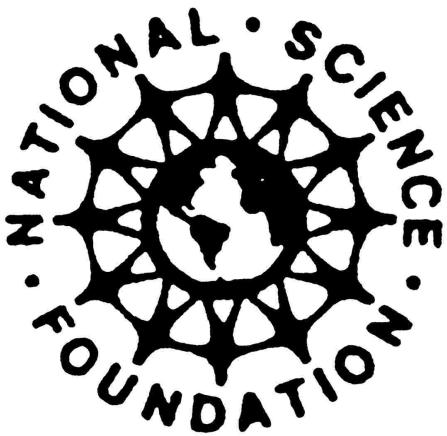
Quasi-Random Rumor Spreading Protocol

Efficiency and Topologies

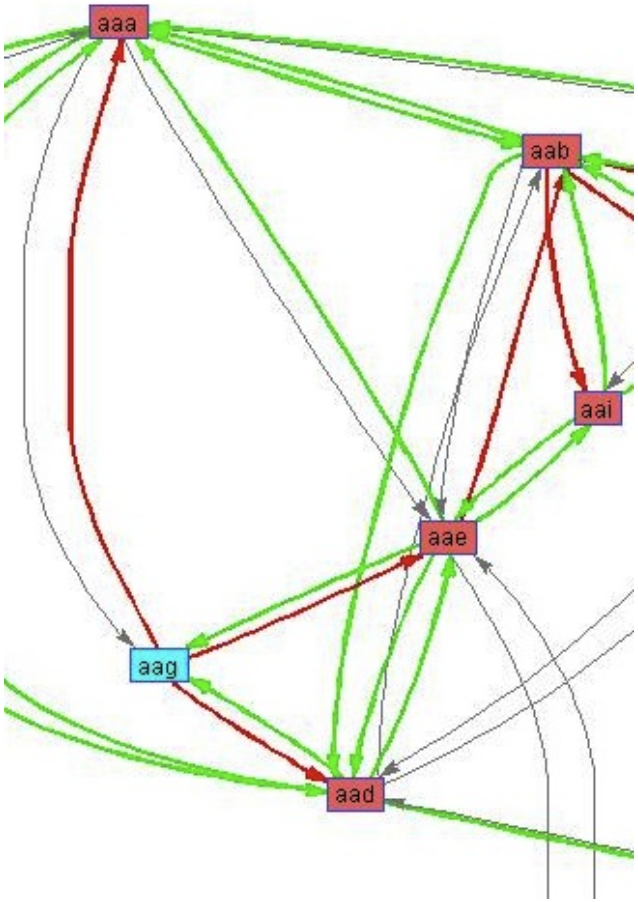
Charles Drake Poole



Rumor spreading protocols are classically used to described how information is spread across a network. Doerr-Huber-Levavi built upon a randomize protocol described at SODA 2008 and discussed the robustness in the communication. We test the efficiency of a randomized and quasi-random rumor spreading protocol on several topologies. (Static Topologies). Also tested is rumor spreading on grids with non-stationary agents. (Dynamic Topology)

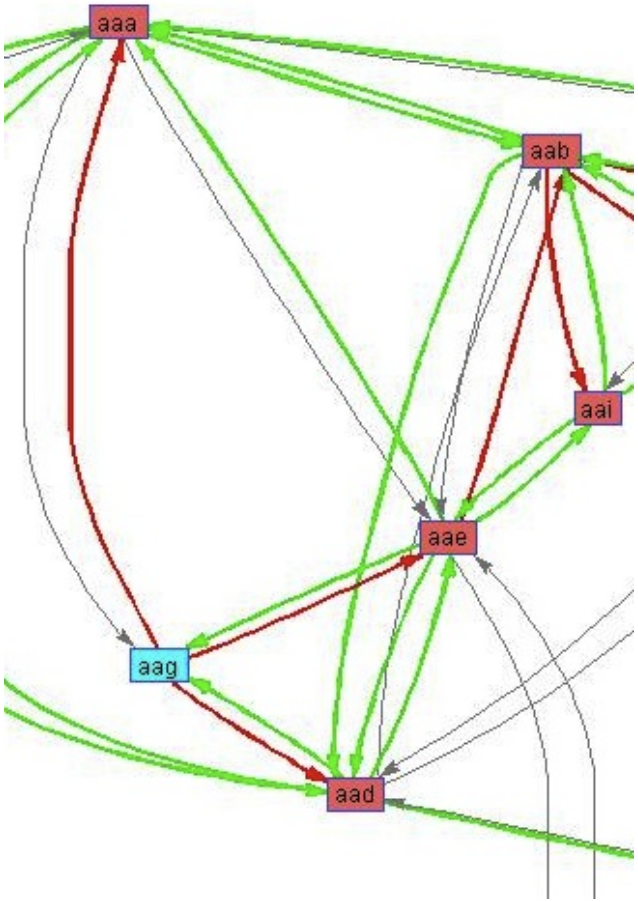
- Quasi-Random rumor spread has rules beyond spray and pray (Classic "Randomized rumor spread method")
- How do we know it's penetrated the whole network?
- Are these protocols incredibly wasteful?
- Is "quasi" the right word for the Doerr-Huber-Levavi scheme?

