dominating set* if it also forms a connected subgraph of G in addition to being a dominating set. A relaxation of the connectedness requirement results in *weakly connected dominating sets*. A weakly-connected dominating set of a graph $G(V, E)$ is a subset of nodes $S \subset V$, such that S is a dominating set, and for each node $v \in S$, there exists another node $u \in S$ which is at most 2 hops away. Unfortunately, finding minimum dominating sets as well as connected or weakly connected dominating sets are shown to be *NP-Complete* problems.

However, a number of approximation algorithms have been proposed in the literature [1, 10, 12, 39]. Alzoubi et al. [1] have proposed a localized distributed algorithm for finding connected dominating sets in a network. The algorithm has a constant approximation ratio as well as linear time and linear message complexity in terms of the network size. Chen and Stojmenovic [10] describe centralized and distributed algorithms for finding weakly connected dominating sets in a network. The distributed algorithms provide $O(log\Delta)$ approximation, where Δ is the maximum degree of the network, and has $O(|V| \times |S|)$ time and message complexity. Dubhashi et al. [12] have proposed several localized distributed algorithms for finding connected or weakly connected dominating sets. These algorithms execute in polylogarithmic number of steps in terms of the network size, and provide an approximation ratio of $O(log\Delta)$. Wu and Li [39] present an algorithm to update the connected dominating set of an ad-hoc wireless network, when the network topology changes due to entry, exit or movement of its nodes.

The above schemes 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 standpoint. In contrast, CDC scheme is entirely based on connectivity structure of the network, and hence, leads to high quality clusters.

In addition to the above discussed literature, the area of P2P systems in general has received considerable attention from the research community in recent years. The research in P2P systems includes characterization and modeling of the overlay network and traffic patterns in P2P systems such as [18, 30], novel architectures, data structures and algorithms for efficient and scalable search and lookup [7, 29, 34], comparative study of various architectures and algorithms [40] and designing new P2P applications [16].

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.

8 Conclusion

We have presented CDC − the connectivity-based distributed node clustering scheme for accurately clustering nodes in decentralized peer-to-peer networks. Our scheme is completely decentralized and does not require a global view of the network structure. The scheme can either cluster the entire network automatically or detect clusters around a given set of nodes. In addition, we have also proposed schemes to efficiently and effectively incorporate new nodes into an existing cluster structure and handle the exit of nodes in the clusters. Our experiments indicate that our approach yields good clusters and effectively handle the node dynamics.

Our work on CDC continues along several directions. First, we want to experiment with graphs of various other topologies to study how our scheme performs. Second, we plan to design variants of the CDC scheme to suit specific needs of different networks, like P2P networks with low bandwidth, networks with devices which have low battery power, etc. These networks may place special constraints on the clusters such as limiting the number of clusters, limiting the maximum number of nodes in any cluster, and so forth. We believe that it is not only important, but also feasible to design variants of the CDC scheme to meet constraints of specific overlay networks. Finally, we are interested in studying the benefits and costs of applying the CDC node clustering scheme to various decentralized overlay networking systems such as sensor networks, mobile ad-hoc networks, and other community-based collaborative networking applications.

References

[1] K. M. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad-hoc networks. In *ACM MobiHoc*, 2002.

[2] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong. Max-Min D-Cluster formation in wireless ad hoc networks. In *IEEE INFOCOM*, 2000.

[3] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *IEEE INFOCOM*, 2003.

[4] S. Basagni. Distributed clustering for ad hoc networks. In *I-SPAN*, 1999.

[5] C. Bettstetter and R. Krausser. Scenario-based stability anlysis of the distributed mobility-adaptive clustering (DMAC) algorithm. In *ACM MobiHoc*, 2001.

[6] M. Chatterjee, S. Das, and D. Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *Journal of Cluster Computing*, 5, April 2002.

[7] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *ACM SIGCOMM*, 2003.

[8] G. Chen, F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic. Connectivity based k-hop clustering in wireless networks. In *Hawaii International Conference on System Sciences*, 2002.

[9] G. Chen and I. Stojmenovic. Clustering and routing in mobile wireless networks. Technical Report TR-99-05, School of Information Technology and Engineering, University of Ottawa, June 1999.

[10] Y. Chen and A. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad-hoc networks. In *ACM MobiHoc*, 2002.

[11] Drineas, Frieze, Kannan, Vempala, and Vinay. Clustering in large graphs and matrices. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1999.

[12] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. In *ACM SODA*, 2003.

[13] J. Falkner, F. Rendl, and H. Wolkowitz. A computational study of graph partitioning. *Mathematical Programming*, 66(2):211–239, 1994.

[14] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM*, 1999.

[15] Freenet home page. http://www.freenet.sourceforge.com.

[16] B. Gedik and L. Liu. PeerCQ: A decentralized and self-configuring peer-to-peer information monitoring system. In *IEEE ICDCS*, 2003.

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

[18] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling and analysis of a peer-to-peer file sharing workload. In *SOSP*, 2003.

[19] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999.

[20] A. K. Jain, M. N. Murthy, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3), 1999.

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

[22] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan. A cluster-based approach for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review*, pages 49–65, April 1997.

[23] C. R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal of Selected Areas in Communications*, 15(7), 1997.

[24] L. Lovasz. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2, 1996.

[25] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad hoc sensor networks. In *OSDI*, 2002.

[26] S. R. Madden, R. Szewczyk, M. J. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Workshop on Mobile Computing and Systems Applications*, 2002.

[27] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[28] W. Peng and X. Lu. Efficient broadcast in mobile ad hoc networks using connected dominating sets. *Journal of Software*, 1999.

[29] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM*, 2001.

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

[31] T. Roxborough and A. Sen. Graph clustering using multiway ratio cut. In *International Conference on Graph Drawing*, 1997.

[32] R. Sablowski and A. Frick. Automatic graph clustering. In *International Conference on Graph Drawing*, 1996.

[33] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *ACM MobiCom*, 1998.

[34] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, 2001.

[35] S. van Dongen. A new cluster algorithm for graphs. In *281*, page 42. Centrum voor Wiskunde en Informatica (CWI), ISSN 1386-3681, 31 1998.

[36] S. van Dongen. Performance criteria for graph clustering and markov cluster experiments. Technical report, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, 2000.

[37] P. Wei and L. Xi-Cheng. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Workshop on Mobile Ad Hoc Network Computing*, 2000.

[38] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *ACM MobiHoc*, 2002.

[39] J. Wu and H. Li. A dominating-set-based routing in ad hoc wireless networks. *Telecommunication Systems*, 18(1-3), 2001.

[40] B. Yang and H. Garcia-Molina. Comparing hybrid peer-to-peer systems. In *VLDB*, 2001.

[41] Y. Yao and J. E. Gehrke. Query processing in sensor networks. In *Conference on Innovative Data Systems Research*, 2003.

[42] W. Ye, J. Heidemann, and D. Estrin. An energy efficient MAC protocol for wireless sensor networks. In *IEEE INFOCOM*, 2002.