SDC: A Distributed Clustering Protocol

Yan Li yan.li@engr.uconn.edu
Computer Science and Engineering
University of Connecticut
Storrs, CT 06269, USA

Li Lao llao@google.com
Google Santa Monica
Santa Monica, CA 90401, USA

Jun-Hong Cui jcui@engr.uconn.edu
Computer Science and Engineering
University of Connecticut
Storrs, CT 06269, USA

Abstract

Network clustering is an important technique used in many large-scale distributed systems. Given good design and implementation, network clustering can significantly enhance the system's scalability and efficiency. However, it is very challenging to design a good clustering protocol for networks that scale fast and change continuously. In this paper, we propose a distributed network clustering protocol SDC targeting large-scale decentralized systems. In SDC, clusters are dynamically formed and adjusted based on SCM, a practical clustering accuracy measure. Based on SCM, each node can join or leave a cluster such that the clustering accuracy of the whole network can be improved. One big advantage of SDC is it can recover accurate clusters from node dynamics with very low message overhead. Through extensive simulations, we conclude that SDC is able to discover good quality clusters very efficiently.

Keywords: network clustering, distributed algorithm, Scaled Coverage Measure, SDC, dynamic network

1. INTRODUCTION

Clustering is an important technique studied in various areas, such as biology, chemistry, linguistics, physics, and sociology. The basic goal of clustering is to group data in such a way that data in the same cluster shares certain similarity. In this paper, we study one interesting type of clustering: *network clustering*, which partitions a network topology into clusters so that nodes in the same clusters are highly connected and between clusters are sparsely connected.

Network clustering has become an important technique in different networking research areas. With a good network clustering algorithm, we can design scalable and efficient routing protocols [13] [3] [1] [23], enhance scalability and efficiency of large-scale distributed systems [16] [2] [21] [9], and resolve many critical networking issues such as virus spreading [26] [15] [32], QoS [19] [18], network robustness [6] [4] [12] [25], to name a few. In [22], network clustering is used to study the clustering features of the AS-level Internet topology and a realistic topology model is designed based on the observed clustering features.

Network clustering can be performed in both centralized and distributed ways. Centralized network clustering is an off-line procedure, in which complete network topology information is required. Thus, centralized clustering is usually used for small networks or off-line data analysis. In our work, we focus on distributed clustering techniques, which are designed for large-scale distributed systems.

To design a good network clustering protocol, we must consider the following design criteria. First of all, as a natural requirement of network clustering, nodes in the same clusters should be highly connected, and less connected between clusters. Secondly, a good clustering protocol should control cluster size (or cluster diameter) well. Thirdly, the number of "orphan" nodes should be minimized. Lastly, a good distributed clustering protocol should take node dynamics into account, especially when the clustering targets are highly dynamic with frequent node entry and exit. We provide a detailed discussion on clustering criteria in Section 2.

In the literature, there has been considerable research effort addressing the problem of network clustering, but very few of them studied the problem in the scenario of large-scale distributed networks. Among the existing approaches, MCL [28] is well accepted as an efficient and accurate network clustering algorithm. However, this approach assumes that complete network topology is available at one central point, which makes it difficult to apply MCL into distributed systems. CDC [27], on the other hand, is a distributed algorithm. It forms clusters based on node connectivity. The main issue with this algorithm is that it can not handle node dynamics in a decent way: a large number of messages must be exchanged to keep accurate clusters.

With these problems in mind, we design a novel network clustering protocol: **SCM-based Distributed Clustering (SDC)**, which satisfies all the design criteria mentioned above. In SDC, clusters are dynamically formed and adjusted based on a practical clustering accuracy metric, Scaled Coverage Measure (SCM) [29]. In SDC, each network node makes its own decision to join or leave a cluster whenever clustering accuracy can be improved. To control cluster size, TTL (Time-To-Live) is piggybacked in exchanged messages to guarantee cluster diameter does not exceed a predefined threshold. SDC is a fully decentralized protocol which requires only neighbor information, and it can handle node dynamics with small message overhead while keeping good quality of clusters. Besides the basic protocol design, we also address some difficulties in scenarios of distributed networks. A common and critical issue addressed in this paper is deadlock, which is caused by simultaneous node actions. We provide some strategies to avoid and resolve deadlock conditions and analyze their effects on the performance of the protocol. Through extensive simulations, we can conclude that our proposed protocol, SDC, is able to discover high quality clusters in a very efficient way.

The rest of this paper is structured as follows. In Section 2, we introduce the background of network clustering and review some related work. In Section 3, we discuss an important concept, SCM which is the basis of our design. Then in Section 4, we present our clustering protocol SDC in detail. We show the performance of SDC by extensive simulations in Section 5. At the end of this paper, we conclude our work in Section 6.

2. BACKGROUND AND RELATED WORK

In this section, we formulate the network clustering problem and introduce a set of criteria for desired clustering approach. Then we review several existing clustering methods.

2.1 Network Model

A targeting network to be clustered can be presented as a connected, undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges connecting network nodes. Let $|V| = n$ and $|E| = m$. Then a partition $C = \{C_1, C_2, \ldots, C_k\}$ of V is named as a clustering C of graph G, and C_i represents the i^{th} cluster. Each cluster must be a non-empty subset of V. Clearly, $\bigcup_{i=1}^{k} C_i = V$. The diameter of a cluster C_i is defined as the maximum length of the shortest paths between all

pairs of nodes in C_i. Accordingly, if a cluster has one node, its diameter is 0. We call the clusters with diameter equal to 0 as orphan nodes.

Another metric associated with a cluster is cluster size that is defined as the number of nodes in a cluster. Cluster size and cluster diameter are closely related. In most scenarios, "control cluster size" is equivalent to "control cluster diameter". We only differentiate these two metrics in the protocol description.

2.2 Criteria of Clustering

The network clustering problem can be formulated as finding a "good" clustering C in G such that C can accurately describe the natural clustering features in the topology. More specifically, in a "good" partition C, the intra-cluster node connectivity should be maximized and the inter-cluster node connectivity should be minimized. Therefore, node connectivity is a basic criterion to be considered in network clustering design. In addition to node connectivity, cluster size is another important metric. In large-scale distributed networks, due to the lack of knowledge about network structure, it is expensive to maintain expanded clusters. In other words, cluster diameters should be carefully controlled. A good network clustering algorithm should also take the number of orphan nodes into consideration. In most scenarios, orphan nodes are not preferred as they violate the goal of clustering and should be eliminated.

As discussed above, node connectivity, cluster diameter, and orphan nodes are important criteria for good network clustering algorithms. However, more issues need to be addressed when we cluster large-scale distributed systems. In such networks, a node only has the knowledge about its neighbors and may join or leave the network at any moment. To obtain a complete view of the network structure, a huge number of messages need to be exchanged to collect the topology information. Moreover, the obtained topology may expire very soon due to node dynamics. Re-collecting topology information on each node-entry and node-exit will overload the network with a huge amount of exchanged messages. Therefore, it is not feasible to maintain a complete and up-to-date topology in such networks. Given these concerns, a good clustering protocol for large-scale distributed systems should form clusters in a fully distributed fashion, i.e., nodes should form clusters automatically without the requirement of complete network topology, and it should be able to recover accurate clusters from node dynamics with small overhead in term of the number of exchanged messages.