The Doerr-Huber-Levavi method is very straight forward.



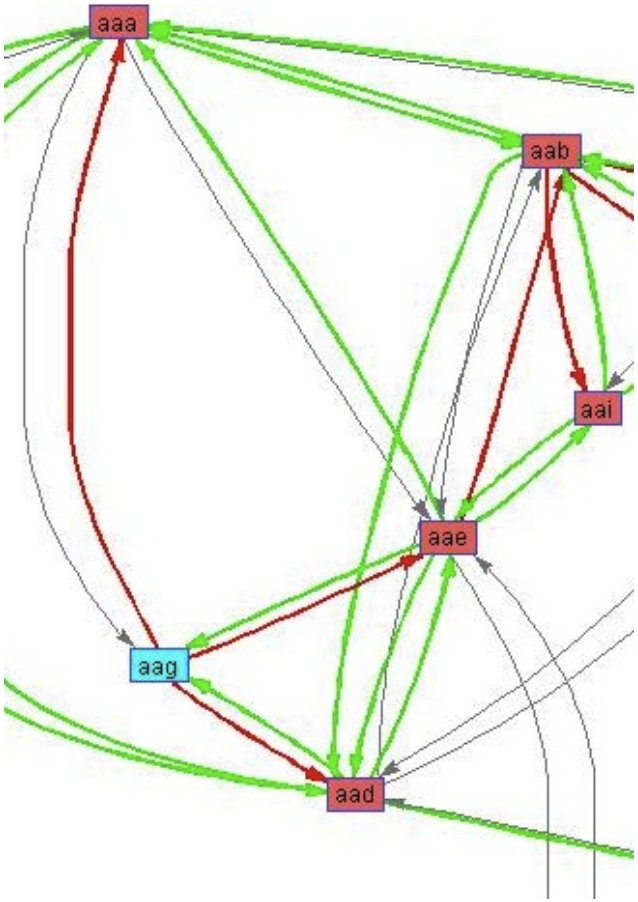
1. Do I know?
2. Who can I tell?
3. Choose a node.
4. Inform that node this timestep.
5. Repeat starting with "Do I know?"

The Doerr-Huber-Levavi method is very straight forward.



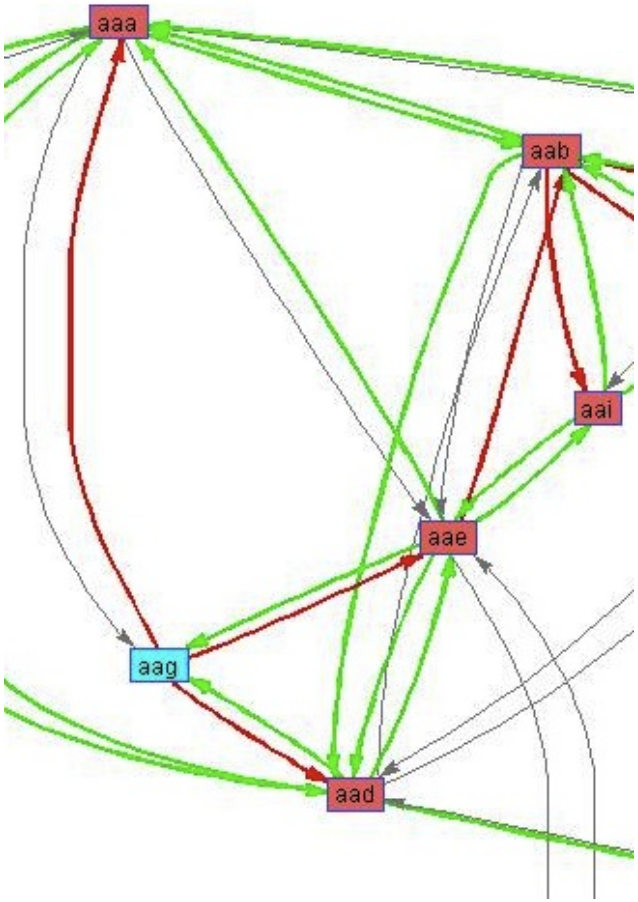
1. Do I know?
2. Who can I tell?
3. Choose a node.
4. Inform that node this timestep.
5. Repeat starting with "Do I know?"

The Doerr-Huber-Levavi method is very straight forward.



