Distributed Clustering from Peer-to-Peer Networks

Georgios Chinis
Computer Science Department
University of Crete
chinis@csd.uoc.gr

May 30, 2011

1 Introduction

Decentralized peer-to-peer (P2P) clustering has wide-ranging applications such as in P2P file sharing systems, mobile ad-hoc networks, P2P sensor networks and so forth. However, it is very challenging to design a clustering protocol for such networks since by design there exist no central point of administration or even an entity with complete knowledge of the network. Moreover, nodes in such systems can communicate only with their neighbors.

Clustering algorithms are divided into two categories based on their clustering criterion. First is the connectivity based clustering. This clustering is based on criteria from the topology of the node. For instance its degree or the probability returning back after a random walk in the graph. Second, there is the content-based algorithm. In this approach clustering is based on some similarity between the nodes that is orthogonal with the graph with its position on the graph, usually nodes are clustered based on a property they acquire from external sources.

2 CDC

The connectivity-based distributed node clustering (CDC) [3] algorithm clusters nodes according to their position in the network. In contrast to central-

ized algorithms CDC only requires local knowledge about neighboring nodes. An important property of this algorithm it the ability to cluster the entire network or discover clusters around a given set of nodes. Moreover the algorithm is capable of handling dynamic entrance and exit of nodes without resorting to re-clustering.

2.1 The Idea

The CDC algorithm tries to simulate the network flow in the network in a distributed and scalable fashion. The rationale behind clustering based on network flow is based on the following intuition. Imagine the network as a set of intersecting roads. Roads represent the edges of the graph and the intersections the vertexes. Suppose that from a node in the graph, called the *originator node*, starts a large number of people each holding some weight. The crowd does not know the topology of graph so it chooses roads, *vertexes* at random. Each time a person reaches a node it leaves some of it weight, chooses a road at random and continues wandering in the graph. From this algorithm we can observe the following results.

- Nodes around the originator node have accumulated more weight than those far from it.
- Nodes that are densely connected with the originator, having more paths with it, have accumulated more weight.

The CDC algorithm is based on the previous observations. If there exist a few originators in the graph from where people could start random walks, the nodes would acquire different weight for each originator. Then, the nodes could join the cluster from whose originator received the maximum weight.

2.2 The Algorithm

The algorithm works by sending messages from the *originator* to the rest of the nodes in the graph. The election of the originators will be discussed in the nect session for now let's assume that there exist a set of $O = \{O_1, O_2, \cdots, O_Q\}$. These originator initiate the clustering by sending messages to their neighbors. Each message contains the ID of the originator that initiated (OID), a Time-to-Live (TTL) and the weight (MWeight). Each originator O_i initiates the $MWeight = \frac{1}{Degree(O_i)}$. TTL is a small integer used

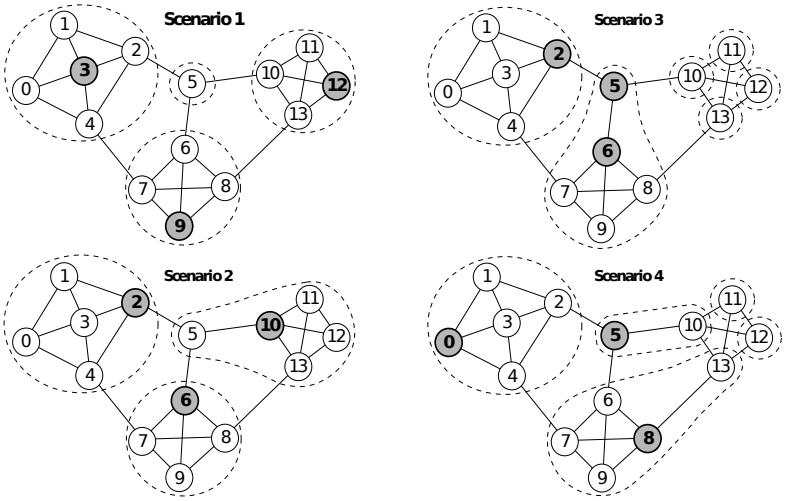


Figure 1: Different clusterings based on the selection of the originator nodes (dark color).

for preventing messages from circulating in the network for ever. Each node that receives a messages decreases the TTL by one before forwarding it to its neighbors. If a node ever receive a message with TTL zero or less than zero it discards it. Finally, the OID is used by the nodes to summarize the weight they receive from each originator.

