# Comparison of Topologies of Self-Organized Randomly Scattered Wireless Networks


James Flemer, David Scherzer, Matthew Schumaker
Departments of Computer Science and Mathematical Sciences
Rensselaer Polytechnic Institute


December 5, 2002

# Contents

# Chapter 1

# Introduction

In wireless networks each "agent" or sensor node has a need to communicate with the other agents some are within its range of communication and some are not. So we would ideally like to create a communications network among the agents to allow information to flow. But we are left with an annoying trade off, if we have too many links then we will be wasting bandwidth, if there are too few links then it will take a significant amount of time for a message to reach the recipient.

In traditional wired networks, we take for granted that someone must sit down and select which computer can communicate directly, since they must put physical links between them. This process implicitly limits the number of neighbors that a given agents can have. So we do not run into the problem of having to many links in wired networks.

There is another phenomenon in wired networks that we would like to explore. No one sets out, well at least successfully implemented, a universal connectivity scheme for the Internet. Local administrators wired up their own computers and then plugged into the bigger system and a seemingly stable topology emerged. We now wish to see if we can create a stable network of wireless devices by allowing them to organize themselves into a network without interference or global knowledge.

# Chapter 2

# Graph Creation Algorithms

In this paper we plan to investigate the topologies of self-organized wireless network. In the wireless networks that we are planning to investigate nodes are scattered arbitrarily in the plane. Each node has a given range of transmission. Although it would be possible to connect each possible pair of nodes this would be a clear waste of bandwidth. So it would be ideal to find a subgraph of the possible connections. Without a central administrator the organization of the subgraph discovery algorithm must be distributed. We plan to implement several different self organization schemes and to rank them on several various variables in order to discover what the best algorithm is to find an optimal subgraph.

It is important to note that we make several basic assumptions about the system in the following creation algorithms. We assume that each node has a physical position and a radius of communication or sight. We also assume that the nodes have a way to initially communicate and a way to tell if its perceived neighbors can see it.

## 2.1 All Visible

This is the trivial algorithm in which two nodes will connect to each other if both nodes are able to communicate to each other. If the nodes are densely spread in plane then we expect that this network would be overburdened and be wasting bandwidth. We wish to find a subset of this graph that would give us a low characteristic path length while not wasting so much bandwidth.

## 2.2 Random Connections

In this model, the nodes are randomly scattered in the plane. The nodes take a few seconds to figure out which other nodes that they are able to communicate with. Each node then randomly chooses some of its neighbors to connect to. It is the hope that after each node has completed this process the network will be connected, although this is not guaranteed.

## 2.3 Preferential Attachment - Theoretical Links

In this model, just like in the previous model, the nodes are randomly scattered in the plane. For an initial period, links are added at random and then the following algorithm is used to add links to the network. The nodes take a few seconds to figure out which other nodes that they are able to communicate with. At each time step a node listens as every node announces its degree with respect to other nodes that is has chosen to connect to. With a given probability a node will try to connect to its neighbor with highest degree of those that it is not already connected to. And with a given probability the node will pick one of its neighbors randomly to connect to. At each time every node uses the above algorithm to select a neighbor to link with. This continues for some preselected amount of time.

## 2.4 Preferential Attachment - Physical Links

In this model, just like in the previous model, the nodes are randomly scattered in the plane. The nodes take a few seconds to figure out which other nodes that they are able to communicate with. At each time step a node listens as every node announces its degree with respect to the number of nodes that is can see, i.e. that it has a physical link with. With a given probability a node will try to connect to its neighbor with highest degree of those that it is not already connected to. And with a given probability the node will pick one of its neighbors randomly to connect to. At each time every node uses the above algorithm to select a neighbor to link with. This continues for some preselect amount of time.