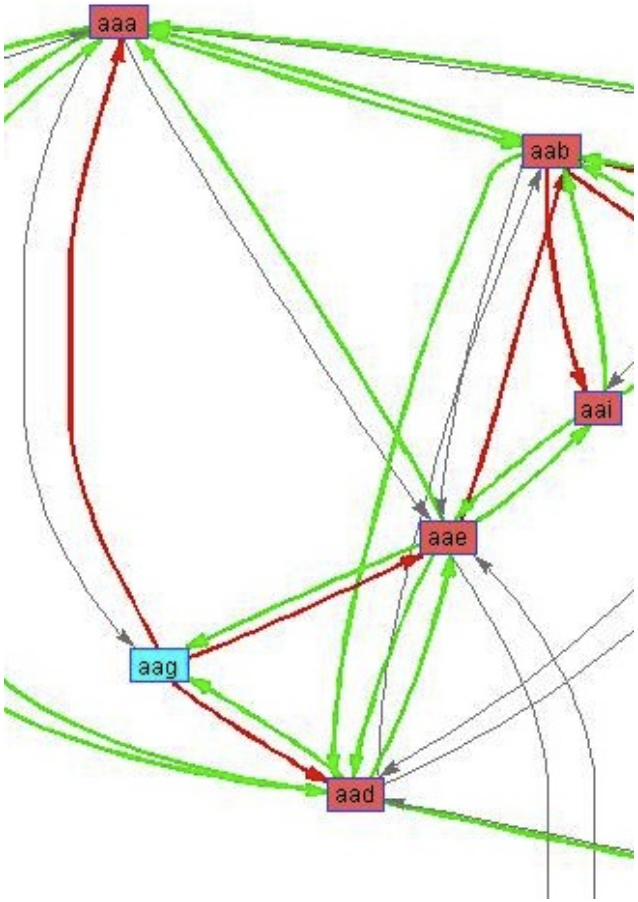
1. Do I know?
2. Who can I tell?
3. Choose a node.
4. Inform that node this timestep.
5. Repeat starting with "Do I know?"

The Doerr-Huber-Levavi method is very straight forward.



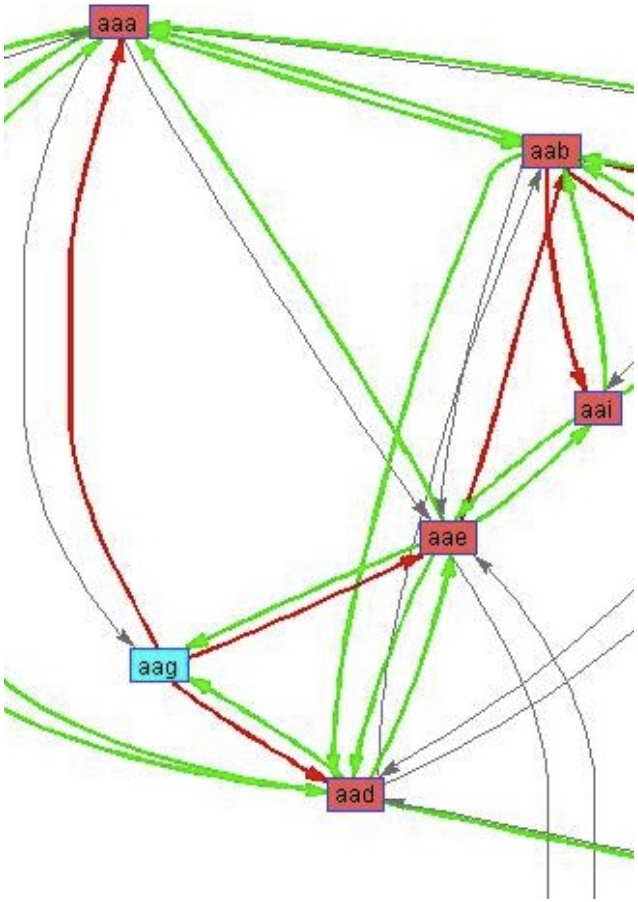
1. Do I know?
2. Who can I tell?
3. Choose a node.
4. Inform that node this timestep.
5. Repeat starting with "Do I know?"

The Doerr-Huber-Levavi method is very straight forward.



1. Do I know?
2. Who can I tell?
3. Choose a node.
4. Inform that node this timestep.
5. Repeat starting with "Do I know?"

The Doerr-Huber-Levavi method is very straight forward.



1. Do I know?
2. Who can I tell?
3. Choose a node.
4. Inform that node this timestep.
5. Repeat starting with "Do I know?"

Doerr-Huber-Levavi and other random and quasi-random schemes boil down to chance.

- Spread the rumor for n rounds, it might be done.
- Simulations and Probability show it **should** be done after $(1+\varepsilon)(\frac{1}{\log_2(1+p)}\log_2 n + \frac{1}{p}\ln n)$ rounds
- Probability of that occurring is $1 - n^{\frac{-p\varepsilon}{40}}$

Chances are it will be done, given you run it the suggested number of rounds on whatever network you're spreading on.

Doerr-Huber-Levavi and other random and quasi-random schemes boil down to chance.

- Spread the rumor for n rounds, it might be done.
- Simulations and Probability show it **should** be done after $(1+\varepsilon)(\frac{1}{\log_2(1+p)}\log_2 n + \frac{1}{p}\ln n)$ rounds
- Probability of that occurring is $1 - n^{\frac{-p\varepsilon}{40}}$

Chances are it will be done, given you run it the suggested number of rounds on whatever network you're spreading on.

So, how can you be sure? If there was a mechanism in place to track completeness, would you save overall computation, thereby completing faster?

Doerr-Huber-Levavi and other random and quasi-random schemes boil down to chance.