2.3 Related Work

Significant research efforts have been devoted into design of network clustering methods [5] [10] [14] [17] [28] [27]. However, many existing clustering algorithms assume that the complete network topology is available at a central point. One typical research line tries to solve the MINIMUM k-CLUSTERING problem which is formulated as follows: Given a network topology G and an integer k, find a partition of G into a smallest number of l subsets so that the diameter of each subset is at most k. The MINIMUM k-CLUSTERING is proved to be NP-Complete in simple and undirected graphs [10], so most of research efforts are focused on special types of topologies. One representative work is presented in [10], which proposes a polynomial time approximation algorithm, DDP, for graphs with dominating diametral path. DDP does not form clusters based on node connectivity, so it can not guarantee accurate clustering.

The Markov Cluster (MCL) [28] is a connectivity-based centralized network clustering algorithm. The basic idea behind this algorithm is flow simulation. In this algorithm, an input graph G is mapped in a generic way onto a Markov matrix. Then the set of transition probabilities are iteratively recomputed via matrix expansion and inflation. An infinite sequence consisting of repeated alternation of expansion and inflation constitutes a new algebraic process called the Markov Cluster (MCL) process. The heuristic behind this algorithm is that a flow between sparsely connected dense regions evaporates after MCL process. Therefore, it is easy to detect dense regions in the original graph which are the output clusters. MCL algorithm can achieve high clustering accuracy. However, due to its centralized feature, it can not be used in distributed systems.

There have been many proposals for network clustering in large-scale decentralized systems [8, 11, 20, 30, 31]. Among existing decentralized clustering algorithms, one representative work is [27], a connectivity based distributed network clustering algorithm, CDC, which is designed for p2p networks. In CDC, a set of peers are selected as "originators" and clusters are discovered around these peers by TTL-controlled message flooding. If the "originators" are well distributed in the network, clusters with good quality can be formed. The CDC scheme is a fully distributed approach and the cluster size can be effectively controlled by TTL. The main issue of CDC is the selection of "originators" which can affect the accuracy of clustering significantly. So far, there is no good solution to well distribute "originators". Thus, the clustering accuracy can not be guaranteed. Another issue with CDC is it can not efficiently handle node dynamics. To maintain good clustering quality, the whole network has to be re-clustered at each node join or leave, which introduces a huge amount of message overhead.

In summary, there is no existing clustering method which can satisfy all the criteria for network clustering in large-scale distributed systems. In this paper, we propose a novel network clustering protocol, SDC. It is fully distributed and can form high quality clusters in highly dynamic systems with small message overhead.

3. SCALED COVERAGE MEASURE

Before introducing our protocol, we first discuss Scaled Coverage Measure (SCM), a practical metric to evaluate the accuracy of connectivity-based clustering algorithms proposed by S.Van Dangon [28].

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

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

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

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

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

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

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

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

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

Thus,

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

and

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

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

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

4. SCM-BASED DISTRIBUTED CLUSTERING: SDC

SDC is a fully distributed clustering protocol. A node in the network maintains only its own state information: 1) the cluster it currently belongs to, identified by a unique id $clust\_id$, 2) the current number of nodes in its cluster $clust\_size$, 3) the two parameters for SCM computation b_{v_i} and a_{v_i}.

To cluster a network from scratch, every node is initialized as an orphan node with its own $clust\_id$ (any unique identifier) and $clust\_size$ that is equal to 1. For any node v_i, the two parameters for SCM computation, b_{v_i} and a_{v_i}, are initialized as $deg\ (v_i)$. Then every node starts its own clustering procedure independently at a random time by sending requests to its neighbors. If the requests are accepted by the neighbors, the clustering procedure continues with a few rounds of message exchange until the node joins a cluster. Otherwise, the node can select to serve its neighbor's requests or start a new round of clustering. Clustering of individual node is an independent and local procedure. The clustering of the whole network ends when no message is exchanged. Next, we describe the protocol in detail.

4.1 Protocol Description

In SDC, nodes form clusters in a greedy manner based on SCM. Each node tries to cluster with a subset of neighbors which leads to a higher SCM value than cluster with the other neighbors. When a node is actively involved in a clustering procedure, it is either in "Clustering" mode, i.e. it starts the clustering procedure, or in "Serving" mode as it is serving another node's clustering, but not both. The clustering procedure of any node, v_i, involves the exchange of a set of messages elaborated as follows.

- **Clust_Request** . A node v_i starts its own clustering procedure by broadcast $Clust\_Request$ message to all the neighbors. Once the message is sent out, v_i is in the "Clustering" mode and waits for the response from its neighbors. A timer is set up to control the waiting time. During this waiting period, v_i can not accept and serve the clustering requests from other nodes. It buffers all the received requests and handles them after the current clustering. If all the neighbors response before timer expires, v_i confirms all the replies and the clustering procedure continues. If the timer expires and no response is received, v_i checks the buffered requests and selects one to serve. If the buffer is empty, v_i restarts its clustering procedure after a small random time period.
- **Clust_Reply**. Upon receiving $Clust\_Request$ from the neighbor v_i, node v_j sends back a $Clust\_Reply$ message if it is not actively involved in any clustering procedure. Otherwise, it refuses the clustering request by sending a $Clust\_Refuse$ message back to v_i as explained next. By sending out a $Clust\_Reply$, v_j claims its willingness to serve the clustering request of v_i and waits for confirmation from v_i. The $Clust\_Reply$ message

carries the current cluster information of v_j such as $clust\_id$, $clust\_size$ and $\Delta SCM(v_j)$ that shows the gain in $SCM(v_j)$ assuming node v_i joins the cluster of v_j if it is not in $Clust(v_j)$ or leaves $Clust(v_j)$ otherwise. The computation of $\Delta SCM(v_j)$ only requires the knowledge of whether v_i and v_j are directed connected. Specifically, let us consider the case that v_j is in a different cluster from v_i. If v_j is a neighbor of v_i,

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

On the other hand, if v_j is not directly connected with v_i,

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

The gain in SCM for nodes in the same cluster of v_i is computed in a slightly different way and can be easily derived.

- **Service_Confirm**. After receives replies from its neighbors, a requesting node needs to confirm the acceptance of the service provided by the neighbors. A $Service\_Confirm$ is sent back to the neighbor for this purpose. Once receives $Service\_Confirm$, a node can not serve others or request clustering for itself.
- **Clust_Lock**. A node that decides to serve a neighbor's clustering may receive $Clust\_Reply$ from its neighbors for its previous requests. This situation happens if a node requests for clustering service but none of it's neighbors are available to serve it at the moment. Those neighbors will buffer the request as mentioned above. After a while, some of the neighbors may become available again and try to serve the buffered requests. When a node that is in the serving mode receives a $Clust\_Reply$ for its previous requests, it replies with a $Clust\_Lock$ message which indicates it is serving others and can not accept the service offer.
- **Clust_Refuse**. If a node is in either "Clustering" or "Serving" mode when it receives a $Clust\_Request$ message, it informs the requester its unavailability for the request by sending a $Clust\_Refuse$ message. If a node is refused by all the neighbors, it either serves one buffered request if there is any or re-starts a new round of clustering procedure after a small random time interval.
- **Clust_Confirm**. If node v_i receives a $Clust\_Reply$ from neighbor v_j during the waiting period after it broadcasts a request, it confirms the service offer by sending back v_j a $Clust\_Confirm$ message and starts waiting for more $Clust\_Reply$ messages from the other nodes in $Clust(v_j)$. When node v_j receives the $Clust\_Confirm$, it enters the "Serving" mode and forwards the request message to all the other nodes in its current cluster through flooding. When another node v_k in $Clust(v_j)$ receives the request, it sends a $Clust\_Reply$ with its own $\Delta SCM(v_k)$ back to the request node v_i.
- **Clust_Reject**. To control the cluster granularity and the number of exchanged messages, every $Clust\_Request$ message carries a TTL field. Based on the value of TTL, a node can determine whether the cluster diameter will exceed a predefined threshold after v_i joins. If the TTL expires, v_j stops forwarding $Clust\_Request$ and sends a $Clust\_Reject$ message to v_i. Once receiving $Clust\_Reject$, v_i does not take $Clust(v_j)$ as a potential cluster to join.
- **Clust_Update**. After node v_i receives $Clust\_Reply$ from all the nodes in its current cluster and the neighbor cluster C_l (in the case that no $Clust\_Reject$ is received from C_l), it computes the overall gain $\Delta SCM(G)$ based on the received information. We use Δ_{leave} and Δ_{join} for the gain in SCM as if v_i leaves its original cluster and joins C_l. Then the overall $\Delta SCM(G)$ is computed as:

