

DECENTRALIZED MESSAGE ROUTING IN MOBILE NETWORKS

JULY 2002

RYAN KENNEDY
UNIVERSITY OF NOTRE DAME

ABSTRACT

Decentralized systems have existed in nature for thousands of years. Each time you look at a flock of birds in the sky or at ants foraging for food, you witness a decentralized system. They rely on the concept of local agent to agent interaction; there is no global communication. This simple local communication has the capability to expand to many of our existing systems, such as in computer networks. Not only would it be more reliable, but it also has the potential to be more efficient. The future of decentralized systems in our society is promising.

This report describes using StarLogo to simulate some simple decentralized systems. The simulations simulate a message traversing through a system of agents and measure message saturation when the message has expired from the system. Individual agents can only receive and send the message for a user-controllable number of time steps. The results show that for either relatively high densities of agents or for a high number of time steps to retain the message, the message saturation is near 100%. This is exciting information when one considers that parallel uses of these simulations could mimic a network of cell phones or even mimic many tiny robots exploring a distant planet. The applications of using local interactions to perform global tasks are virtually endless. My work will show that simple agents obeying simple rules can lead to grand achievements.

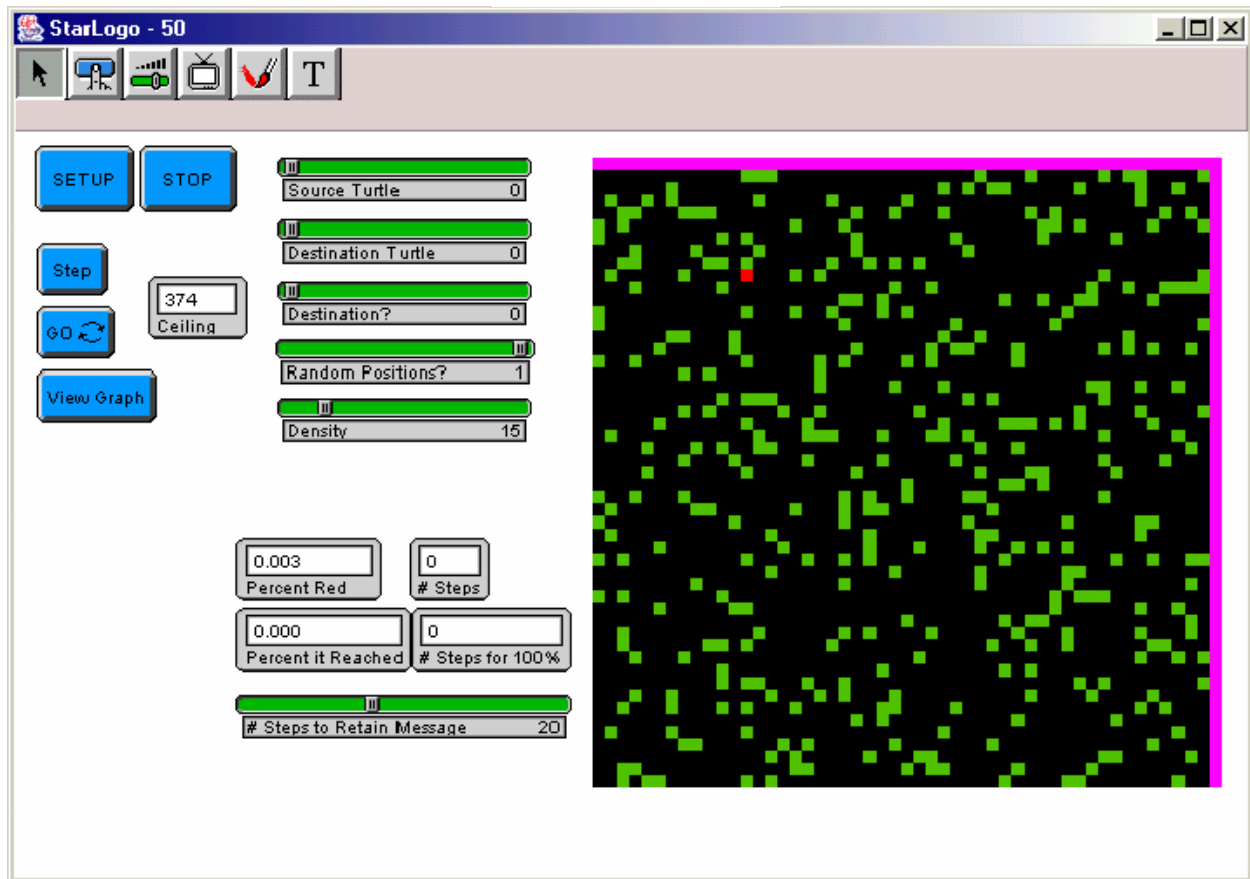
INTRODUCTION

The use of wireless communication has escalated in our society, and new technologies have been developed to meet this increasing demand. However, most of these existing networks rely on centralized knowledge routing. Decentralized message routing is a promising “new” technology that has the potential to be much faster and much more dependable. Without a fixed infrastructure, these systems offer more protection against shortages and system failure. If the proper steps are taken, decentralized message routing

has the capability to revolutionize the mobile computing industry.

Decentralized message routing has been slow to rise in mobile message routing because of a couple main concerns: how the message will traverse to its destination and how the individual nodes will communicate. Some work has been done in this area, but no decisive decisions have been made regarding the most efficient way to go. My strategy was to work alongside a graduate student doing research in this area and attempt to mimic some of his experiments in another simulation program while trying to improve and expand

Figure 1



the simulation when necessary. The remainder of this report details my work.

SIMULATION ENVIRONMENT

My simulations were run in StarLogo, which is a variant of the well-known Logo programming language. StarLogo is a very simple program that was designed for use in secondary education. It specializes in simulating decentralized systems by utilizing massive parallel execution. Because StarLogo simulations are easy to control and monitor, it served as a perfect environment for this purpose of this report. A general StarLogo window is shown in Figure 1.

