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

Introduction

- “Agents” or sensors have a need to communicate with each other.
- We want to create a network among them to allow information to flow.
- Trade off between wasting bandwidth and the time for a message to reach the recipient.
- We are in a different situation than in wired networks.

Wired Networks

- In traditional wired networks physical links must exist between computers.
- This implicitly limits the number of neighbors that a given agents can have.
- Organization of wired networks
  - No one set out, well at least successfully, a universal connectivity scheme for the Internet.
  - Local administrators wired local networks and then plugged into a bigger system and a topology emerged.
- We now wish to see if we can create a stable network of wireless devices through self-organization without interference or global knowledge.

Graph Creation Algorithms

- Nodes are scattered arbitrarily in the plane.
- Each node has a given range of transmission.
- Connecting each possible pair of nodes is a clear waste of bandwidth.
- Idea is to find a subgraph of the possible connections, that is optimal, without a central administrator.
- We plan to implement several different self organization schemes and to rank them on several variables in order to discover which is the best algorithm for finding a subgraph.

Subgraph Discovery Algorithms

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.
- The nodes have a way to initially communicate and a way to tell if its perceived neighbors can see it.

Models

- All Visible
- Preferential Attachment
- Social Networks
- Spaning Tree
- Random Connect

All models assume that initially, nodes are randomly scattered in the plane. Then the nodes take a few seconds to figure out which other nodes that they are able to communicate with.

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 placed in the plane then we expect that this network would be overburdened and waste 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.

Preferential - Theoretical Links

In this model, 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 90% 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 10% 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.

Preferential - Physical Links

This model is similar to the previous. At each time step a node listens as every node announces its degree with respect to the number of nodes that it can see, i.e. that it has a physical link with. With a 90% 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 10% probability the node will pick one of its neighbors randomly to connect to.

Social Networks

In this model we start off again with our basic algorithm for the nodes to discover who is around them. At each time-step a node will try to select a neighbor of one its neighbors. The way these are selected is through a weighted probability where each candidate node gets more weight for each unique path of length 2 that exists from the node to the candidate node. 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.

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.

Random Connect

Following the placement and initialization, each node in this model, randomly chooses some subset of its neighbors to connect to. Those neighbors are then notified that they have been selected. It is the hope that after each node has completed this process the network will be connected, although this is not guaranteed.

Graph Variables

- 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.
- A better way to ensure that the distance between any two nodes is small is to look at the characteristic path length.
- If a graph is highly clustered then it is very likely wasting bandwidth.
- 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.

Characteristic Path Length

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.

Clustering Coefficient

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 ith 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, γ, is the average of all γ_v over all v.

Expectations

Our ideal graph representing the network will have:

- A small characteristic path length.
- Shortest paths evenly distributed about the graph.
- A small clustering coefficient.

Distributions

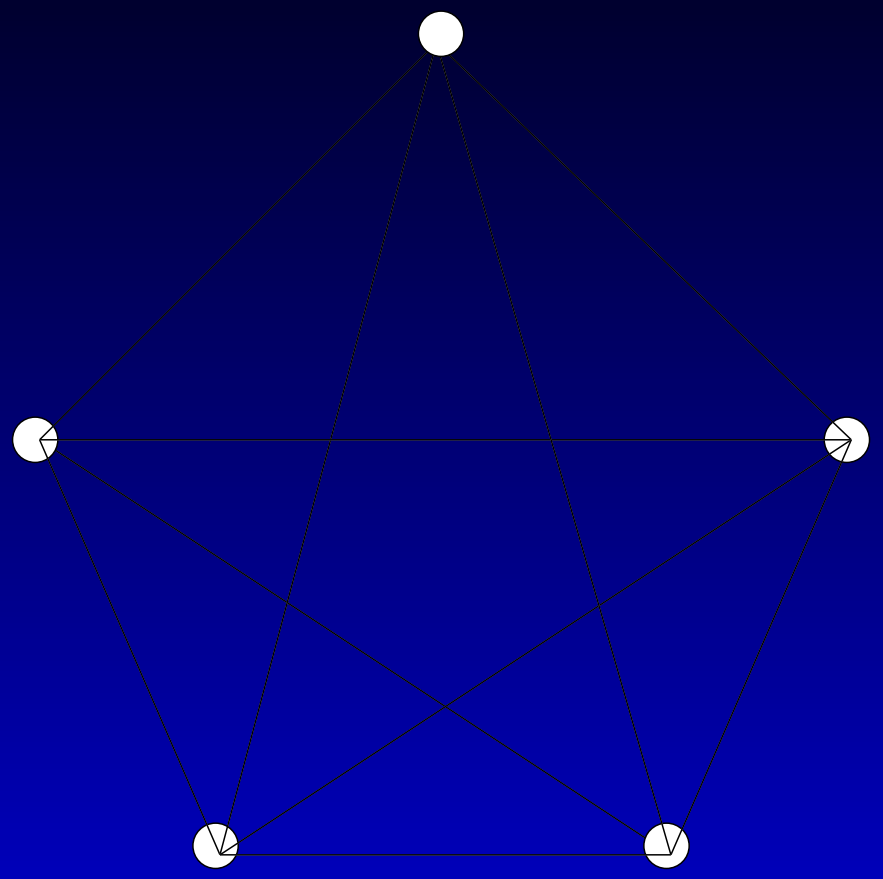
The distribution of the shortest path lengths is going to be a key to looking at what kind of network that was constructed. In both scale-free and small world topologies the distributions are linear on a log-log plot. This is indicative of many very short paths, then a few medium length paths and almost no, although some do exists, longer paths. This would be our ideal situation.