Upon receiving a message a node V_i it updates the total weight received from the corresponding originator. Then it checks the TTL of the message, if it is greater then zero, the weight is divided by the degree of the node, the TTL is decreased by 1 and the new message if propagate to all its neighbors. Each node can receive multiple messages from each originator, after a node has received the last message, the node joins the cluster from whose originator it received the maximum weight. If the node has not accumulated enough weight from any originator then it can choose to remain an outlier.

For a node to join a cluster it need to send message to the originator of the cluster containing its ID. In the future if the node accumulate more weight from a different originator it can withdraw from its current cluster and join the new one, informing the two originators about the change.

2.3 THP Originator Determination Scheme

The algorithm is capable of automatic partitioning the whole graph or partitioning around some designated nodes. In the latter case the choice of the originators is straightforward in the former case the choice of the originators greatly affects the quality of the clustering. As shown in Figure 1 the originator in scerarios 1,2 results to fairly good clustering, on the other hand scenarios 3,4 the clustering is far from optimal. In scenario 3, the originators are very close to each other which causes areas off the graph to be remote from any originator. In scerarion 4, the originators in spite of being scattered fail to provide a good clustering. This leads to two properties that originators must hold.

Property 1 The set of originators should be spread out in all the regions of the graph.

Property 2 A node V_i is considered to be a good originator if it acquires more weight due to messaged initiated by it than the weight acquired by other originators.

Based on the previous two properties the authors draft an algorithm for originator selection. Each node wakes up spontaneously and perform two operations. First, it checks if there are other originator in the *vicinity*. The vicinity configurable parameter. The node checks from the messages it has received if $MSG.TTL > InitialTTL - Vicinity$ if this inequation holds than the node is unsuitable for originator. In order to distributively check if the second property holds the node computes the *two hop probability*. $TwoHopProb(V_l) = \sum_{V_i \in Nbr(V_l)} (\frac{1}{Degree(V_l) \times Degree(V_i)})$ The two hop probability is the probability of returning the originator after performing a random walk from the originator of length 2.

2.4 Handling Dynamics of Nodes

Since this algorithm is designed to work on P2P networks it is crucial to handle nodes entering and leaving the system dynamically without the need of re-clustering the network from the beginning. The first problem it handling node entry. A node entering the system knows only its immediate neighbors. Based on that knowledge the node computes how strong it is attracted to neighbor clusters. $ClustAttraction(CL_i) = \sum_{V_i \in Cl_i \wedge V_i \in Nbr(V_N)} \frac{1}{Degree(V_i)}$ The

node can choose to join the cluster with maximum ClustAttraction or remain an outlier if it is not attracted enough from any neighboring cluster.

In case of a node leaving the system than the neighbors need to evaluate their clustering since the leaving node may was the one that connected them to their cluster. Hence, when a node exits the system the neighbors need to compute the ClusterAttaction, as in node entry, to decide if they need to change their clustering.

3 SDC

The SCM-based Distributed Protocol (SDC) [2] is another approach for distributed connectivity-based clustering. The main criterion for this clustering is that in a good cluster the intra-node connectivity should be maximized and the inter-cluster connectivity should be minimized. Also, the size of the clusters should remain inside some boundaries since expanded cluster are difficult to administered in decentralized systems and provide little to the overall stability of the system.

3.1 The Idea

The algorithm is based on the Scaled Measured Moverage (SCM) metric proposed in [1]. The SCM is defined as:

$$SCM(v_i) = 1 - \frac{|FalsPos(v_i, C)| + |FalseNeg(v_i, C)|}{|Nbr(v_i) \cup Clust(v_i)|}$$

Nbr(v_i) is the set of neighbors of node v_i.

Clust(v_i) is the set of node in the same cluster as node 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.

Based on for the SCM for one node we also define the SCM for a graph as $SCM(G) = \frac{\sum_{v_i} SCM(v_i)}{n}$. It is easy to see that the higher the SCM the

smaller the connectivity between clusters and the higher the connectivity inside clusters. For graphs containing only isolated clusters that are themselves fully connected, the SCM value is 1. Based on the previous observations the problem of network clustering can be simplified as partitioning a network topology so that SCM is maximized.