- Spread the rumor for n rounds, it might be done.
- Simulations and Probability show it **should** be done after $(1+\varepsilon)(\frac{1}{\log_2(1+p)}\log_2 n + \frac{1}{p}\ln n)$ rounds
- Probability of that occurring is $1 - n^{\frac{-p\varepsilon}{40}}$

Chances are it will be done, given you run it the suggested number of rounds on whatever network you're spreading on.

So, how can you be sure? If there was a mechanism in place to track completeness, would you save overall computation, thereby completing faster?

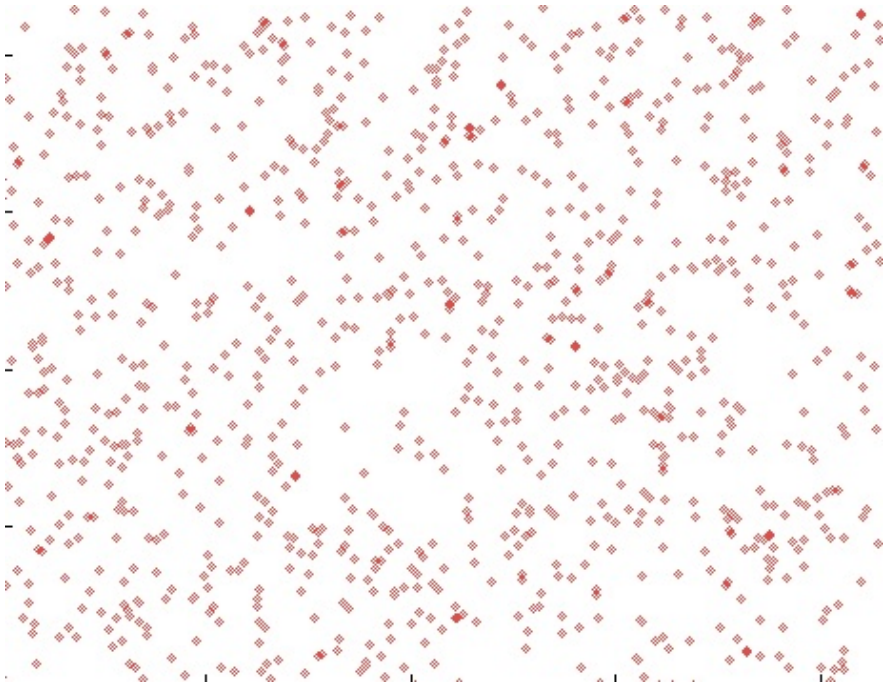
Too bad alterations to the scheme is beyond the scope of this talk.

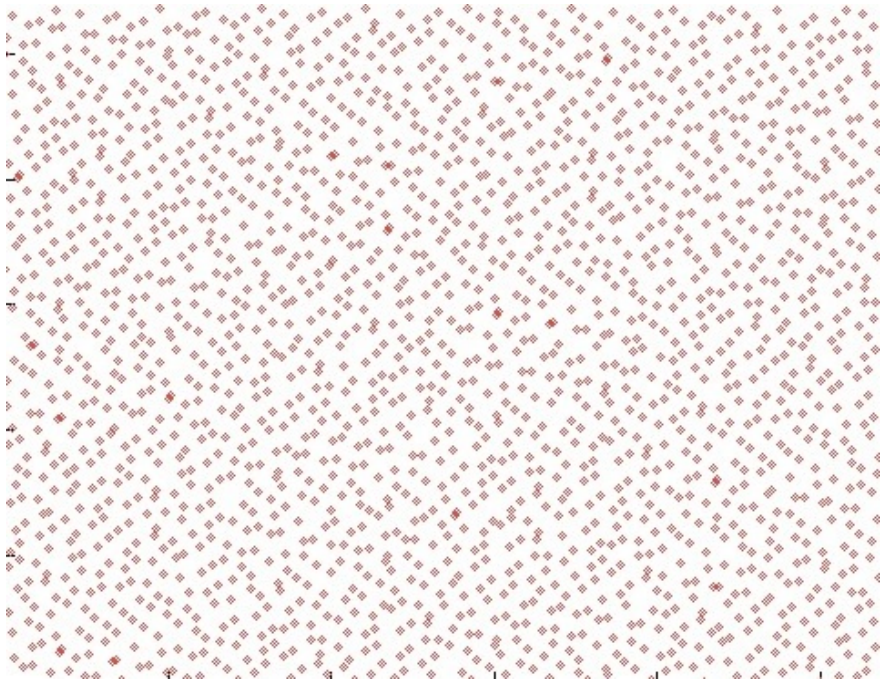
Psuedo-Random vs Quasi-Random

We want a random number generator to meet some basic criteria,

- Must reasonably represent a known probability distribution function.
- Do not want any built in trends, biases, or periodicities.
- Do not want the value generated at a given time to be correlated in any way with previous values.
- "Psuedo-Random" and "Quasi-Random" can be read as "Fake-Random" and "Sorta-Random".
- Mathematically,
  - Psuedo-random processes return numbers that appear random.
  - Quasi-random processes return numbers that are more uniformly distributed.
  - Or, Psuedo-random values will tend cluster, Quasi-random will not.