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.

Expected Performance

- We would expect the social networks algorithm to have a ‘nicer’ shortest path length distribution, i.e. that is is linear on a log-log plot.
- Both of the preferential attachment models should produce Internet like graphs which would be ideal.
  - Theoretical v. Physical
- The spanning tree protocol should be efficient, unless the tree is poorly rooted, unbalanced or fairly large.
- In the random edge creation model it is very uncertain what kind of graph that we will get.

Implementation



Preferential - Theoretical Links

1. Each node announces its number of connections
2. Each node sorts through the messages and selects the node with the most connections
3. The node chooses to connect to that node with a 90% probability

Preferential - Physical Links

1. Each node announces its number of neighbors
2. Each node sorts through the messages and selects the node with the most connections
3. The node chooses to connect to that node with a 90% probability

Social

1. Process the neighbors connected to by each neighbor
2. Weight each neighbor using the number of common neighbors that connect to it
3. Use the weight to create a weighted probability for use in selecting a node to connect to
4. Add the selected node to the list of neighbors and broadcast it to all neighbors

Spaning Tree Protocol

- Goal is for each device to learn their parent in the spaning tree
- Devices begin by assuming they are the root
- Broadcast a message every time-step with the following fields:

Root Device	Hops to Root	Source Device

- Device chooses a new parent when
  - See a message with a lower root device
  - See a message with same root, but fewer hops
- Converges after no device changes parent for a set number of time-steps

Spaning Tree Protocol (D)

- Variation on first spaning tree protocol
- Add an additional field in messages *Distance*
- Distance is aggregate physical distance between hops
- Add an addition condition for changing parent
  - Same root device, same number of hops, smaller distance

Random Connect (N)

- Completes in two time-steps
- First time-step
  - Device chooses N of the devices visible to it as peers
  - Device sends messages to each of the chosen devices with its own device ID
- Second time-step
  - Devices receive messages sent on first step, and add the senders to list of connected neighbors

Random Connect (P)

- Essentially the same as the first Random Connect model
- Instead of connecting to N neighbors, connects with probability P to some subset of its neighbors

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.

Characteristic Path Length

# of Sensors	Algorithm	C.P.L.
50	DSTP	8.484
50	PrefPhy	5.949
50	PrefThe	7.535
50	RandomN	8.222
50	RandomP	5.393
50	STP	8.020
50	Social	5.888
100	DSTP	11.872
100	PrefPhy	9.822
100	PrefThe	10.301
100	RandomN	12.347
100	RandomP	7.642
100	STP	11.438
100	Social	8.869

# of Sensors	Algorithm	C.P.L.
300	DSTP	10.857
300	PrefPhy	9.836
300	PrefThe	10.020
300	RandomN	N/C
300	RandomP	N/C
300	STP	9.510
300	Social	7.448

Characteristic Path Length cont.

- We want to have a small characteristic path length
- The random connection models gave us small characteristic path length, when it managed to connect the graph.
- The social network model also produced a small characteristic path length.
- Both preferential attachment models performed well with respect to characteristic path length.
- The spanning tree algorithms came in with significantly larger characteristic path lengths.