3.2 The Algorithm

In the beginning, each node consists of a cluster by itself. Then nodes start creating clusters in a greedy manner. Each node sends a *cluster request* to its neighbors notifying them for its willingness to perform clustering. The neighbors receive the request and respond if they are willing to participate in the clustering. A node can deny taking part to some nodes clustering if it is already involved in some else's node clustering. If the neighbors confirm that they will co-operate then the second phase of the clustering begins. In this phase the originating node sends a 'cluster confirm' message to its neighbors, which in turn propagate this message to the other nodes in the same cluster as them until all nodes in the cluster have received it. When a node receives a 'cluster confirm' message it sends back the originator a message containing the ΔSCM. ΔSCM is computed as follows, if the node belongs in the same cluster as the originator then ΔSCM is the difference of the current value of the SCM for this node and the value of the SCM is the originator leaves the cluster. If the node is not in the same cluster as the originator is the difference between the current SCM and the SCM if the originator joins the cluster. The originator receives the responses from all the nodes in the same cluster and all nodes in the neighboring clusters and compute the $\Delta SCM(G) = \Delta_{join} + \Delta_{leave}$. Should this value is positive the node will abandon its current cluster and join the neighboring cluster. After that, the node notifies its neighbors so that they recompute their position and maybe change cluster. If there are more than one neighboring clusters then the node compute $\Delta SCM(G) = \Delta_{join} + \Delta_{leave}$ for each one of them and joins the one with the maximum positive value.

The algorithm can vulnerable to deadlock because for clustering operation of one node there have to co-operate two clusters. If two nodes start clustering simultaneously and they share a common neighboring cluster then it is possibly that some of the nodes in the cluster will participate in the one node's clustering and the other nodes in the cluster to the other node's clustering. In that case no originating will have the co-operation of the full

cluster and hence will wait forever until the rest of the nodes are available. The authors propose the use of timeout where the originating node wait a specific amount of time and then cancels the clustering and second the use of random timing so that no two nodes decide to cluster at the same time.

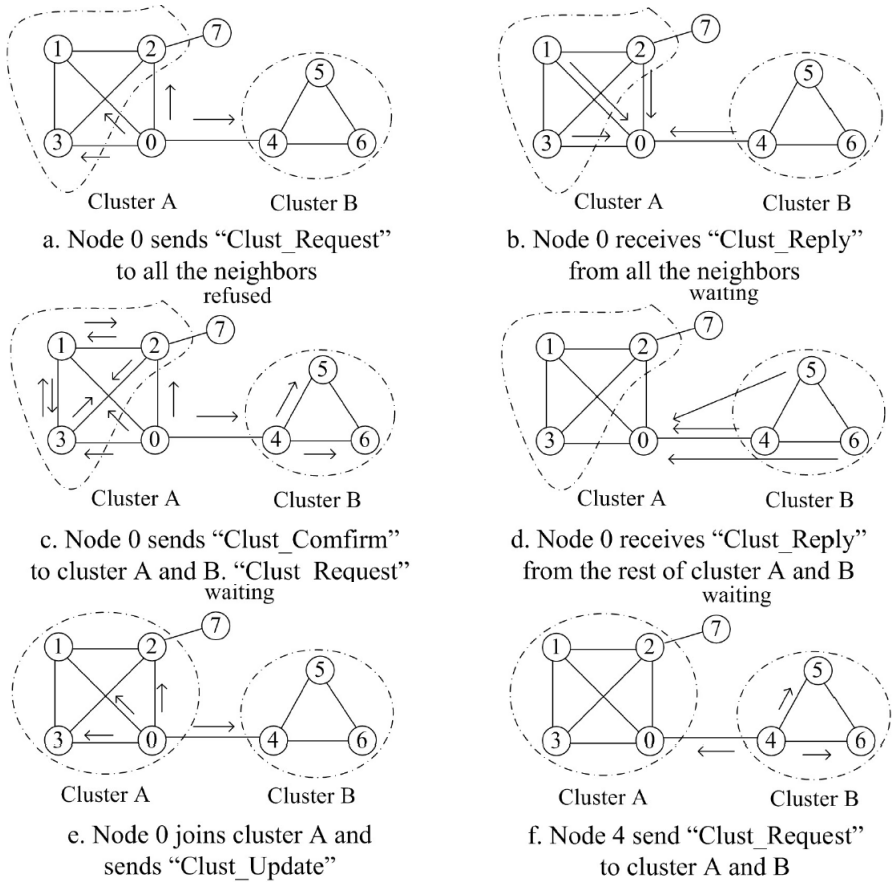


Figure 2: Example of the SDC protocol

4 CAGA

The Clustering Algorithm with Granularity Awareness (CAGA) algorithm is another connectivity-based clustering algorithm [5]. It has been proposed as solution for P2P isolation in the face of threats like worm contamination.