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

where Δ_{join} is the sum of the gain in SCM received from all the nodes in v_i's current cluster and Δ_{leave} is the sum of the received gain in SCM from all the nodes in the neighbor cluster C_i. If $\Delta SCM > 0$, v_i should join C_i. There might be multiple clusters for which ΔSCM are positive, v_i should join the one corresponding to the maximum ΔSCM. Once v_i determines which cluster to join, a $Clust\_Update$ message containing v_i's node id and its original $clust\_id$ is flooded in its original cluster and the new cluster it is joining. Then, v_i and every node receiving $Clust\_Update$ need to update the clustering information.

After v_i joins the new cluster, its neighbors in the other clusters are affected. These nodes will check whether they should move to v_i's cluster for a higher SCM in the same way as node v_i does. The whole clustering procedure ends if there is no exchanged message.

4.2 Handling Deadlocks

A critical issue that distributed network protocols must handle is how to detect and resolve deadlocks. In SDC, nodes may enter deadlock conditions if they start the clustering request simultaneous as their neighbors. When deadlock occurs, none of them can get served as they are all waiting for each other's replies. Deadlocks can cause the involved nodes to be in a busy waiting state infinitely, so those nodes will never get clustered.

We using the following two methods to avoid and resolve deadlocks.

- State Reset: If a node is in a waiting state which can be waiting for clustering reply, service confirm, etc., it will enter the next state automatically after a certain amount of time whether or not it receives replies from its neighbors. For example, a node sends $Clust\_Request$ to its neighbors. It starts a timer right after the requests are sent. When the timer times out, the node enters the corresponding next state based on whether or not it receives any $Clust\_Reply$. The length of the timer can be estimated based on RTT and the predefined TTL.
- Randomization: A direct reason for the deadlock scenario is the simultaneous node actions. Therefore, we introduce a randomized delay for each node before it takes actions. This randomization can affect the performance of SDC significantly. A short randomized delay may not be effective to resolve the deadlock condition while long delays are more effective but can slow down the whole clustering procedure. To analyze the effects, we provide simulation evaluations in the next section.

4.3 An Example

A simple example is shown in Fig. 1 to illustrate the SDC clustering procedure. In this example, TTL is set to 2, so the diameter of any cluster will not exceed 2. In Fig. 1a, node 0 wants to be clustered with other nodes. It first sends $Clust\_Request$ messages to all of its neighbors. Each neighbor node upon receiving the $Clust\_Reply$ sends its $clust\_id$, $clust\_size$ and SCM gain back to node 0 as shown in Fig. 1b. After receiving replies from all the neighbor, node 0 sends $Clust\_Confirm$ to accept the service offer so that its requests are forwarded to every node in the clusters of A and B via flooding, as shown in Fig. 1c. At the same time, node 7 also starts its clustering procedure by sending a $Clust\_Request$ to its neighbor 2. Since node 2 is in the "Serving" mode, the request from node 7 is refused. Thus node 7 waits for a small period of time before the next clustering attempt. In clusters A and B, every node which receives $Clust\_Request$ computes its SCM gain and sends $Clust\_Reply$ back to 0 (Fig. 1d). Node 0 then computes $\Delta SCM(G)$ based on the received information and joins Cluster A at the end (Fig. 1e). Since node 4 is affected by node 0's clustering, it starts its own clustering procedure in the way as node 0 does (Fig. 1f).

4.4 Handling Node Dynamics

In large-scale distributed systems, node entry and exit can occur at arbitrary time and the network structure may change continuously. Node dynamics can degrade the existing clusters and must

be handled by the clustering protocol. Re-do the whole clustering procedure may recover good clustering accuracy. However, it is very inefficient and the procedure may never stabilize if node entry and exit happens frequently. Therefore, designing an effective and efficient scheme to handle node dynamics is a critical demand for distributed clustering.

Our SDC protocol can naturally handle node dynamics in a decent way. Whenever a new node v_i joins the system, it is first initialized as an orphan node and gets its own $clust\_id$ and $clust\_size$ (which is 1). Since the network structure between node v_i and its neighbors is changed, a **Join** message carrying v_i's $clust\_id$ is issued by v_i to all of the neighbors so that they can update their SCM. As v_i's joining changes its neighbors' connectivity, the affected neighbor nodes should perform a new round of clustering procedure. When a node wants to leave, it sends a **Leave** message to each of its neighbors as well as every other node in its cluster through flooding so that the $clust\_size$ and SCM values of the affected nodes can be updated. This will also activate a new round of clustering procedures at these affected nodes. The logic behind this scheme comes from the fact that node entry and exit are localized events and only a few nodes are affected and need to be re-clustered.