Clustering Coefficient

- Our ideal clustering coefficient was small, but not too small.
- There exists a tradeoff between redundancy and the waste of bandwidth.
- We would like our network to remain stable if a link or two are broken.
- We do not wish to over connect all of the nodes resulting in the networking being overburdened by a single transmission.

Clustering Coefficients cont.

# of Sensors	Algorithm	C. C.
50	DSTP	0.000
50	PrefPhy	0.497
50	PrefThe	0.490
50	RandomN	0.223
50	RandomP	0.176
50	STP	0.000
50	Social	0.604
100	DSTP	0.000
100	PrefPhy	0.342
100	PrefThe	0.599
100	RandomN	0.100
100	RandomP	0.305
100	STP	0.000
100	Social	0.513

# of Sensors	Algorithm	C. C.
300	DSTP	0.000
300	PrefPhy	0.340
300	PrefThe	0.508
300	RandomN	0.062
300	RandomP	0.288
300	STP	0.000
300	Social	0.485

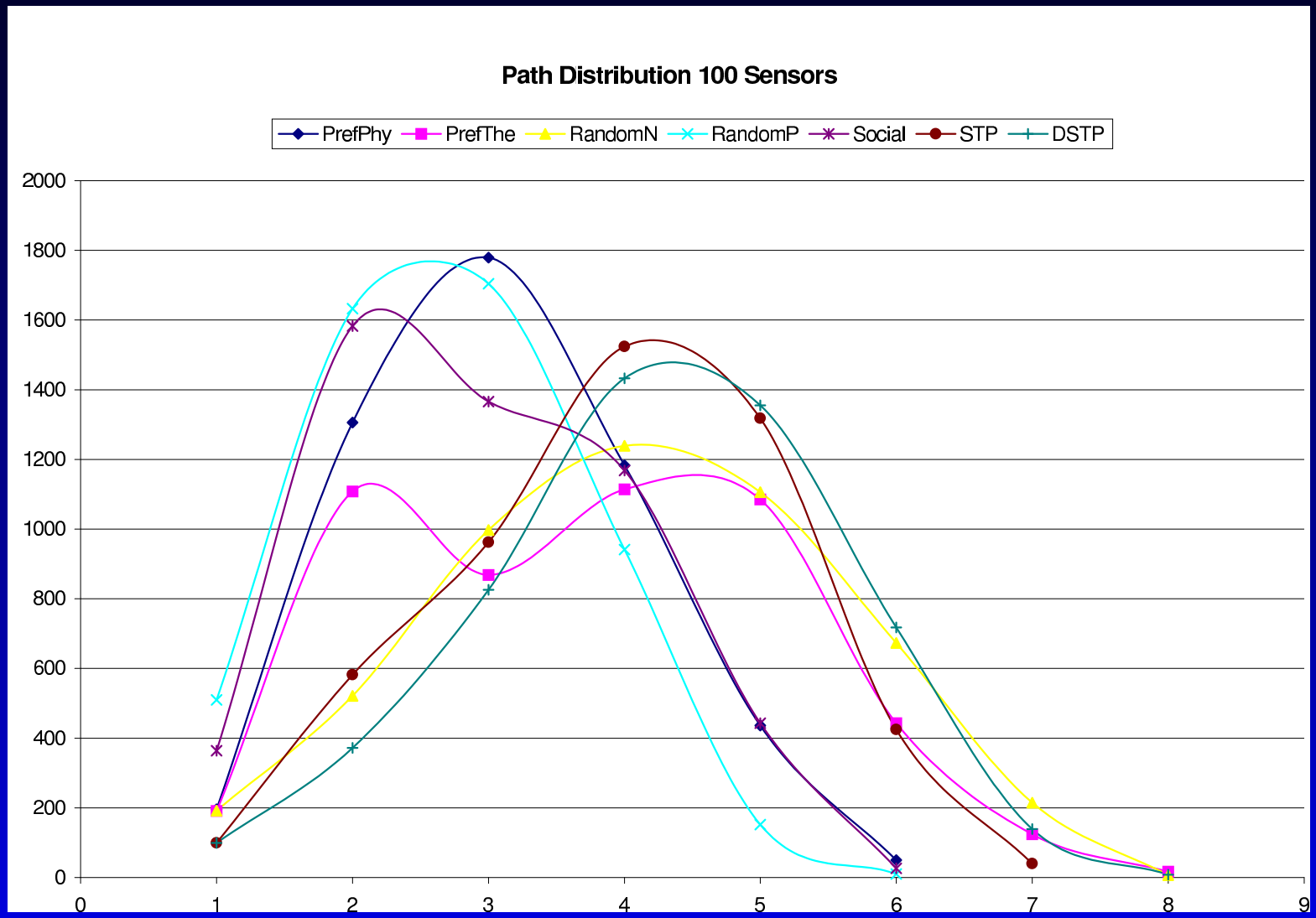
Clustering Coefficient cont.