## 2.5 Social Networks

In this model we start off again with our basic algorithm for the nodes to discover the nodes that are around them. Again the nodes use probabilities to determine which nodes that they are going to talk to. At each iteration every node receives a message from each neighbor telling it what nodes the neighbor is connected to. It then keeps track of how many of its neighbors are connected to any another given neighbor. After processing all the messages, a node knows how many connections a given neighbor has. This value is then used to weight the probability that a given node will be selected to connect to. The result being, those nodes that have many connections will have a higher probability of being connected to. Again this algorithm is repeated for each node over a number of time steps, for some set amount of time. We would expect to get a topology that is similar to a social network, that is it has a low characteristic path length but a high clustering coefficient. This topology is likely to have fewer links than a random graph to obtain a low characteristic path length, although it will probably still waste bandwidth.

## 2.6 Spaning Tree Protocol

This model builds a rooted tree in each connected component of the network after the initial neighbor discovery. In each component, a node will be elected as the root of the tree and each remaining node of the component will learn the shortest path to the root. This will produce a topology with no redundancy, but with very little wasted bandwidth. We will use two slightly different implementations of this model.

# Chapter 3

# Graph Variables

We now wish to investigate which of the above algorithms creates the ideal situation. Our networks would ideally have a small distance between any two points. One way to measure this is to find the diameter of the graph, but a better way to ensure that the distance between any two nodes is small is to look at the characteristic path length. As we suggested earlier, if a graph is highly clustered then it is very likely wasting bandwidth when it sends out a message since it sends it to many necessary neighbors, which it can reach in two hops. We can look at a measure of the clustering of a network by looking at the clustering coefficient of the graph that models the network.

Every graph has several basic variables that clearly describe it. Define $k$ as the average number of edges that each vertex is an endpoint of. We can define the function, $d(i,j)$, to be the minimum number of edges traversed to get from vertex $i$ to vertex $j$. We say a graph, $G$, is connected if there exists a path from all of the vertices of $G$ to every other vertex of $G$. This is needed to ensure that in the graph that represents our network, every vertex can be reached by every other vertex and there are no isolated vertices.

The characteristic path length, $L$, of a graph is the median of the means of the shortest path length connecting a vertex to all others over all $v \in V(G)$. The median is substituted for the mean in the definition because for large graph it becomes quite cumbersome to calculate, the median gives a very good approximation of the mean.

The neighborhood of a vertex $v$, $\Gamma(v)$, is the graph that consists of the points connected to $v$, not including $v$. We can extend this to get the $i^{th}$ neighborhood of a vertex $v$, $\Gamma^i(v)$, is the graph that consist of the points $\{x|d(v,x) = i\}$. The clustering coefficient of a neighborhood is the ratio of edges to the number of possible edges, so the clustering coefficient of $v$, $\gamma_v = \frac{|E(\Gamma(v))|}{\binom{k_v}{2}}$. Then the clustering coefficient of the graph $G$, $\gamma$, is the average of all $\gamma_v$ over all $v$. [1]

[1] Borrowed from Substrates: Are they needed in Small Worlds' Graphs?, Scherzer, Dec. 2001.

# Chapter 4

# Expectations

Our ideal graph representing the network will have not only a small characteristic path length but also a small clustering coefficient. The small characteristic path length requires that the average distance from one point to another is small, however it tells us nothing about the distribution of the path lengths. Although we wish to also observe this distribution we can also look at the clustering coefficient. The clustering coefficient is a measure of how clustered the graph is. If we want the shortest paths evenly distributed about the graph, then we can not have a high clustering coefficient because all of the short paths would necessarily be contained in small patches in the graph. So as far as our two graph variables, we would like to obtain a network that has a small clustering coefficient and a small characteristic path length.

The distribution of the shortest path lengths is going to be a key to looking at what kind of network that we are looking at. In both scale-free and small world topologies the distributions are linear on a log-log plot. This is indicative of many very short paths, then a few medium length paths and almost no, although some do exists, longer paths. This would be our ideal situation, since we have seen in other networks with this property that they are almost at a minimal critical point. That is that if we try to change the topology using the same number of edges and vertices that is almost impossible to recreate a graph with such a small characteristic path length. If we get a distribution that is something more like a bell curve, we know that this graph could probably be generated purely with a statistical rule and that out topology is not ideal since most of the shortest paths are of medium length, as opposed to being minimal.