Psuedo-Random **vs** **Quasi-Random**

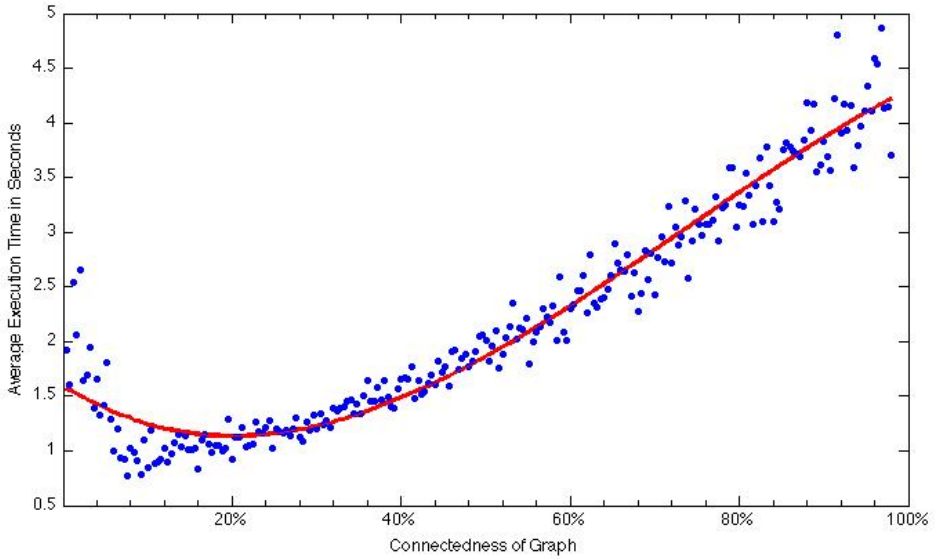




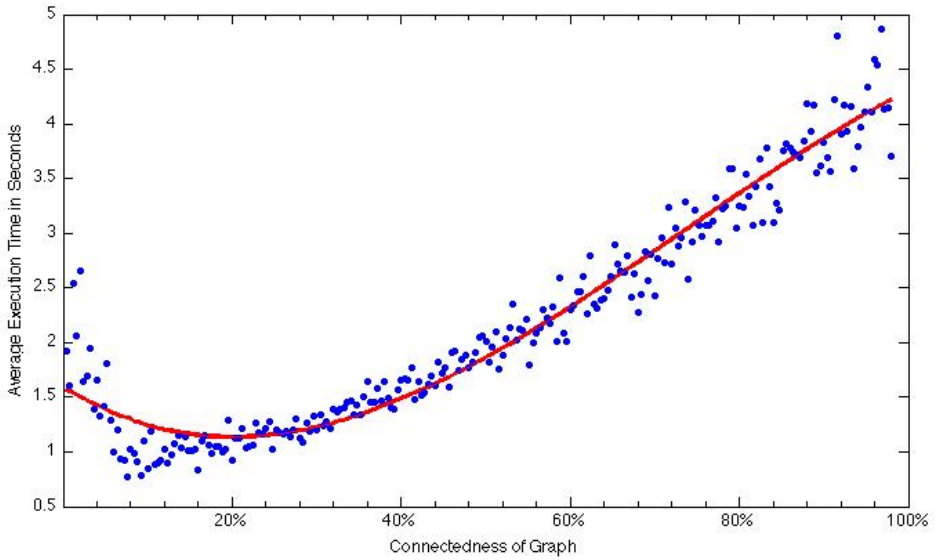
So, while both populated the domain randomly, the points generated with a quasi-random method appear to be more evenly distributed

The scheme for Quasi-Random rumor spreading is extremely wasteful for the sake robustness.

- For a fully connected graph, every round requires $\mathcal{O}(n^2)$ calculations, where n is the number of nodes.
- For a very sparse graph, every round requires $\mathcal{O}(n)$ calculations, where n is the number of nodes.
- There seems to be a point where total time in minimized based on the connectedness of the graph.



This graph represents the execution time vs the average number of connections per node. The minimal execution time can be explained by realizing a sparser graph will have less total computation per round ($\mathcal{O}(n)$), but takes on average a larger number of rounds to be fully informed.