The algorithm is designed to allow peers to maintain some level of service even while they are isolated in their cluster.

4.1 The Idea

In order for the nodes to continue receive the service provided by the network even while isolation is necessary that each node belongs to the same cluster as its neighbors. Moreover, tightly connected (high degree) peers should not be separated from its neighbors. Based on the previous assumptions the authors propose two metrics to help them cluster the network.

- In a cluster with m members, for each member who has c_i outward links.
$$\delta = \frac{(\sum_{i=1}^{m} c_i)}{m}$$
- In a cluster with m members and n internal links for each member who has c_i outward links.
$$\epsilon = \frac{(\sum_{i=1}^{m} c_i)}{n}$$

The smaller the δ and ϵ are, the better the clustering becomes.

4.2 The Algorithm

The algorithm works using a set of initial peers just as CDC in Section 2. As initial peers can be chosen the most stable peers of the system, for instance the ones with the bigger uptime. Each of these peers represent a cluster and tries to expand. In order to expand the cluster sends messages to all neighboring peers that are not member of the cluster and ask them to provide some information about themselves. Based on that information the originator of the cluster computes the metrics δ and ϵ if one of these metrics is improved by entering the node in the cluster then the originator sends an invitation to the node to join the cluster. If the node does not already belong to a cluster in accepts the invitation, it then sends to the originator a least of all its neighbors so that the originator can continue expanding the cluster by inviting those neighbors.

4.3 Complexity

CAGA must check each member of the cluster once and generate a *neighbor-list* and an invitation message as well as their responses, so the time is $O(1)$ and the traffic is $O(4)$. A few N' peers may lead to redundant checking by different clusters. So the total running time is $O(N) + O(cN')$ while the traffic is $O(4N) + O(2cN')$. As $N' << N$, the running time is $O(N)$ and the traffic is $O(4N) = O(N)$.

5 Schelling

The Schelling's algorithms [4] in unique in two ways. First, it belongs to the category of content-based clustering in contrast to connectivity based-cluster that we previously saw. This means that the clustering is based on some property that the nodes exhibit, we try to cluster node having the same property. Second, this algorithm does not try to discover existing clusters, it tries to create them by re-arranging the graph.

5.1 The Idea

The model was proposed by the sociologist Tomas Schelling to explain the existence of segregated neighbors in urban areas. The world is modeled as a 2-D grid. Some cells are populated by blue or red turtles and the rest are empty. All turtles desire at least a number of neighbors to have the same color. If a turtle is not satisfied it can move to an adjacent empty cell. Using game theory has been proven that the stable state for the system is a segregated state. In the Schelling's model turtles have only local view of the grid which is a great analogy for the peers in a P2P system. The neighbors can model

5.2 The Algorithm

Each peer desires to be connected with at least a percentage of peers with similar property $PNSP_{desired}$. If the peer is not satisfied in its current position it executes its topology adaptation steps. The adaptation steps consists of dropping a link with a neighbor of a different property and then trying to connect with a neighbor of the same property. Drop a connection increases the $PNSP_{desired}$ both in this node and also in its neighbor. Searching for

a different neighbors can be implemented with many algorithms. An exhaustive approach with high probability of success could be using the BFS algorithm, but this could impose a high traffic overhead. Another approach for reduced traffic overhead it the use of random walks. Random walks are cheaper but will not a peer's neighborhood as thoroughly and is less likely to find a similar peer. Another alternative is biased random walk that perform a more exhaustive search when compared to random walk. In this technique the random walk is biased towards peers with a high degree because they have more information about resources on the overlay network.

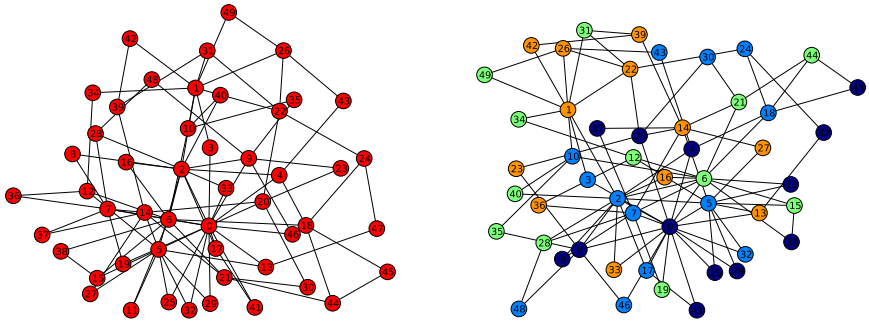
A satisfied peer need not estimate its satisfaction state its satisfaction state as small intervals. The same holds for peers that are unable to successfully execute their topology adaptation steps.