Of our graph creation algorithms, none explicitly stand out as giving us exactly the graph that we would like to see as our ideal network, but they each offer a glimmer of hope. We expect that the network formed using every possible link will likely have a small characteristic path length, but it will probably have a high clustering coefficient. We also expect the same kind of statistics from the social networks algorithm, however we would expect the social networks algorithm to have a 'nicer' shortest path length distribution, i.e. that is linear on a log-log plot. Both of the preferential attachment models should produce Internet like graphs which would be ideal. The preferential attachment model that is reliant on theoretical links, might run into the problem of being limited by its number of physical links. So this algorithm may not be able to reach its full potential. However on the converse the preferential attachment model that is reliant on the physical links does not take into account which links actually exist in the network that it is creating, so it may leave several nodes very far away from the rest of the giant component. The spanning tree protocol should ideally produce a spanning tree. This may appear to be efficient, unless the tree is poorly rooted, unbalanced or fairly large. This may give us a very high characteristic path length, although we are just about guaranteed to get a very small clustering coefficient. In the random edge creation model it is uncertain what kind of graph that we will get.

It is important to remember that in our models that we are looking at a very small subset of all possible graphs on our vertices. We are limited by the "visibility" of each agent. We can not have edges that go from one corner of our network to the other since it is very unlikely that two agents will have such a large vision. So we are almost limited to just local links which Watt's cites as a property that will not allow for the creation of a small worlds topology. If this is the case, we may have to accept the fact that our characteristic path length is going to be large and just try to minimize it or just try to minimize the clustering in the graph without making our already large characteristic path length any larger.

# Chapter 5

# Implementation

## 5.1 Preferential Attachment - Theoretical Links

The Theoretical Links model of Preferential Attachment works by having nodes prefer to connect to nodes with high numbers of connections.

The nodes in this model do not contain any additional information other than who they are connected to. At each time-step, a given node will broadcast how many other nodes it has connections to. This message contains the node ID how many connections it has. While a node is processing its messages it keeps track of which neighbor had the highest number of connections. During the initial time-step, each node announce that is has zero connections to each of its neighbors. During the second time-step, which ever message is the first will be used as the largest. The node will then choose that node with a 90% probability to connect to, otherwise it will connect to some other neighbor randomly. Once a neighbor connects to another one, it will never attempt to connect to that neighbor again, even though that neighbor may have a much larger number of connections than any other neighboring nodes.

This algorithm converges very quickly only requiring 3 or 4 iterations before the graph become connected. As a node becomes more connected, it will continue to get more connections. Obviously those nodes that get more than one connection initially will become the more connected nodes as the algorithm progresses. However since the same node cannot be connected to twice, like a secondary node will become very connected as well results in redundancy of the network. Looking at the produced graphs as well as degree distributions it becomes evident that the majority of nodes only get 2 connections will a minority become hubs.

## 5.2 Preferential Attachment - Physical Links

This model is very similar to the Theoretical Links model of Preferential Attachment. The difference being that the Physical model prefers to connect to those nodes that can see the most while the Theoretical model prefers to connect to those that have the most established connections. The message passing and algorithm remain the same except for this key difference.

Like the Theoretical model, the Physical model converges very quickly requiring only 3 or 4 iterations before a complete graph is formed. The difference between the networks generated is that more nodes become hubs and with lesser degrees than in the Theoretical model.

## 5.3 Social Network

The Social Network model basic idea is that a given node wants to connect to those nodes that its neighbors are connected to.

The nodes in this model do not need to store any additional information other than ID and what neighbors it has connected to. During the initial time-step, each node randomly selects one of its neighbors to connect to. This is then broadcasted to each of its neighbors. During the next, and each subsequent time-step, th node goes through all the messages it gets keep track how many connections, from a any given neighbor, each of its neighbors has. This count is used to weight each neighbor for the selection process. The node then chooses a node. The weight of a node is used to increase the probability of a given node being selected. The results being the more neighbors connected to a common node, the greater the chance the common node will connected to.