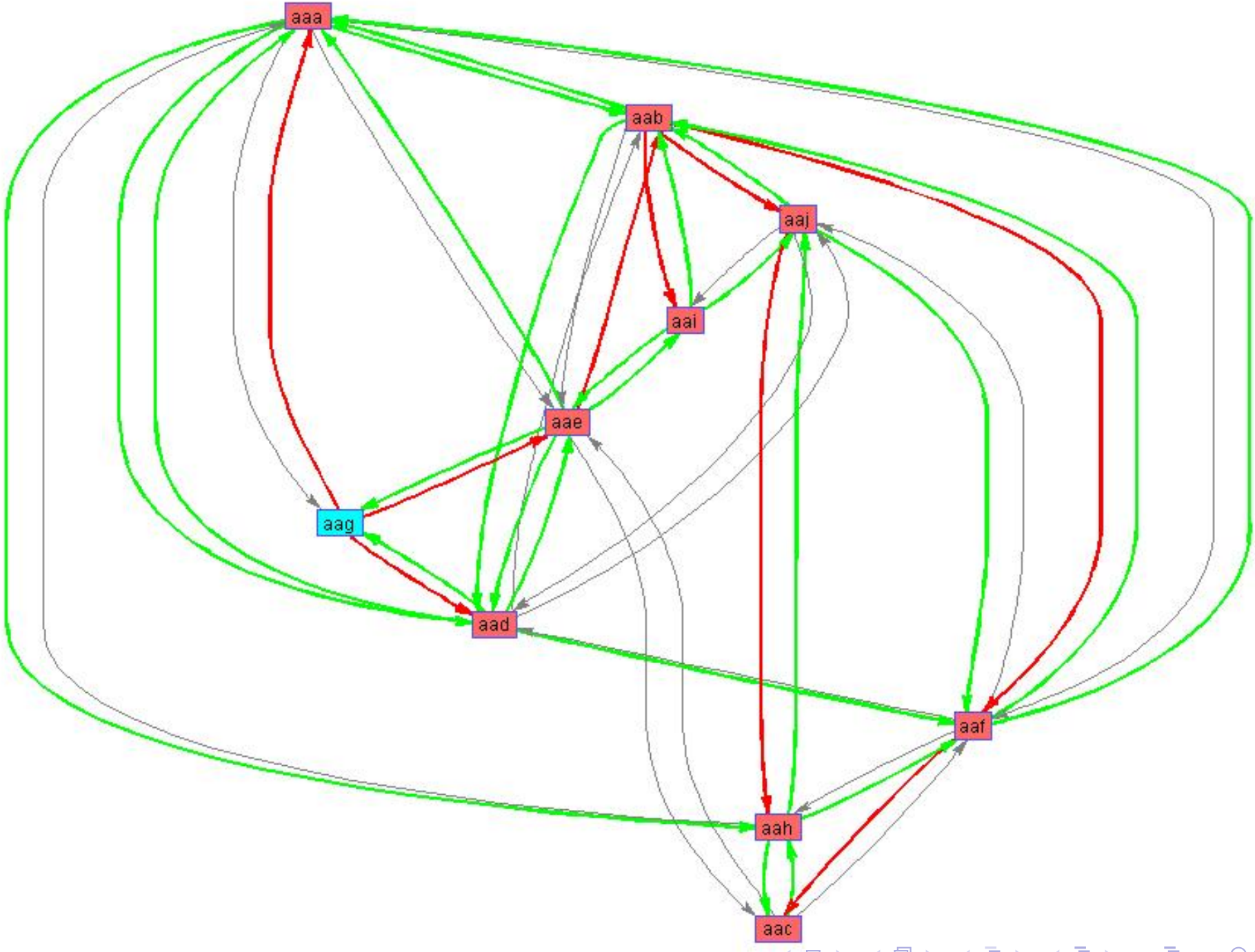


A fully connected graph will require much more computation per round($\mathcal{O}(n^2)$), but have overall fewer rounds.

Experimentally we have shown that there is an optimal total execution time.

- Exists where the number of edges between nodes is enough that there is a reasonable chance for each node to be informed
- Not so large that the number of steps per round approaches our upper bound of $\mathcal{O}(n^2)$.
- This optimal value is expressed in connectedness of a static graph.
- The value represents the average percentage of other nodes each node is connected to, and is around 12.8%.

This value was found by repeated simulation of graphs of increasing size. As total number of nodes increase the average converged to having approximately 12.8 percent of nodes connected to each node.


aaa
aab
aaj
aai
aae
aag
aad
aaf
aah
aac

Overall this algorithm for static graphs leads to much repetition.

- The idea is this will lead to robustness
- There is an optimum connection density for computation
- This has consequences for networks such as mesh networks
  - Instead of connecting to every nearby node, connect to optimum number (12.8% possible connections?)
  - Same with routing, spread packets to 12.8% of nodes, robustness can be shown.