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

5. SIMULATION EVALUATIONS

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

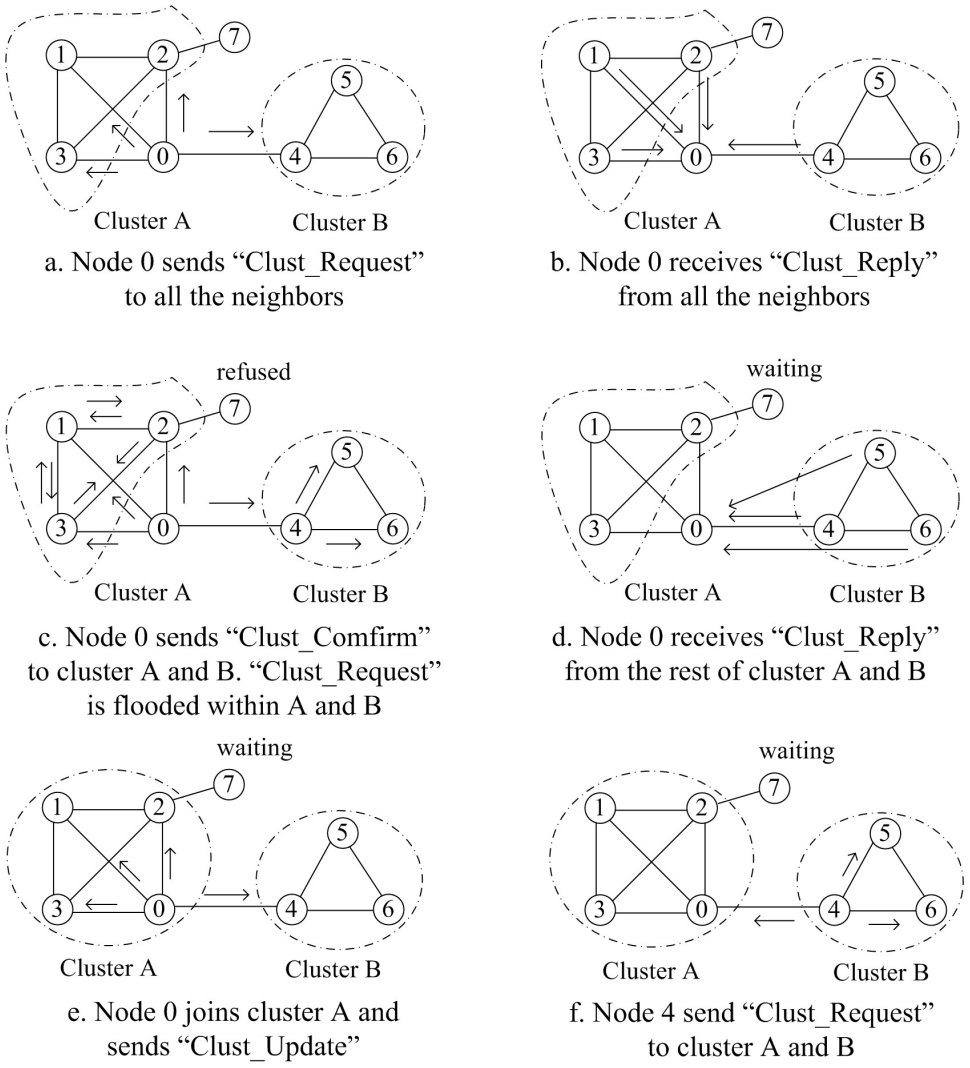


Figure 1: A simple example of SDC protocol (*TTL* = 2).

5.1 Experiment Settings

To test the applicability of our clustering protocol to different network structures, we use two types of topologies: Waxman topologies from GT-ITM topology generator [7] and power-law topologies from the BRITE generator [24].

We implement both the SDC and CDC algorithms and evaluate their performance on different types of topologies. There are several configurable parameters for the CDC scheme: *Vicinity, TwoHopThreshold, WeightThreshold* and *MinWeight*. These parameters affect the performance of CDC significantly: increasing the values of these parameters reduces the number of discovered clusters and causes more orphan nodes. We tune these parameters carefully towards the best clustering accuracy of CDC. Specifically, we set the values of *Vicinity, TwoHopThreshold, WeightThreshold, MinWeight* as: 1, 0.1, 0.0001, 0.0001 respectively. Besides these parameters, TTL is also critical to CDC as it affects the clustering accuracy by controlling he granularity of discovered clusters. Based on [27] and our observations, higher TTL values correspond to more accurate clusters with the tradeoff of increased number of messages. A TTL of 2 is used as the accuracy of CDC is not affected significantly by higher TTL values while the message overhead is much smaller than the overhead caused by higher TTL values.

We simulate the fully distributed systems in which a node only knows its directly connected neighbors. The one way 1-hop delay of any exchanged message is set to 1 simulated time.

The performance of SDC is evaluated in two network scenarios:

1. Static system where network nodes form a fixed topology throughout the whole simulation. In the beginning, every node shows up as an orphan node and starts its own clustering procedure independently at a random time $t \in [0, T]$. The simulation ends until every node is clustered and no message is exchanged.
2. Dynamic system in which the topology is changed by adding *X* new nodes to or removing *X* existing nodes from the system.

The main metrics used to evaluate the performance of the two algorithms are: *SCM*, *Message Overhead* and *Convergence Time*. SCM is used to evaluate the accuracy of the algorithm. Message Overhead is defined as the number of exchanged messages among nodes. Convergence time is the amount of simulated time from the first clustering operation until the end of the simulation. We also study the influence of node degree and TTL on the performance of SDC.

5.2 Performance of SDC in Static Systems

We first simulate the performance of SDC in clustering static systems, i.e., no node enters or exits the network. We are interested in the performance of SDC on different topology structures and topology sizes.

5.2.1 Affect of Clustering Time Span T

In SDC, a node may receive multiple clustering requests simultaneously especially when all the nodes start clustering in a short period of time, i.e. *T* is small. When receiving multiple requests, a node randomly picks one to serve and rejects the others. The smaller the value of *T*, the more rejected requests and control messages. Therefore, the value of *T* is a factor that can influence the performance of SDC significantly. We conduct simulations for different value of *T*, starting from 1 to 100 using a power law topology with 1000 nodes. The clustering accuracy, message overhead and convergence time are shown in the Fig. 2 □ 4.

As shown in Fig.2, increasing the value of *T* can improve SDC's clustering accuracy. The reason is straightforward. In the scenario of large *T*, a node can get service from more neighboring clusters due to the less competitions and thus is more likely to choose the best cluster to join. With the increase of *T*, the SCM keeps raising towards the maximum SCM value of the topology with a slower rate.

Fig.3 shows the message overhead under different value of *T*. When *T* is small (less than 10 for our simulation), the message overhead shows an increasing trend with the value of *T*. The reason can be explained as follows: When *T* is small, there are a lot of simultaneous clustering requests in the same area of the topology. Based on the SDC algorithm, only 1 request can be served at a time and the others have to be rejected and served later. Therefore, when *T* is small, a lot of clustering requests must be rejected which results in high control message overhead and many requests must be re-sent which also increases the message overhead. With the increase of *T*, the number of simultaneous requests in the same area is reduced, which contributes to the decreased message overhead. We can also observe a point in the value of *T* after which the message overhead shows an increasing trend. When *T* is large, although the overhead caused by simultaneous requests in the same area is reduced, the clustering operations are less aggregated. For example, a node n_i may postpone its moving attempt to a cluster C_j due to its neighbor's current clustering operation. If its neighbor also joins the cluster C_j, node n_i should perform another moving attempt that can be aggregated with the previous postponed moving operation. With the increase of *T*, clustering operations are less aggregated which increases the exchanged messages. When *T* is large enough, the increased number of messages due to the

less clustering aggregation becomes more significant than the reduced number of messages due to the less simultaneous clustering requests and the overall message overhead shows an increasing trend.

The last metric we are interested in is the convergence time under different value of *T*. With the increase of *T*, the convergence time grows slowly. When *T* is larger than 30, the convergence time shows a rough linear correlation with the value of *T*. We can observe that the clustering procedure is more time consuming when *T* is small. The reason is small *T* causes simultaneous clustering attempt at more nodes, which further results in more time for handling the simultaneous requests and re-clustering.