This model converges very quickly requiring only 3 or 4 iterations before a complete graph is created. The produced graph will have the characteristics that a majority of the nodes will have a few connections, and then there will be a decreasing number of nodes having more connections, with a few nodes having many connections.

## 5.4 Spaning-Tree Protocol (Hop-Count)

The spaning tree protocol (STP) builds a rooted tree in each connected component of the network. In each component, the node with the lowest ID will become the root of the tree and each remaining node of the component will learn the shortest path to the root.

The nodes in this model contain of four things: their own node ID, the node ID of the root, the number of logical hops to the root, the node ID of the next-hop towards the root (the parent). At each time-step, each node broadcasts a message (advertisement) to every node within its vision. This messages contains the ID of the sender, the ID of the node it believes is the root, and the number of hops it is from the root. When each node initializes, it assumes that it is the root node, and therefor the distance to the root is zero hops. For each message received, the node examines the root ID, and the hop count. If the advertised root has a lower ID than the current root, the sender is made the next-hop towards the root, and the root ID and hop count are updated. Similarly if the root IDs are the same and the hop count is smaller, the next-hop is changed to the sender and the next-hop and count updated.

This algorithm converges very quickly, and has low processing overhead. It can easily be seen that it will take a number of time-steps equal to the distance from the root to the node farthest from it. This value has an upper bound of the diameter of the network.

## 5.5 Spaning-Tree Protocol (Distance-Sum)

The Distance-Sum Spaning-Tree Protocol is very similar to the Hop-Count implementation. The distance-sum variation includes one additional field, the sum of the edge lengths back to the root. This value is sent in each advertisement. If a node receives a smaller distance-sum with the same root and hop-count, it will change its next hop to the sender of the advertisement.

The convergence of this algorithm is identical to the Hop-Count implementation.

## 5.6 Random Connect (N-neighbors)

In the Random Connect model, each node arbitrarily chooses $N$ of the nodes within its vision to "peer" with. The node then sends a message containing its own ID to the "peer", notifying this other node that is has been chosen as a peer. A node that receives a message from another node, adds the senders ID to its list of peer, thus each link is bi-directional. This algorithm converges in two time-steps, one to choose peers and send messages, and one to process the messages sent in the first step.

## 5.7 Random Connect (P-probability)

This second Random Connect model is nearly identical to the first. The difference is in how neighbors are chosen to be peers. In this model each neighbor is considered individually, and with probability $P$ is chosen to be a peer. The messages passed are the same as in the previous random connect model.

# Chapter 6

# Results

After creating 3 scenarios with 50, 100 and 300 sensors each, we ran each algorithm on the same data. We recorded and analyzed the characteristic path lengths, clustering coefficients, shortest path distribution and the degree distribution and choose a winning algorithm.

## 6.1 Characteristic Path Length

We ideally wanted to have a small characteristic path length. The random connection model gave us small characteristic path length, when it managed to connect the graph. When the visible network was sparse neither of the random connection models where able to form a connected network.

Our next best performing model was the social network model. It consistently formed a subgraph that was convenient to navigate. For a network of 300 nodes it could be expected that one node could reach any other in only 7 hops. And we would expect that this number would not grow significantly as the number of sensors is increased.

Both preferential attachment models performed well with respect to characteristic path length. This would be fairly well expected. Just as with the social networks model we have seen this model consistently build networks that can be traversed fairly quickly. For a network with 100 nodes, the average distance between any two vertices on average is under 6. The fact that the characteristic path length for the subgraphs with 50 and 300 nodes are very close suggests that the characteristic path length will not grow significantly as the number of nodes increases.