6 Comparison

For this project i have implemented three of the previous algorithms CDC, CAGA and Schelling's. Since the algorithms are supposed to be distributed in my implementation each vertex is a different entity and entities can communicate only with other entities that are neighbors. For each algorithm i initiated the system and waited until it balanced by itself, then out of the new system i extracted the graph. For my experiments i used graphs from 50-150 vertexes with power law distribution, since this distribution best describes the P2P system that exist in the wild. Unfortunately, due to limited processing resources my experiments did not pass the 150 nodes.

6.1 Graphical Comparison

In Figure 3 we one random graph of 50 nodes following the power low distribution in the degrees. The normal graph will be used as input in the CDC and CAGA algorithm whereas the colored one in the Schelling's algorithm. The second graphed was colored randomly. In Figure 4 we can see the result that produced each clustering algorithm. For clarity in Figure 5 we can see the clusters when there are no inter-connectivity between the clusters.



(a) Original graph (b) Original colored graph

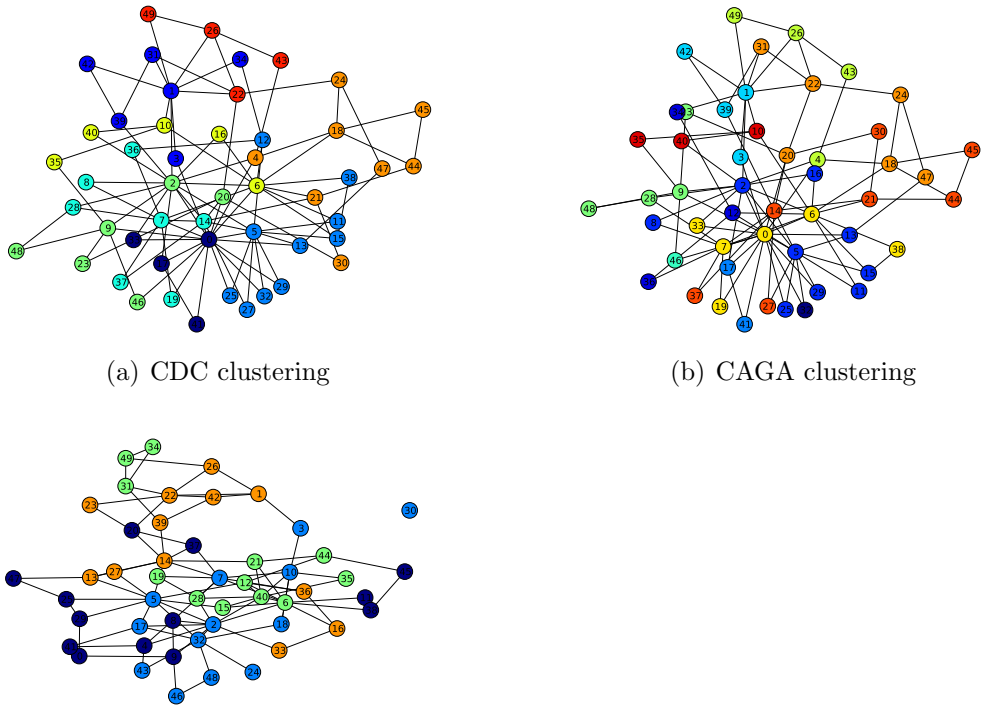
Figure 3: Base line graphs

6.2 Message Complexity

In Figure 6 are the experimental results for the message complexities for the three algorithms. It is obvious that the CDC algorithm has the worst performance. This is because in the CDC algorithm each node propagates each message to all its neighbors, this also includes the node from which it received the message. Hence its message it travel once over an edge when it reaches it destination it floods all edges including the one from which it currently arrived consequently each message will travel each edge multiple times and each time it will create a flood. Until the message is discarded from the TTL. For the other two algorithms the results are directly comparable. In Figure 7 we can observe how the algorithms scale as the number of nodes in the system increases. Note that the y-axis is in log scale. It is obvious that algorithms scale exponentially as the number of nodes in the system increases. But on the other hand these algorithms provide specific operation for handling node entry after the clustering has finished, so the need of re-clustering as the size increases is not grave.