In StarLogo, the agents are called turtles. Each turtle is assigned a number, starting at 0. These numbers go up to $n - 1$, where n

represents the number of turtles. The turtles move on a graphics canvas made up of patches. Each patch represents one grid square. Turtles may only exist on one patch at a time, and, for my simulation, only one turtle can exist on a patch at any given time step. The normal simulation environment is similar to a torus, but I have modified it so there are four distinct boundaries, forming an enclosed square environment. The turtles are free to move one grid step at each time step. A time step is defined as one attempt to pass the message and one attempt to move. Turtles only move if their new destination is not occupied by another turtle, but they will always attempt to share. The turtles were not programmed to be able to share patches because of the difficulties replicating this in Swarm, which is briefly described below. The initial density of turtles (total number of turtles per total number of patches) and the

number of time steps to remember the message are controllable from the program and the default grid size (in terms of patches) that the turtles move on is 50 x 50.

A graduate student here at the University of Notre Dame, Michael Kirkpatrick designed these simulations in Swarm. Swarm is a software tool developed at the Santa Fe Institute. It is a much more powerful tool than StarLogo, as its simulations are written in Java. The underlying concept of decentralization is present in Swarm as well.

SIMULATION METHODS

I simulated the system by varying the density of agents and the number of time steps before the agent's message expired on a 50 x 50 grid. Once the message expired from a turtle, it could not learn or pass the message again, but it continued to move. The densities used ranged from 3% coverage to 35% coverage and the time steps to retain the message ranged from 5 – 30. The initial distribution of the turtles was random and turtle # 0 was always the message source. This effectively randomized the message source's location. Turtles that knew the message were red, while turtles that didn't know the message were green. After each simulation, I recorded the percentage of agents the message reached (not including the source). A simulation stopped when the message expired from the system.

The simulation used fairly simple code (Appendix A) to operate, and there were few difficulties in developing it. A turtle would first attempt to share the message to all turtles exactly one patch away, specifically the 8 adjacent patches. Next, the turtle would move to a new patch exactly one patch away from

its previous position, namely to one of the 8 adjacent patches. A turtle would only move to a patch not occupied by another turtle. When a turtle was next to a wall, it was only able to move to the patches confined by the wall. The wall was one patch in width, so the turtles could not share the message or move through the walls. A turtle would only attempt to share the message and move once per time step.