Lastly, the spanning tree algorithm came in with significantly larger characteristic path lengths. This is not surprising to to get from one node to the other you may have to go up one side of the tree and down another. This can be exceptionally long, especially if the tree is poorly rooted or unbalanced.

| Number of Sensors | Algorithm | Characteristic Path Length |
|---|---|---|
| 50 | DSTP | 8.484848485 |
| 50 | PrefPhy | 5.949494949 |
| 50 | PrefThe | 7.535353535 |
| 50 | RandomN | 8.222222222 |
| 50 | RandomP | 5.393939394 |
| 50 | STP | 8.02020202 |
| 50 | Social | 5.888888889 |
| 100 | DSTP | 11.8729097 |
| 100 | PrefPhy | 9.822742475 |
| 100 | PrefThe | 10.30100334 |
| 100 | RandomN | 12.34782609 |
| 100 | RandomP | 7.642140468 |
| 100 | STP | 11.43812709 |
| 100 | Social | 8.869565217 |
| 300 | DSTP | 10.85714286 |
| 300 | PrefPhy | 9.836734694 |
| 300 | PrefThe | 10.02040816 |
| 300 | RandomN | Not Connected |
| 300 | RandomP | Not Connected |
| 300 | STP | 9.510204082 |
| 300 | Social | 7.448979592 |

Figure 6.1: The Characteristic Path Lengths of our networks.

## 6.2 Clustering Coefficient

When looking at the clustering coefficients, is when we really saw the different models start to pull away from each other. Our ideal clustering coefficient is small, but not too small. This paradox represents the tradeoff between redundancy and the waste of bandwidth. We would like our network to remain stable if a link or two are broken. However we do not wish to over connect all of the nodes resulting in the networking being overburdened by a single transmission.

The social networks had the largest clustering coefficients through out the simulations. This is not surprising because the algorithm tends to create cliques. Although we may see this paradigm in social networks, it is sub-optimal for computer networks since nodes close the originator of a message will see the message several times before the message reaches the recipient. So this algorithm is not optimal.

Both of the spanning tree methods had a clustering coefficient of zero, which is what we expected because of the fact that they produce trees. Although this reduces the waste of bandwidth that we saw in the social network it destroys the robustness of the network. If one connection were to fail then the network would become disconnected. In an environment where things are constantly changing we would like to have some stability.

Both of the preferential attachment models performed well considering their clustering coefficients. This is to be expected since they are not necessarily forming cliques but might form cliques when two adjacent nodes try to connect to a central high degree node. The model that gave preference to nodes that had a high number of neighbors gave a more reasonable clustering coefficient than the model that gave preference to nodes that had chosen to talk to a large number of nodes. This is probably due in part to the fact that if a node is initially given many links there is no guarantee that it can continue to be connected to since it may be in an isolated part of the graph.

| Number of Sensors | Algorithm | Clustering Coefficient |
|---|---|---|
| 50 | DSTP | 0 |
| 50 | PrefPhy | 0.497177 |
| 50 | PrefThe | 0.490714 |
| 50 | RandomN | 0.223143 |
| 50 | RandomP | 0.176667 |
| 50 | STP | 0 |
| 50 | Social | 0.60446 |
| 100 | DSTP | 0 |
| 100 | PrefPhy | 0.342747 |
| 100 | PrefThe | 0.599476 |
| 100 | RandomN | 0.10031 |
| 100 | RandomP | 0.305547 |
| 100 | STP | 0 |
| 100 | Social | 0.513223 |
| 300 | DSTP | 0 |
| 300 | PrefPhy | 0.340518 |
| 300 | PrefThe | 0.50861 |
| 300 | RandomN | 0.062786 |
| 300 | RandomP | 0.288352 |
| 300 | STP | 0 |
| 300 | Social | 0.485543 |

Figure 6.2: The Clustering Coefficients of our networks.

## 6.3 Preferential Attachment

After observing these results we decided that the preferential attachment model that was dependent on the number of visible neighbors was the optimal algorithm that we investigated for finding a subgraph. After doing this we decided to look more in depth into the resulting graphs from the preferential attachment model.