References

[1] Stijn Van Dongen. A new cluster algorithm for graphs. Technical report, National Research Institute for Mathematics and Computer Science in the, 1998.



(a) CDC clustering

(b) CAGA clustering

(c) Schelling's clustering

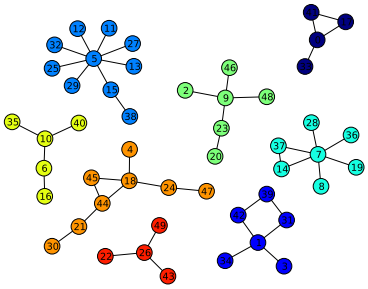
Figure 4: Clustering result

[2] Y. Li, L. Lao, and J.H. Cui. Sdc: A distributed clustering protocol. *International Journal of Computer Networks (IJCN)*, 2(6):205, 2011.

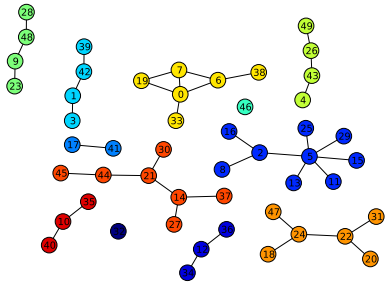
[3] L. Ramaswamy, B. Gedik, and L. Liu. A distributed approach to node clustering in decentralized peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 814–829, 2005.

[4] A. Singh and M. Haahr. Decentralized clustering in pure p2p overlay networks using schelling's model. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 1860–1866. IEEE, 2007.

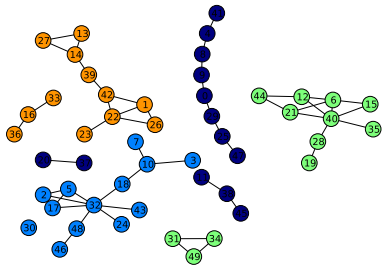
[5] Sirui Yang, Hai Jin, Bo Li, Xiaofei Liao, and Hong Yao. Worm containment in peer-to-peer networks. In *Scalable Computing and Communications; Eighth International Conference on Embedded Computing, 2009.*



(a) CDC clustering forest



(b) CAGA clustering forest



(c) Schelling's clustering forest

Figure 5: Clustering result when all inter-cluster edges have been removed.

SCALCOM-EMBEDDEDCOM'09. International Conference on, pages 308 –313, sept. 2009.

	CDC	CAGA	SCHELLING
40	52057.6	743.304	740.691
60	101241	1369.5	1233.1
80	148198	1369.38	1703.5
100	184982	2242.3	2189.4
120	226233	3164.36	2495.36
140	219495	3967.6	3138.9

Figure 6: Number of exchanged messages per algorithm per number of nodes in the system.

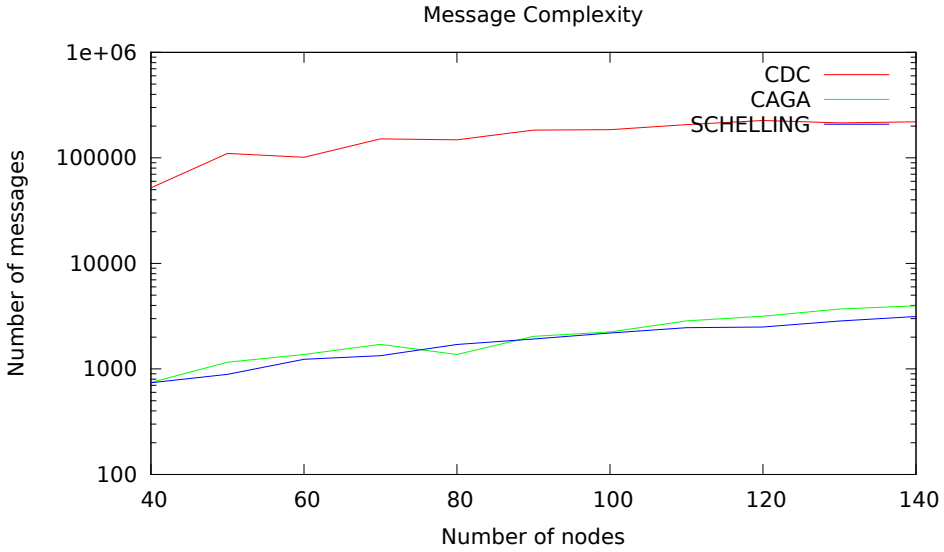


Figure 7: Comparison between the message complexities and how the evolve over time