The simulation was also tested with a smaller grid size. In addition to the initial size of 50 x 50, a 25 x 25 grid was used in the simulations. For these simulations, density was varied from 5% coverage to 25% coverage and the number of time steps to retain the message was set at 15. Everything else was kept the same as in the previous simulations. These parameters were used to minimize simulation time and to get the best median results possible.

RESULTS

The simulations show that as the number of time steps to retain the message was increased, the lower the density required to reach near 100% saturation. In addition, plotting the density versus the percentage of agents that the message reached, results in a graph with many "S" curves, shown in Figure 2. Each point on the graph in Figure 2 is the average percent of agents the message reached based on 20 runs on a Sun Blade 1000 machine running Solaris 8. These 64-bit UltraSparc computers run at 600 MHz with 512 MB of RAM. Unfortunately, error bars couldn't be added to the graph. The actual data is shown in Table 1. When the density reached a certain threshold for a given number of time steps to retain the message,

Figure 2

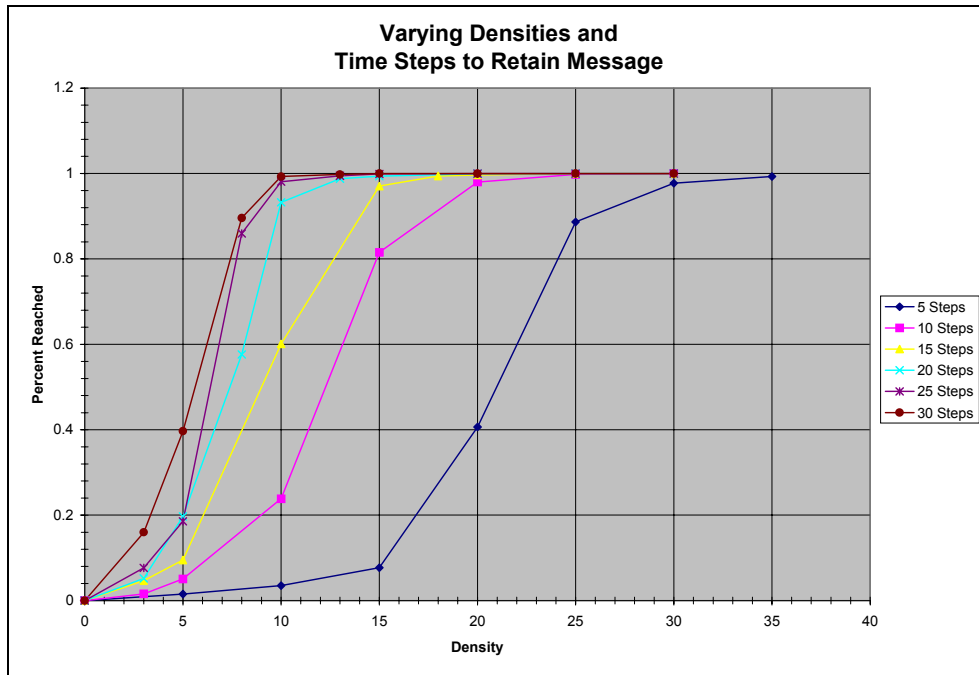
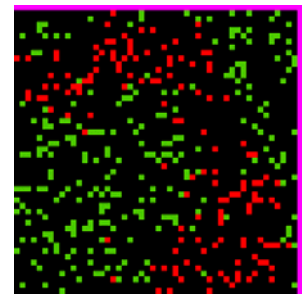


Table 1

Time Steps Message Retained

	5	10	15	20	25	30
3	---	0.015	0.047	0.052	0.076	0.159
5	0.015	0.051	0.095	0.197	0.186	0.397
8	---	---	---	0.577	0.86	0.9
10	0.035	0.239	0.601	0.932	0.981	0.993
13	---	---	---	0.988	0.995	0.998
15	0.077	0.815	0.97	0.994	0.999	1
18	---	---	0.994	---	---	---
20	0.406	0.98	0.996	0.999	1	0.999
25	0.887	0.998	0.999	0.999	1	1
30	0.977	0.999	0.999	1	1	1
35	0.993	---	---	---	---	---