### 6.3.1 Path Distribution

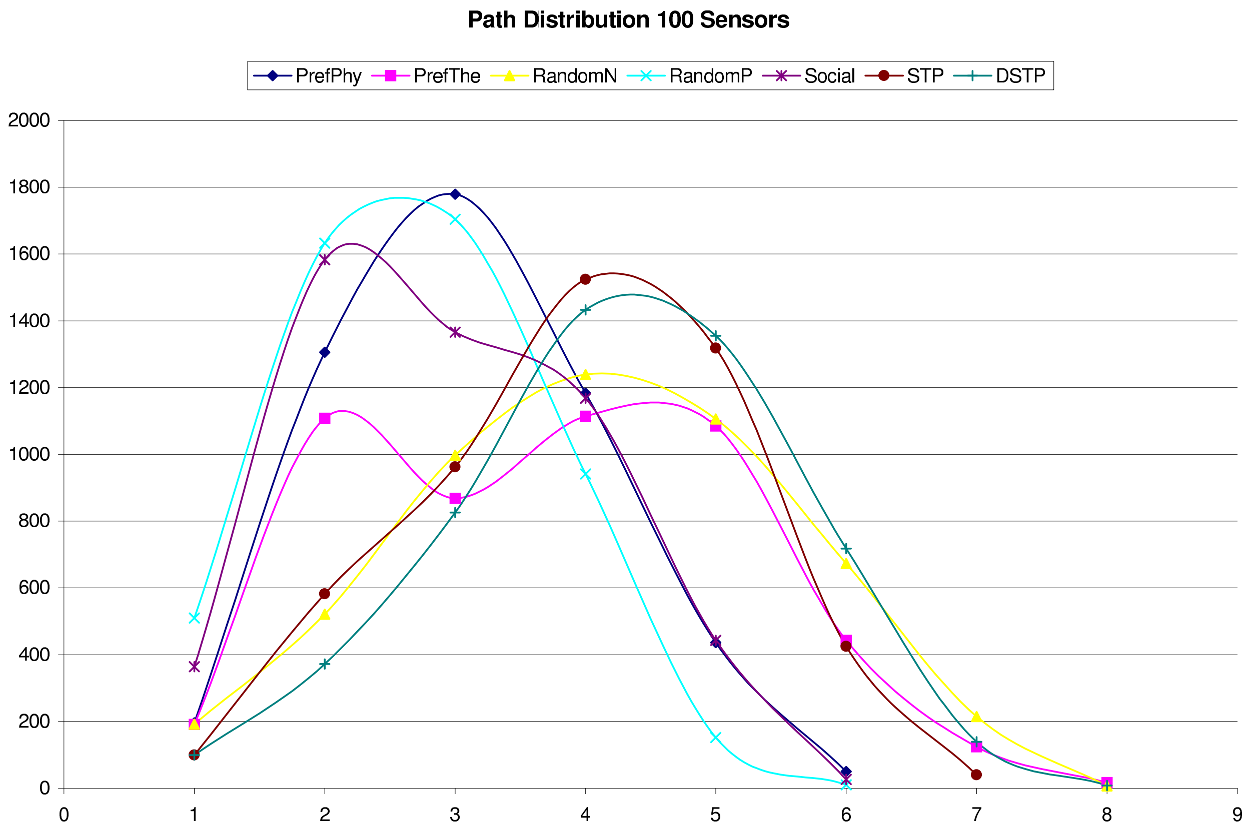


Figure 6.3: The shortest path lengths distribution for networks with 100 vertices.

We can easily see that most of the paths for the preferential attachment model based on physical links are not only short but they are congregated around 3. Most of the shortest paths are about the same length. We even have a diameter of 6 for a graph of 100 vertices. There is also not a lot of variation in the distribution as there is with many of the other models. Having a small variance tells us that we do not have many long shorts paths and many small shortest paths but a significant number of short paths and a few very short paths and a few longer paths.

### 6.3.2 Degree Distribution

We then went on to investigate the distribution of the vertex degree.