- The social networks had the largest clustering coefficients.
- Both of the spanning tree methods had a clustering coefficient of zero.
- The preferential attachment models performed well considering their clustering coefficients.
- The random algorithms performed the best but we disqualified them since they failed to connect the graphs.

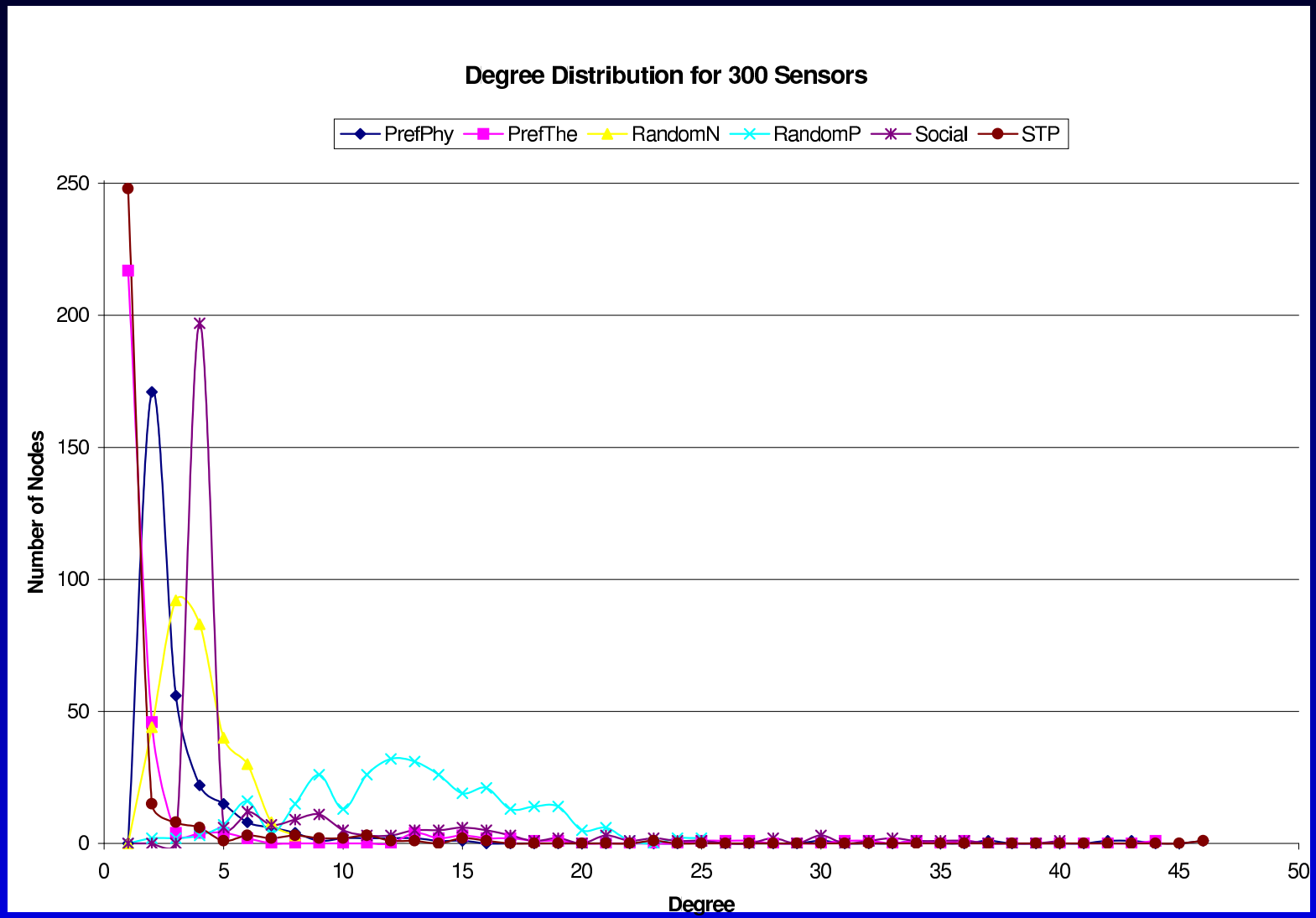
Preferential Attachment

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.