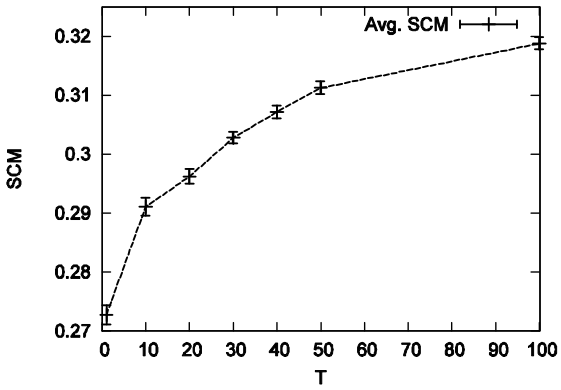


Figure 2: Clustering accuracy on power laws topology of 1000 nodes

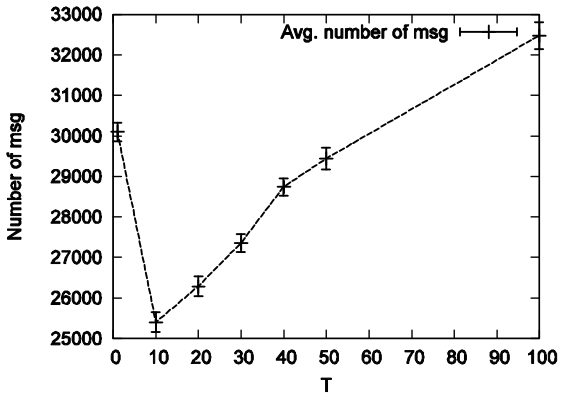


Figure 3: Message overhead on power law topology of 1000 nodes

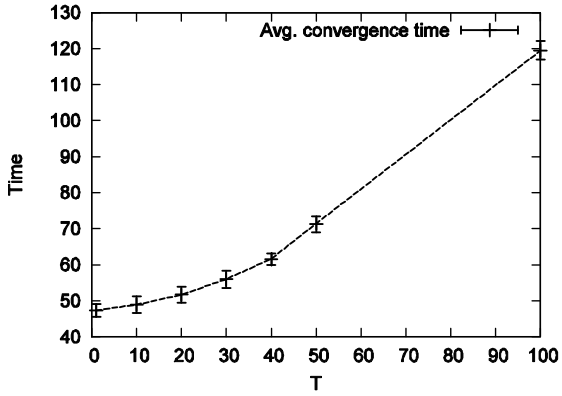


Figure 4: Convergence time on power law topology of 1000 nodes

5.2.2 Performance in Different Topology Size

In this set of simulations, we want to evaluate the performance of SDC in handling different size of topologies. Two types of topologies are used: random topologies from the Waxman topology model and power law topologies from the BA model. Both sets of topologies scale from 1000 nodes to 6000 nodes. We use the existing distributed clustering algorithm CDC as the reference point of our evaluation.

We first evaluate the performance of SDC on Waxman topologies. In this type of topologies, nodes are connected in a random way: only the distance information is considered. The average node degree is controlled to around 4 for all topologies.

The performance of both algorithms are shown in the Fig.5 □ 7. Since the topology structure is unchanged with the topology size, both algorithms have very stable clustering accuracy as shown in Fig.5. When compare the accuracy of the two algorithms, it is obvious that SDC performs a lot better than CDC. Recall that the range *T* for SDC is set to 2, so we expect even better performance of SDC in the scenario of larger *T*.

Fig.6 shows the message overhead of both algorithms. With the increase of topology size, the message overhead of SDC and CDC has a linear growing trend. Compared with CDC, SDC generates much lower message overhead which increases with topology size slowly. On the other hand, the message overhead of CDC is very high and increases rapidly with topology size as it is a flooding-based method.

The only metric in which CDC outperforms SDC is the convergence time. For CDC, clusters are formed right after message flooding. Therefore, the convergence time of CDC is determined only by the TTL of flooding and is unchanged with topology size. For SDC, since a node can serve only 1 clustering request at a time, the convergence time increases with the number of nodes. As multiple clustering requests at different areas of the topology can be served simultaneously, SDC's convergence time has a sublinear correlation with topology size.

Fig.8 □ 10 shows the performance of both algorithms on power law topologies. This type of topologies have very skewed degree distributions. For a BA topology with 5000 nodes, the highest degree is more than 100 while the average degree is only 4. The clustering accuracy of SDC on the power law topologies is slightly lower than on the Waxman topologies. The reason is the power law topologies have less significant clustering features that result in lower SCM values. Compared to CDC, the performance of SDC is consistently better in terms of clustering accuracy and message overhead. The convergence time is reduced on power law topologies but still higher than CDC.