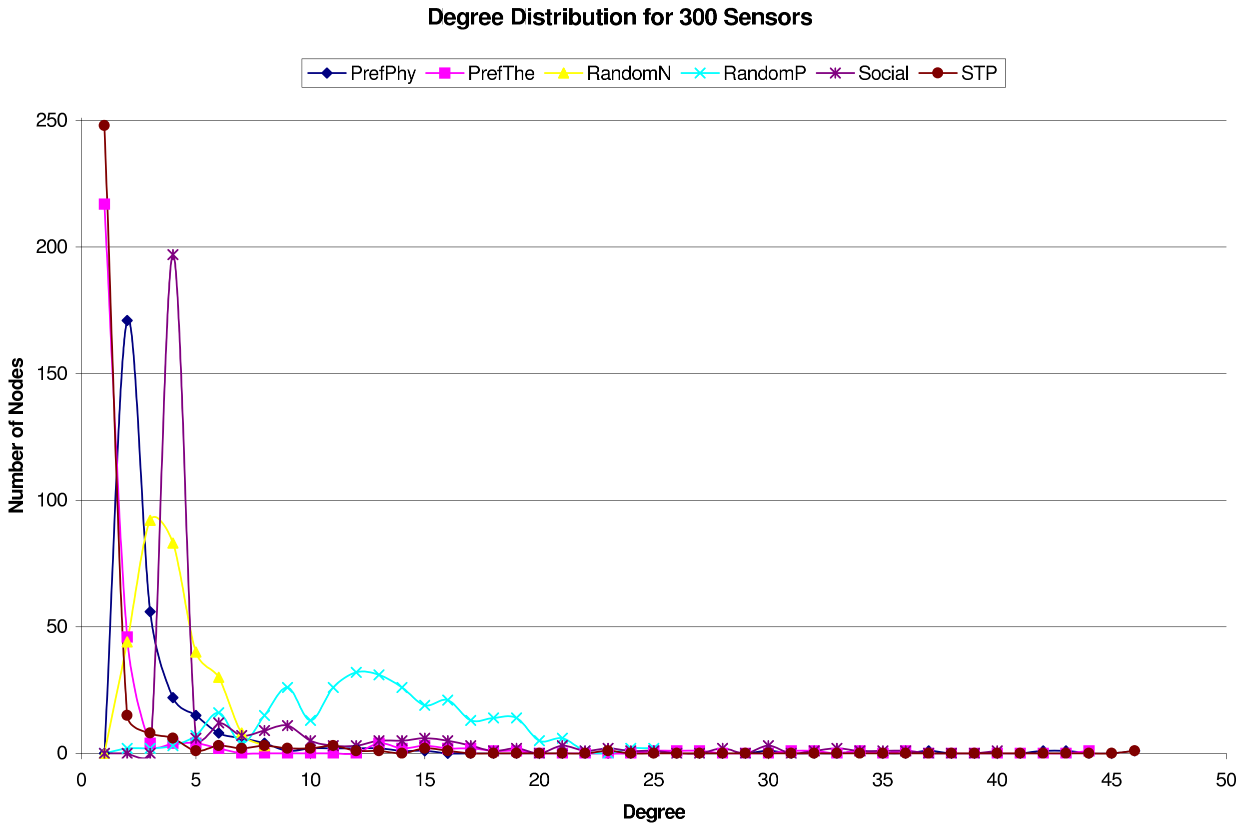


Figure 6.4: The degree distribution for networks with 300 vertices.

Looking at figure 6.4 it is hard to see, but we note that most of the nodes have a low degree and there are a few with higher degrees. Looking at the actual data for the preferential attachment model wee see that more than half of the nodes have degree 2 and that no nodes have degree 1. This not only means that every node will remain connected if a link is interrupted but that the nodes have arranged them selves in such a way that every node has several links but the graph is not overly clustered.

### Self-Organized Criticality?

Out of curiosity we examined the degree distributions on a log-log scale, after we had already selected the preferential attachment model as our winner and we noticed an interesting phenomenon.