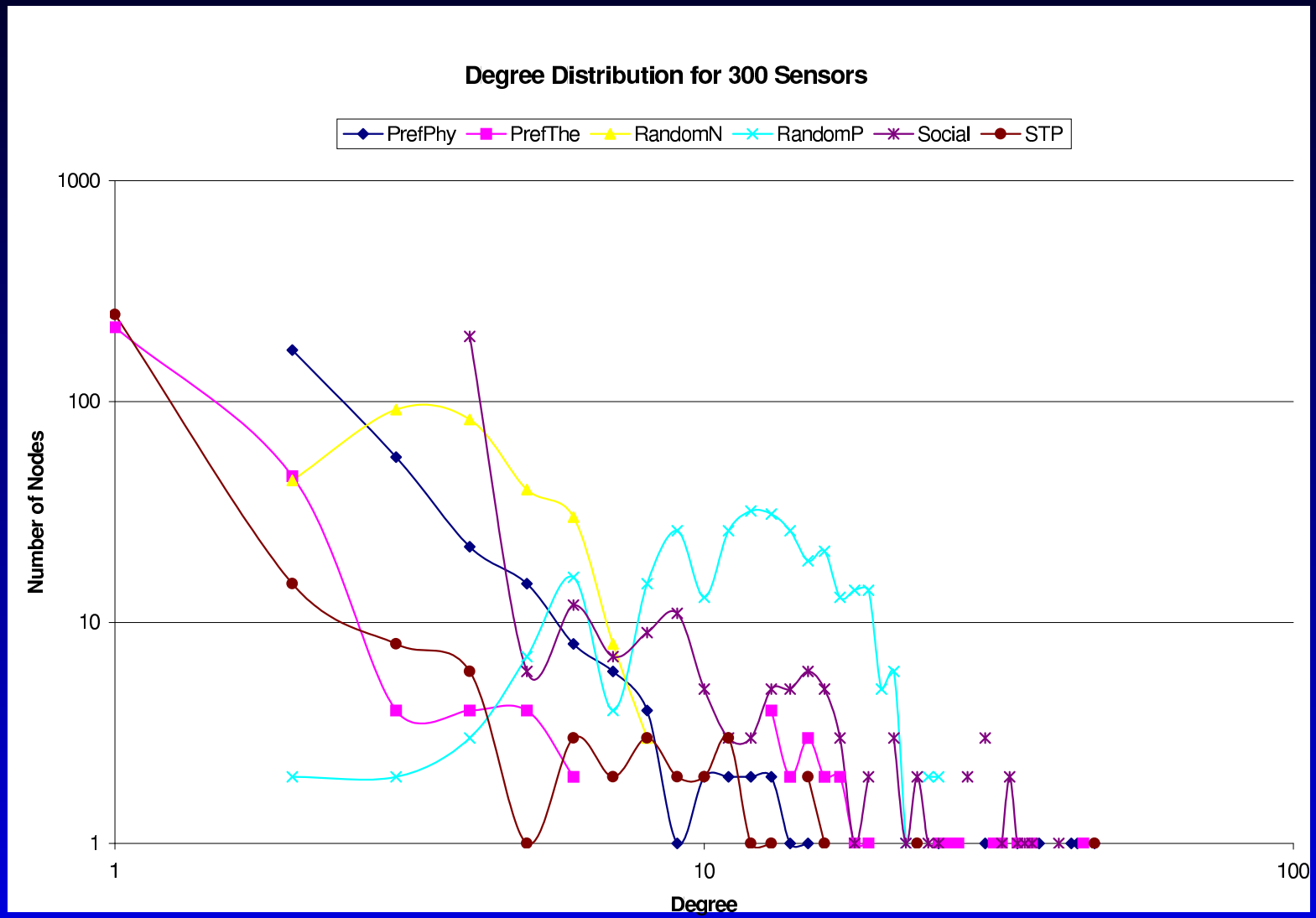
Shortest Path Distribution



Degree Distribution



Self-Organized Criticality?



Future Work

- Dynamic Networks - through changing vision or removal and addition of nodes or links.
- Testing these algorithms on larger more realistically sized graphs.
- It would also be pertinent to test such algorithm in 3-space instead of in the plane.

Questions

Slides produced with Prosper and LaTeX.

http://prosper.sourceforge.net/