Figure 3



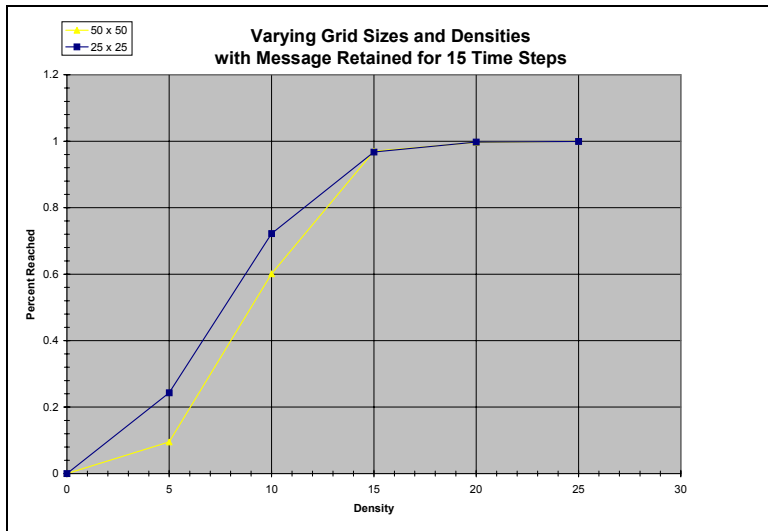
the percent of agents the message reached increased dramatically. On average, just below the threshold less than 20% of the agents received the message, while just above it more than 80% of the agents received the message before it expired. The slope of the threshold generally increased as the number of time steps to retain the message increased.

When testing densities ranging from 10% coverage to 20% coverage and a message

retention rate of 10 time steps are nearly identical to those obtained by Michael Kirkpatrick in his research. Unfortunately, the remainder of his results were not available at the time this was written, so more comparisons could not be done. Such comparisons of results are called docking.

When the grid size was decreased to 25 x 25, the percent of agents the message reached was slightly higher than an identical simulation

Figure 4



run on a 50 x 50 grid. No distinguishable difference in slope was noted for the different grid sizes, and the same “S” curves resulted. The graph and data are respectively shown in Figure 4 and Table 2. Again, there are no error bars on the graph, but each data point represents the average of 20 runs.

DISCUSSION AND CONCLUSIONS

My results show that the only condition that needs to be satisfied to reach near 100% saturation is either a relatively high time step to retain the message or a relatively high initial density. My results support that near 100% saturation will be achieved if the message is retained for 30 time steps with a density of only 8%. Similarly, near 100% saturation is achieved if the message is retained for only 5 time steps where the density is 25%. Both the number of time steps to retain the message and the density have critical roles in the simulation because they affect how many times the message is potentially shared. As density increases, the likelihood that sharing is possible increases.

Table 2

		Grid Size	
		25 x 25	50 x 50
Density	5	0.243	0.095
	10	0.722	0.601
	15	0.967	0.97
	20	0.998	0.997
	25	0.999	0.999

Additionally, as the time steps to retain the message increases, the likelihood that the agents will have a message to share (when they can share) increases.

My results for the 25 x 25 grid are inconclusive. There is not enough variance in my data (or enough data) to determine if additional relationships are present. The 25 x 25 grid did have a slightly higher saturation than the 50 x 50 grid for most densities. A possible explanation for this is that the percentage of agents reached may be dependent on the relationship between the number of time steps the message is retained and grid size. However, my simulations yield nothing to determine whether this is true.

FUTURE WORK

In future studies, an exploration of simulations with more data points would be beneficial. In my simulations, I did not test more data points because of a time constraint. In addition, simulations with larger grid sizes would be appropriate. Again, these were omitted from my study because as the grid

size increases, the time it takes to run a simulation also greatly increases. This new data could allow us to make conclusions as to whether grid size does affect the percentage of agents reached, and, more particularly, if the number of time steps to retain the message and receive near 100% saturation is related to the grid size. Such information would also be beneficial to the field for the purpose of docking.

A randomly positioned destination agent added to the simulation could replicate a message that has a specific destination. Here, saturation wouldn't matter as long as the destination agent received the message. Perhaps the destination agent could send back an acknowledgement to the source. Specific agent to agent message sending is one of the most important aspects of this growing research field.

In many of the simulations with a low density or low time steps to retain the message, the message often expired before it could be passed. A possible fix, similar to one noted by Haas, Halpern, and Li, would involve the message not expiring from the source until it is passed at least once. This would allow the message spreading to get started in the system and potentially increase saturation.