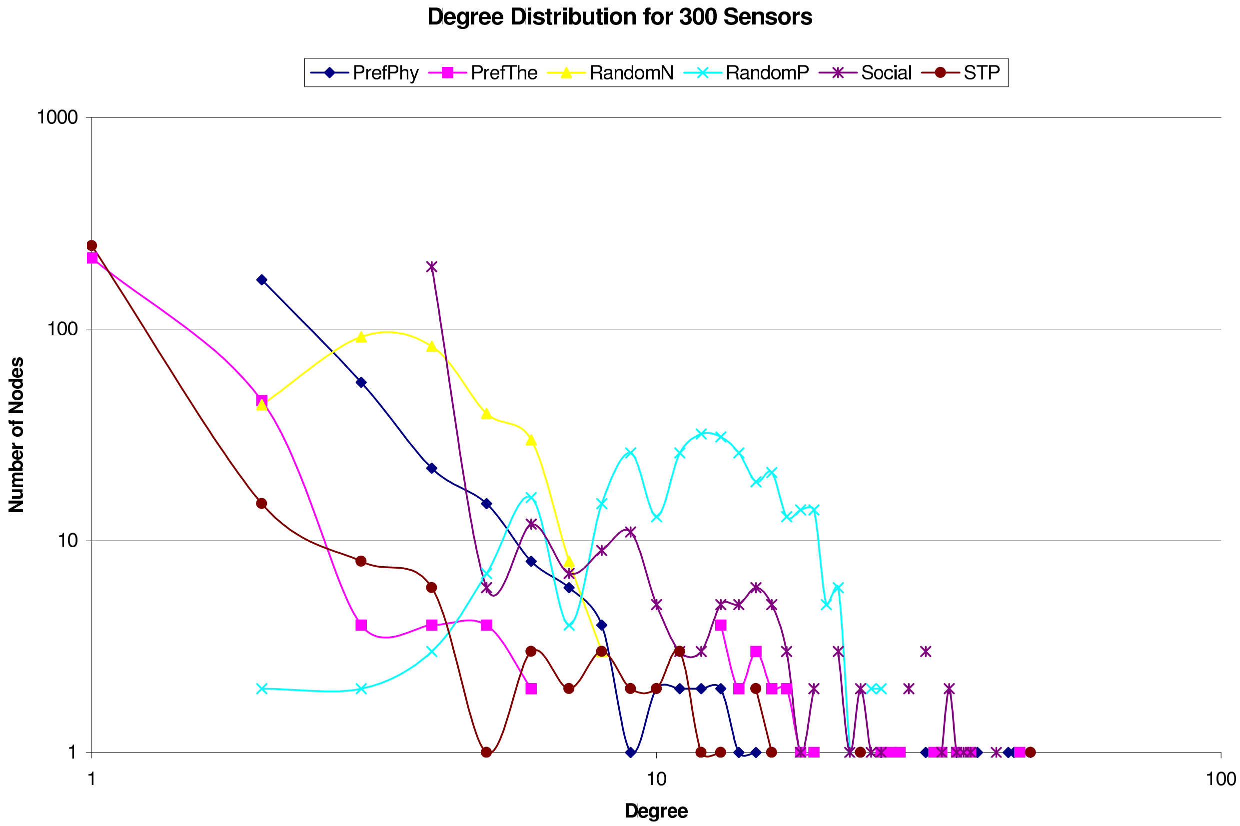


Figure 6.5: The degree distribution for networks with 300 vertices on a log-log scale.

We notice that the degree distribution for the preferential attachment model that is dependent on physical links is nearly linear. (see figure 6.5) This is exactly a feature that to paraphrase Per Bak is the signature of self-organized criticality. It is rather clear that our network is self-organized. It is less obvious that it is in a critical state. It is critical in the sense that if we were to take the same network and rewire it we we quickly loose the properties of the network that caused us to chose this algorithm as the best generator of a subgraph.

## 6.4 Future Work

Although this work solves several interesting questions it makes several large assumptions and creates several new questions. It assumes that the network is static. We would be interested in dynamic networks either through changing vision or removal and addition of nodes or links. Both of these areas would be very interesting to observe. Also these models were tested on relatively small graphs, it would be good to test them on large more realistically sized graphs. It would also be pertinent to test such algorithm in 3-space instead of in the place.