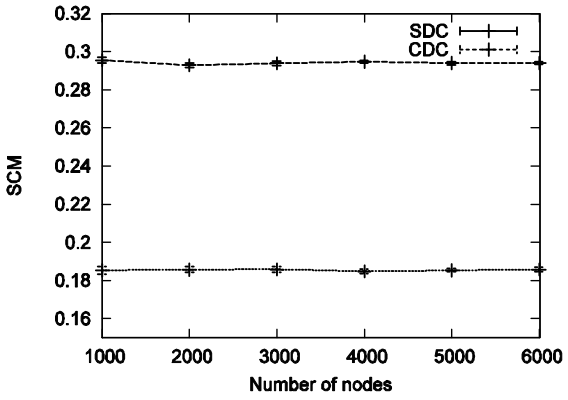


Figure 5: Clustering accuracy on Waxman topologies

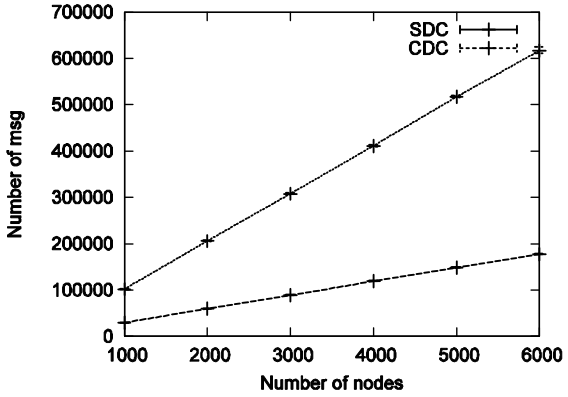


Figure 6: Message overhead on Waxman topologies

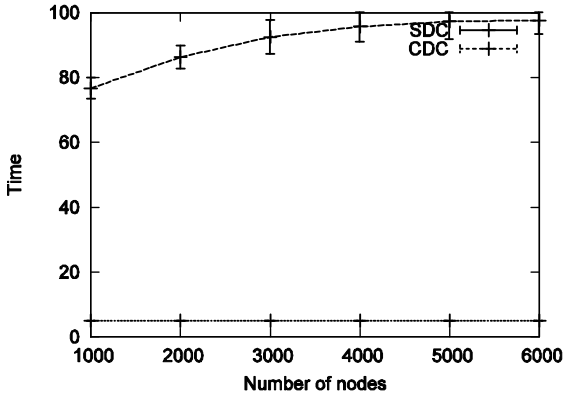


Figure 7: Convergence time on Waxman topologies

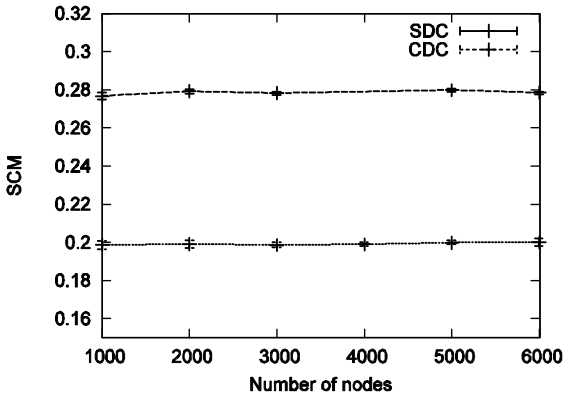


Figure 8: Clustering accuracy on BA topologies

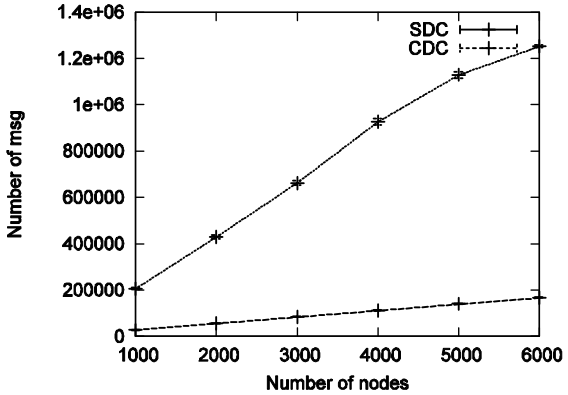


Figure 9: Message overhead on BA topologies

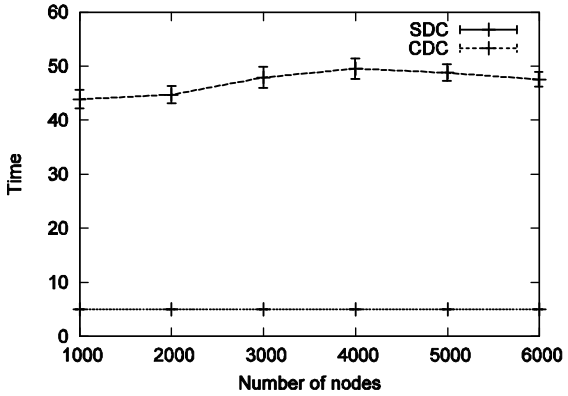


Figure 10: Convergence time on BA topologies

5.3 Performance of SDC in Dynamic Systems

This set of simulations are conducted to evaluate the performance of SDC in handling node dynamics. The topologies used in this section are power law and Waxman topologies with 1000 nodes. To simulate the Join event, we first cluster the initial topology using the SDC algorithm and then we add *X* new nodes simultaneously to the network. Each of the new nodes connects with the existing nodes independently based on the topology model so that the topology structure can be maintained. For the Leave event, we randomly remove *X* existing nodes simultaneously from the topology. The simulations end when there is no message exchanged. We change the value of *X* from 1 to 50 and measure the clustering accuracy after node dynamics, message overhead and convergence time since the first Join/Leave event takes place.

Fig.11 □ 16 show the performance of SDC on handling node dynamics in a power law topology. The clustering accuracy after the simultaneous Join/Leave events is slightly changed, which

indicates SDC is able to maintain accurate clustering from simultaneous node dynamics. The logic is that the topology structure is not changed significantly and therefore, accurate clustering must result in an SCM value that is close to the initial SCM before node dynamics. We also observe the SCM value after the Leave events increases slightly with the number of removed nodes. This performance is reasonable because removing a few existing edges makes the clustering features of the topology clearer. Since SDC can accurately form clusters, an increased SCM value that is consistent with the more significant clustering features can be observed. Following the same logic, when adding new nodes to the topology, the clustering features become less significant, which results in the slightly reduced SCM value.

Fig.12 shows the message overhead for node dynamics in the power law topology. For both Join and Leave events, the message overhead shows a linear correlation with the number of dynamic nodes. Moreover, the Leave events cause more exchanged messages than the Join events. This is because more nodes are affected and need to re-cluster after a Leave event than after a Join event. After a node leaves, all of its previous neighbors and all the nodes in its original cluster should re-cluster but after a new node joins the topology, only its neighbors are affected and re-cluster.

Fig.13 shows the convergence time of SDC for node dynamics. With the increase of Join/Leave events, the convergence time increases slowly. We can also observe that Leave events take longer to converge than Join events because more nodes are affected by a Leave event and need to re-cluster.

The performance on the Waxman topology is consistent with the performance on the BA topology. We can see the advantage of SDC in handling dynamic systems: With low message overhead, accurate clusters can be maintained after a different number of simultaneous Join and Leave events. This performance is especially suitable for a system with continuous node entry and exit. To maintain an acceptable clustering accuracy, the existing algorithm CDC must re-cluster the whole network after a certain number of node entry and exit, which causes a high message overhead and is not scalable for large networks.

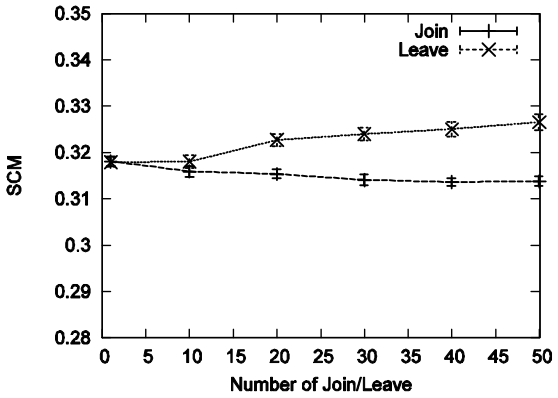


Figure 11: Clustering accuracy for node dynamics on BA topology with 1000 nodes

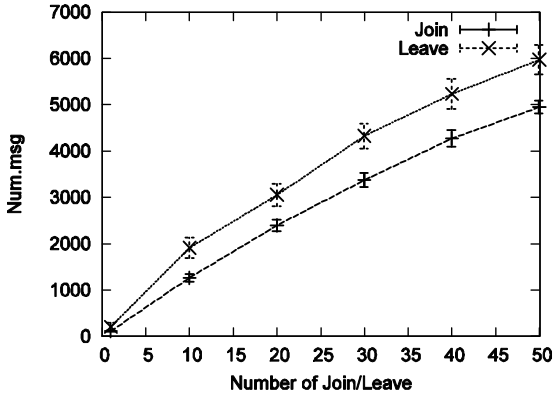


Figure 12: Message overhead for node dynamics on BA topology with 1000 nodes

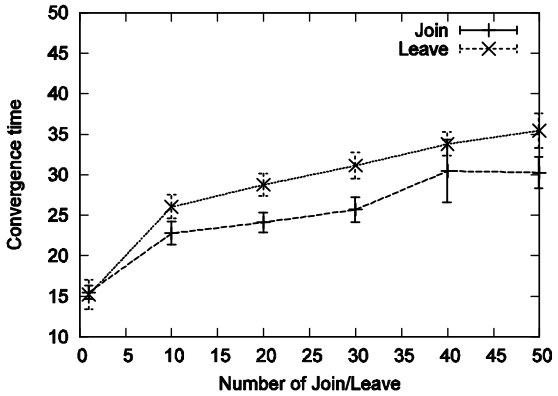


Figure 13: Convergence time for node dynamics on BA topology with 1000 nodes

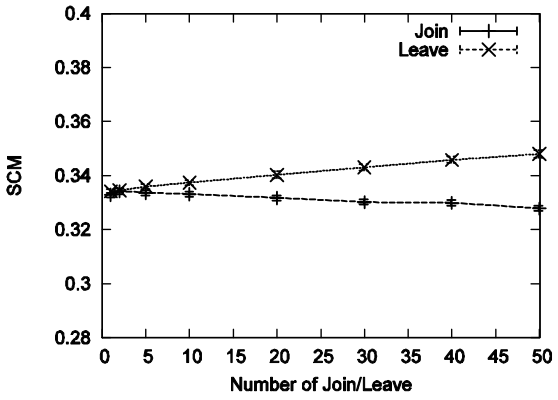


Figure 14: Clustering accuracy for node dynamics on Waxman topology with 1000 nodes