Another interesting simulation would involve agents spreading the message based on an user-controllable probability or spreading the message every i^{th} time, also controllable by the user. This would likely lead to fewer agents carrying the message in the system at any given time and would likely yield saturation values very similar to my results. It would be important to compare the number of time steps for the message to reach a certain

saturation or to expire and then to compare those results to the results from this report.

Finally, it would be valuable to explore the use of super agents and irregularly shaped simulation environments, as suggested by Michael Kirkpatrick. Perhaps super agents could move and share twice in a time step, while normal agents could only do each once. Or maybe they could pass the message to a greater radius, not just their 8 adjacent neighbors. This would likely lead to interesting results. Having an irregularly shaped grid would also be a beneficial expansion of the simulations tested in this report, as the environments of real world applications are often not well-defined.

The possibilities for further research in this area are boundless. The simple concept of patches described in this report could be used as a metaphor to represent transmission ranges or numerous other mobile networking ideas. In addition, the idea of a rumor being passed here could be thought of as a disease, among other things. Specifics to the simulations performed here could be easily manipulated and new variables added. Most simply, one could record the time steps or time required both for the message to reach all agents and for the message to expire. The future of decentralized mobile disconnected networks shows great potential.

REFERENCES

- Haas, Halpern, and Li. "Gossip-Based Ad Hoc Routing." Cornell University, 2002.
- Kirkpatrick, Michael. "Using Swarm Intelligence to Route Messages in Disconnected Ad Hoc Mobile Networks." University of Notre Dame, 2002.

Madey, Gregory. University of Notre Dame.

(Advisor)

StarLogo - <http://www.media.mit.edu/starlogo>

Swarm - <http://www.swarm.org/>

Appendix A

Java StarLogo
1.2

****Graphing Capabilities are not included in this version
*Contact Ryan Kennedy at kennedy.76@nd.edu for other versions***

or questions

`turtle`

```
turtles-own [dir steps rec? rch]
globals [time once count cnt]
```

```
to bump ;procedure that makes sure the turtles begin on a patch by themselves
if (count-turtles-here > 1) or (pc = 125)
  [setxy random screen-width random screen-height bump]
end
```

```
to share ;procedure that shares the message
if color = red [
if (rec?-at 0 1) = 1 [setc-at 0 1 red setrch-at 0 1 1] ;check to see if
agent can
if (rec?-at 0 -1) = 1 [setc-at 0 -1 red setrch-at 0 -1 1] ;receive message
if (rec?-at 1 0) = 1 [setc-at 1 0 red setrch-at 1 0 1] ;if it can, change
its
if (rec?-at -1 0) = 1 [setc-at -1 0 red setrch-at -1 0 1] ;color to red and
set
if (rec?-at -1 -1) = 1 [setc-at -1 -1 red setrch-at -1 -1 1] ;its reached
variable to 1
if (rec?-at -1 1) = 1 [setc-at -1 1 red setrch-at -1 1 1]
if (rec?-at 1 1) = 1 [setc-at 1 1 red setrch-at 1 1 1]
if (rec?-at 1 -1) = 1 [setc-at 1 -1 red setrch-at 1 -1 1]
if who != src [setrch 1]]
end
```

```
to set-dir ;randomly sets the turtle's direction
seth pick [0 45 90 135 180 225 270 315]
if ((pc-at dx dy) = 125) [set-dir]
end
```

```
to move ;move procedure (only executed if destination patch is empty)
set-dir
ifelse (heading = 45) or (heading = 135) or (heading = 225) or (heading = 315) [
  if ((count-turtles-at dx dy) = 0) [fd (sqrt 2) setxy (round xcor) (round
ycor)]] [
  if ((count-turtles-at dx dy) = 0) [fd 1]]
if color = red [setsteps steps + 1]
end
```

```
to update
if steps >= buf [setc green setsteps 0]
if color = red [setrec? 0] ;don't receive message again
end
```

`observer`

```
to setup
settime 0
setonce 1
setcount 0
ct
crt ((screen-width - 1) * (screen-height - 1)) * (density / 100)
if rand = 0 [set-random-seed 4444]
ask-turtles [setc green setxy random screen-width random screen-height bump]
```