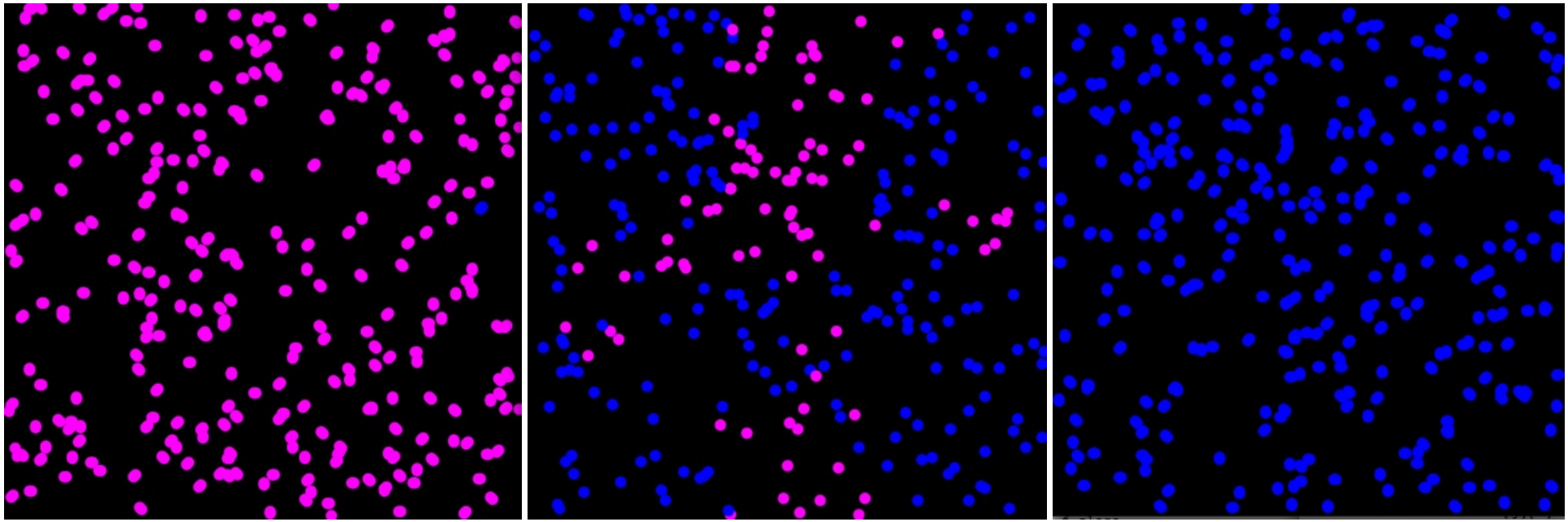
Overall this algorithm for static graphs leads to much repetition.

- The idea is this will lead to robustness
- There is an optimum connection density for computation
- This has consequences for networks such as mesh networks
  - Instead of connecting to every nearby node, connect to optimum number (12.8% possible connections?)
  - Same with routing, spread packets to 12.8% of nodes, robustness can be shown.

When we are using a graph with connectedness of 12.8% the average number of rounds to complete is at most 114% the minimum number of rounds. The minimum number of rounds comes with a fully connected graph, but this is also when we experience the most computation per round.

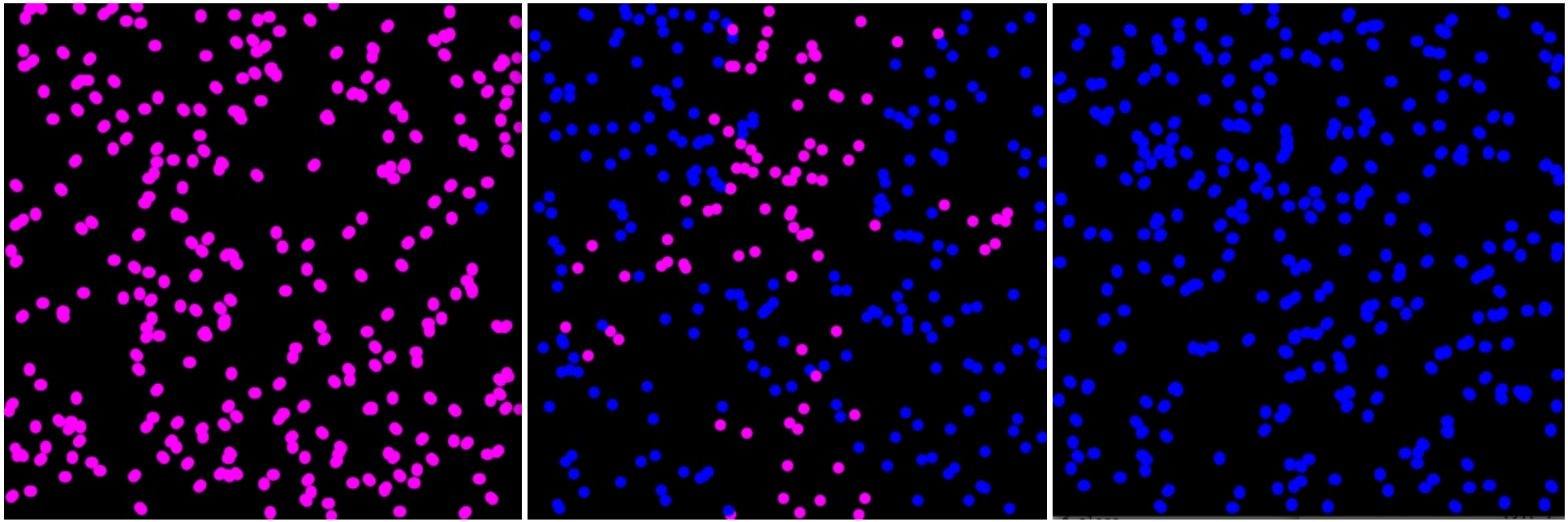
We used the term "Dynamic Topology" to describe rumor spreading amongst agents constrained to movement on a grid.

Beginning | Roughly Half Point | End



We used the term "Dynamic Topology" to describe rumor spreading amongst agents constrained to movement on a grid.