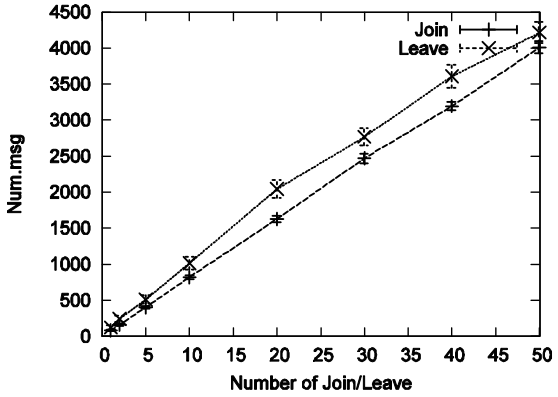


Figure 15: Message overhead for node dynamics on Waxman topology with 1000 nodes

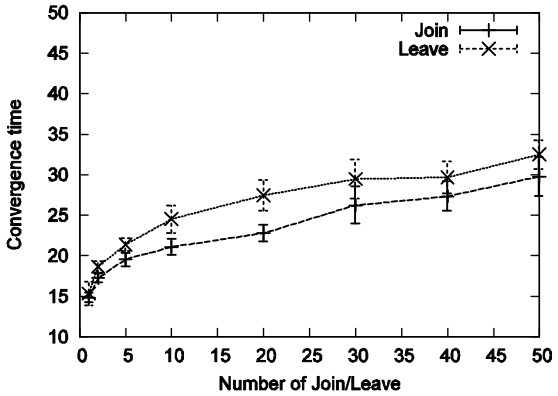


Figure 16: Convergence time for node dynamics on Waxman topology with 1000 nodes

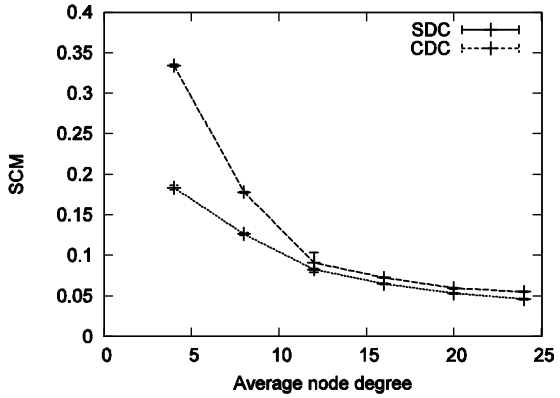


Figure 17: Effect of node degree on clustering accuracy

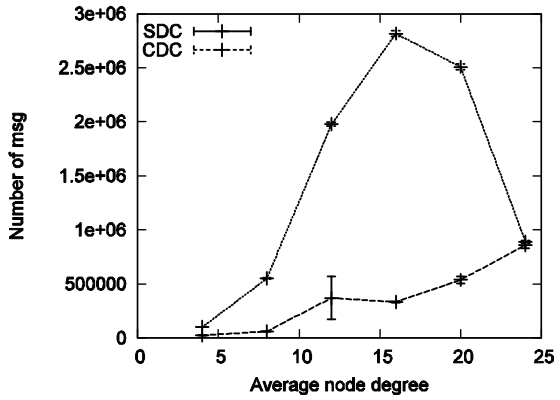


Figure 18: Effect of node degree on message overhead

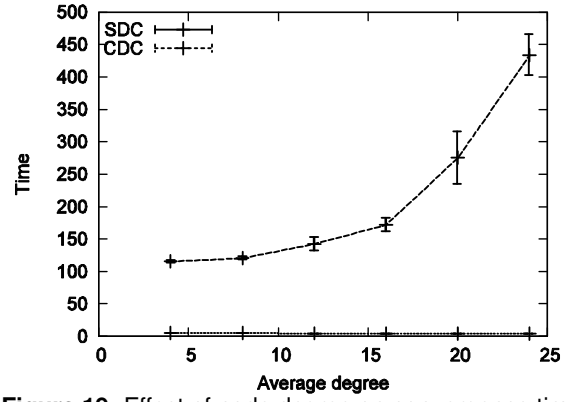


Figure 19: Effect of node degree on convergence time

5.4 Influence of Node Degree

Node degree is an important factor to the performance of SDC since the clustering procedure is based on node connectivity. In this set of simulations, we study how different average node degree can affect the performance of the algorithm.

The topologies used in this section are Waxman topologies with 1000 nodes and different average degree ranging from 4 to 24. The range T is set to 100. Again, we use CDC as the reference point of the evaluation.

Fig. 17 shows the clustering accuracy of the two algorithms against different average node degrees. Clearly, the SCM values of both SDC and CDC drop with the increase of average degree. This decline of SCM is mainly caused by the decrease in clustering features of the topology other than the clustering algorithms. Since a connection is determined in a random manner by the Waxman model, increasing the average degree adds more randomness to the topology as translated to less clustering features. However, SDC can capture the clustering features better than CDC as shown by the higher SCM values, especially when the clustering features are more significant.

Fig.18 shows the message overhead of both algorithms when clustering topologies with different average degrees. It is shown that the increase in average degree leads to higher message overhead for SDC. This fact is under our expectation. In SDC, after a node finishes its current clustering operation, its neighbors need to start a new round of clustering procedures. If a topology has higher average degree, more nodes are involved in each clustering procedure and need to take actions after the current clustering, which leads to more message overhead. When compare the two algorithms, we claim a better performance of SDC as the generated messages in SDC are much less than in CDC. A quick drop in the message overhead of CDC can be observed when the average degree exceeds 16, which is caused by the parameter setting. The convergence time of SDC also inclines with the increase of average node degree due to the same reason.

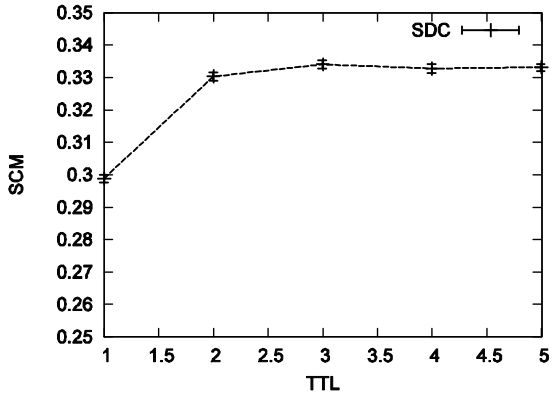
As a summary, SDC is able to detect accurate clusters in both sparse and dense topologies. The message overhead and convergence time do increase with average node degree due to increased number of nodes involved by each clustering operation.

5.5 Influence of TTL

In SDC, a node's clustering request is rejected if the TTL of the request message expires. Therefore, clustering results are affected by the value of TTL. Intuitively, large TTL values do not influence clustering accuracy because nodes can always join a cluster that leads to the highest SCM value without being rejected due to the expire of TTL. The questions studied in this section are how large the TTL should be to guarantee accurate clustering and how the other performance metrics are affected by different TTL values. We cluster a 1000 node Waxman topology using SDC with TTL varied from 1 to 5 and measure the performance metrics.

Fig.20 shows the clustering accuracy of SDC against different TTL. It is under our expectation that the SCM is improved with the increase of TTL. The most significant SCM improvement happens when TTL changes from 1 to 2. Obviously, for the Waxman topology in our simulation, clusters with 1-hop diameter are not accurate at all. When we further increase TTL, the SCM value does not benefit from higher TTLs. Based on the definition of SCM, accurate clusters do not have large diameters because the number of non-neighbor nodes should be minimized. Thus, clusters do not grow further when their diameter reaches 2, which leads to the stable SCM values at TTL of 2 and higher.

When we examine the message overhead, we observe positive effect of increasing TTL as shown in Fig.21. The steady decrease of message overhead is a result of reduced $Clust\_Reject$ messages due to higher TTL values. Similar to SCM, the most significant reduction happens when TTL increases from 1 to 2 and the message overhead stabilizes when TTL is further increased. These results can be explained as follows: When TTL is 1, many clustering requests are rejected due to TTL expiration, which results in high message overhead. When we increase TTL from 1 to 2, the $Clust\_Reject$ messages are reduced significantly and the clusters are able to grow towards the highest accuracy, which results in the significant drop in message overhead. Further increase in TTL has little effect on message overhead because most clusters stop growing when diameter reaches 2 and TTL no longer expires. The convergence time under different TTL values is shown in Fig.22 that can be explained in the same way as we do for message overhead.



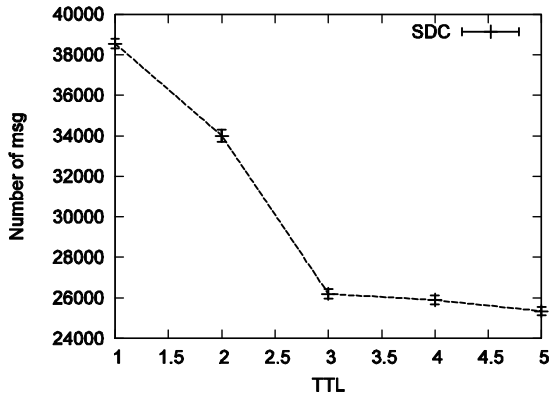


Figure 20: Effect of TTL on clustering accuracy

Figure 21: Effect of TTL on message overhead

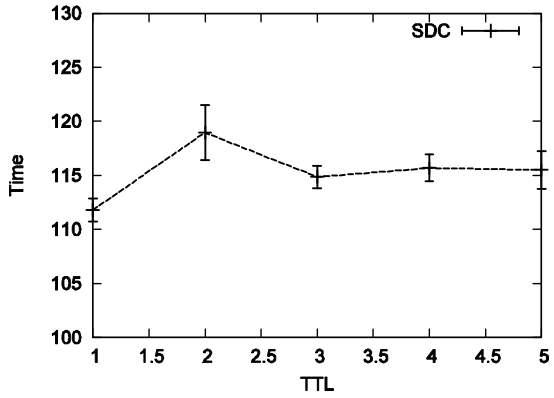


Figure 22: Effect of TTL on convergence time

6. CONCLUSIONS

In this paper, we target the challenging problem of clustering large-scale distributed systems such as P2P networks. We identify the main issues in the existing clustering algorithms: First, many existing algorithms are centralized methods and therefore they are not scalable and efficient in handling large-scale distributed systems. Second, although several distributed clustering algorithms have been proposed, they can not guarantee accurate clustering and low message overhead especially when handling dynamic systems. Therefore, we propose a novel distributed clustering protocol SDC for large-scale distributed systems. We conduct extensive simulations to evaluate the performance of SDC under various scenarios. The simulation results show the promising performance of SDC: it is a scalable and accurate clustering approach with small

message overhead on various topologies. The most attractive feature of SDC is that it can reserve high clustering accuracy from multiple simultaneous node entry and exit with small message overhead.

7. REFERENCES

[1] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30:2826–2841, June 2007.

[2] J. Ahuja and J.-H. Cui. A scalable peer-to-peer file sharing system supporting complex queries. *UCONN CSE Technical Report*, *UbiNet TR05-01*, January 2005.

[3] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, December 2004.

[4] R. Albert, H. Jeong, and A.-L. Barabasi. Error and attack tolerance of complex networks. *Discrete Applied Mathematics*, 406:378–382, August 2000.
[5] A. Barbu and S.-C. Zhu. Graph partition by swendsen-wang cuts. *Ninth IEEE International Conference on Computer Vision Volume 1*, 10 13 - 10 2003, Nice, France.

[6] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85(25):5468–5471, Dec 2000.

[7] K. Calvert and E. Zegura. Gt-itm: Georgia tech internetwork topology models. *http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz*, 1996.

[8] M. Chatterjee, S. K. Das, and D. Turgut. Wca: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5:193–204, April 2002.

[9] A. Crespo and H. Garcia-Molina. Semantic overlay networks for p2p systems, 2002.

[10] J. S. Deogun, D. Kratsch, and G. Steiner. An approximation algorithm for clustering graphs with dominating diametral path. *Inf. Process. Lett.*, 61(3):121–127, 1997.

[11] N. Dimokas, D. Katsaros, and Y. Manolopoulos. Node clustering in wireless sensor networks by considering structural characteristics of the network graph. *International Conference on Information Technology*, 00:122–127, 2007.

[12] D. Doleva, S. Jaminb, O. Mokrync, and Y. Shavittc. Internet resiliency to attacks and failures under bgp policy routing. *Computer Networks*, 50:3183–3196, November 2005.

[13] D. Estrin, Y. Rekhter, and S. Hotz. Scalable inter-domain routing architecture. *In SIGCOMM '92: Conference proceedings on Communications architectures & protocols*, pages 40–52, 1992.

[14] Y. Fernandess and D. Malkhi. K-clustering in wireless ad hoc networks. *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 31–37, 2002.

[15] A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology on the spread of epidemics. *In IEEE INFOCOM*, volume 2, pages 1455–1466, March 2005.

[16] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G. Urvoy-Keller. Hierarchical p2p systems. *In Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*, 2003.

[17] C. Gkantsidis, M. Mihail, , and E. Zegura. Spectral analysis of internet topologies. *IEEE INFOCOM,* 2003.

[18] R. Hoes, T. Basten, W.-L. Yeow, C.-K. Tham, M. Geilen, and H. Corporaal. Qos management for wireless sensor networks with a mobile sink. *In EWSN '09: Proceedings of the 6th European Conference on Wireless Sensor Networks*, pages 53–68, Berlin, Heidelberg, 2009. Springer-Verlag.

[19] J. Jin, J. Liang, J. Jin, and K. Nahrstedt. Large-scale qos-aware service-oriented networking with a clustering-based approach. *In 16th International Conference on Computer Communications and Networks*, pages 522–528, August 2007.

[20] V. Kantere, D. Tsoumakos, T. Sellis, and N. Roussopoulos. Groupeer: Dynamic clustering of p2p databases. *Information Systems*, 34:62–86, March 2009.

[21] G. Kwon and K. D. Ryu. An efficient peer-to-peer file sharing exploiting hierarchy and asymmetry. *In SAINT*, pages 226–233, 2003.

[22] Y. Li, J.-H. Cui, D. Maggiorini, and M. Faloutsos. Characterizing and modeling clustering features in as-level internet topology. *In Proceedings of IEEE INFOCOM*, 2008.

[23] A. McDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communication*, Vol. 17, No. 8, August 1999.

[24] A. Medina, A. Lakhina, I. Matta, , and J. Byers. Brite: Universal topology generation from a user's perspective. *In Proceedings of Workshop the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS '01)*, October 2001.

[25] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. *In SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61, New York, NY, USA, 2004. ACM.

[26] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86(14):3200–3203, Apr 2001.

[27] L. Ramaswamy, B. Gedik, and L. Liu. A distributed approach to node clustering in decentralized peerto-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 16(9), Sept. 2005.

[28] S. van Dongen. A new cluster algorithm for graphs. *Technical report INS-R9814, Centrum voor Wiskunde en Informatica (CWI)*, ISSN 1386-3681, Dec. 1998.

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

[30] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. *In IEEE INFOCOM*, 2004.

[31] W. Zheng, S. Zhang, Y. Ouyang, F. Makedon, and J. Ford. Node clustering based on link delay in p2p networks. *In SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 744–749, 2005.

[32] C. C. Zou, D. Towsley, and W. Gong. Modeling and simulation study of the propagation and defense of internet email worm. *IEEE Transactions on Dependable and Secure Computing*, 4:105–118, 2007.