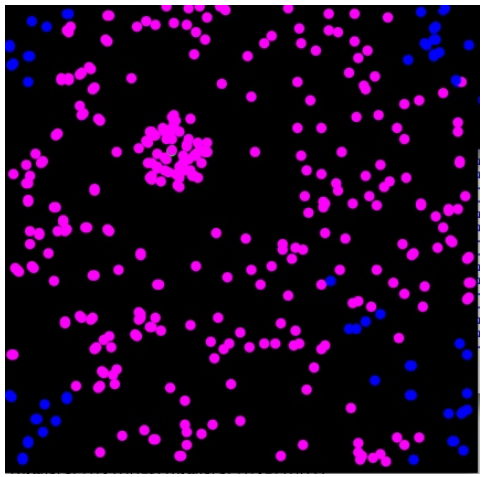
Beginning | Roughly Half Point | End



Video Examples - (Outside of this, LaTeX wouldn't compile)

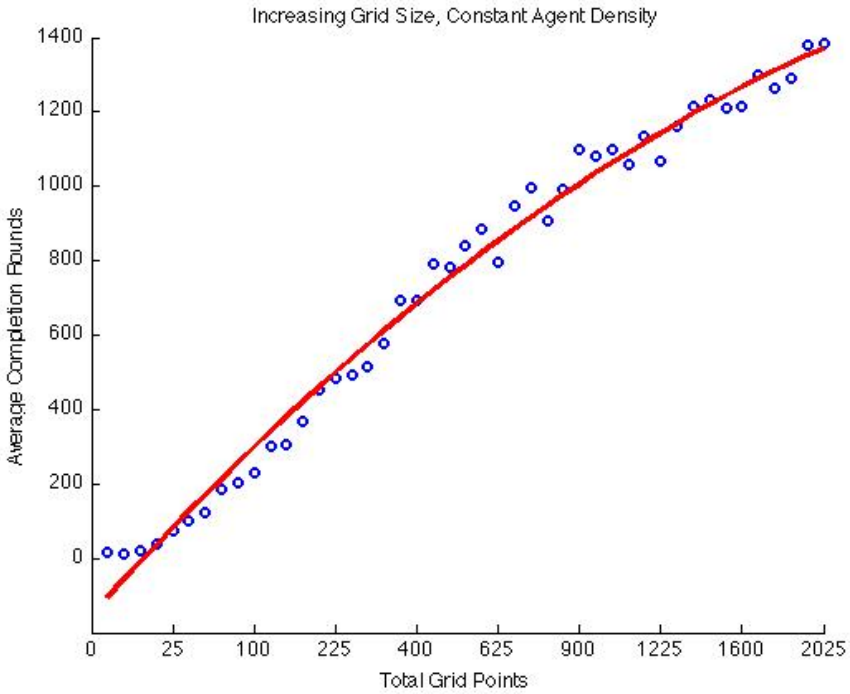
Rules for this simulation went along the lines of,

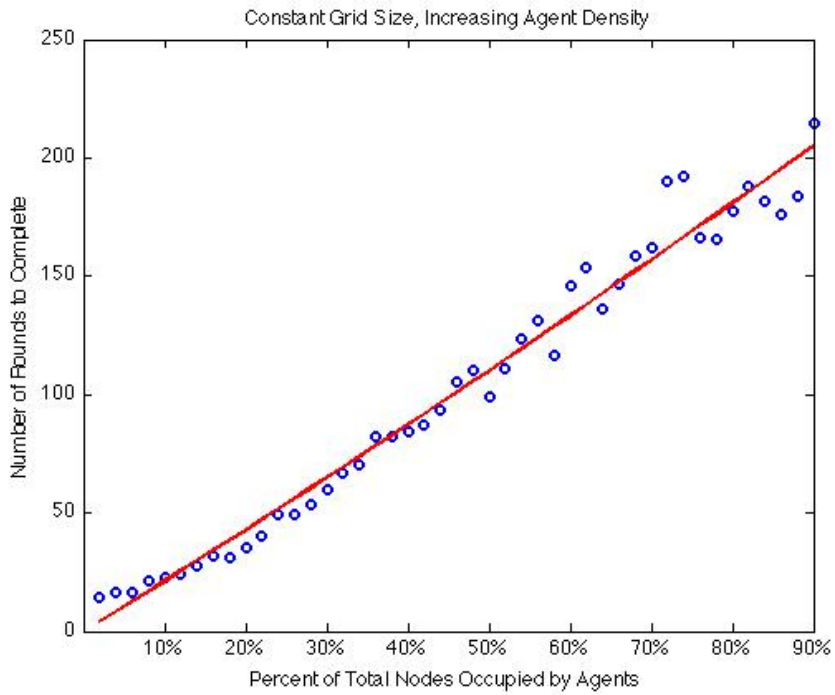
- Do I feel like moving?
  - Move one grid point in any of the allowed direction
  - Don't move
- If someone is near enough to hear me, shout at them.
  - Shout at everyone in range
  - They all hear the rumor



There were some functions included just for play, such as a clustering function.

There appears to be some interesting stuff going on here.





We've done all we were interested in numerically with the Static Topologies. We've barely touched our Dynamic Topologies.

- Static Topologies
  - Explore probabilities and proofs to confirm numerical results
- Dynamic Topologies
  - Show Robustness
  - Define probability of completion
  - Describe Robustness in terms of nodes and rounds