Agent-Based Tools for Modeling and Simulation of Self-Organization in Peer-to-Peer, Ad Hoc, and Other Complex Networks

Muaz Niazi and Amir Hussain, University of Stirling

ABSTRACT

Agent-based modeling and simulation tools provide a mature platform for development of complex simulations. They however, have not been applied much in the domain of mainstream modeling and simulation of computer networks. In this article, we evaluate how and if these tools can offer any value-addition in the modeling & simulation of complex networks such as pervasive computing, large-scale peer-to-peer systems, and networks involving considerable environment and human/animal/habitat interaction. Specifically, we demonstrate the effectiveness of NetLogo — a tool that has been widely used in the area of agent-based social simulation.

INTRODUCTION

The last decade has seen unprecedented growth in the scale and complexity of computer networks; networks today are much larger and far more complex than originally anticipated. In turn, the larger and more complex the network, the harder it becomes to test and evaluate its design and eventual implementation. Modeling and simulation (M&S) serves a vital role in the design and development of distributed interacting systems because of their peculiar stochastic nature, especially if they involve systems with decentralized self-organizing capabilities [1, 2]. Although networks have grown in both size as well as complexity, the tools to model and simulate them have not scaled at the same rate. This has not gone entirely unnoticed, so papers such as [3] have attempted to point out problems with the current simulation methodologies for telecommunications networks, such as the use of pseudo-random number generators. However, another problem with the existing toolset, which has not received much attention, lies in what initially was their strongest point: their focused orientation on computer networks. Since newer networks have grown in size and complexity in directions such as pervasive computing (including but not limited to body area networks, vehicular area networks, and other pervasive networks), the existing M&S tools are not quite designed to deal with these emerging paradigms. With so much unanticipated change in the air, there is a need for alternative flexible tools equipped with effective techniques for modeling modern networks as they evolve into complex systems. This article seeks to address this important area of research for the M&S community in the domain of computer networks by demonstrating, for the first time, the use of agent-based modeling tools for modeling self-organizing mobile nodes and peer-to-peer (P2P) networks. We focus on NetLogo, a tool that has proven its worth in the M&S domain of complex systems. We conclude that not only can agent-based tools model P2P and ad hoc networks, they can also be used to effectively model complex systems where the network is only a part of the system, and where humans, animals, habitat, and other environment-related interactions need to be modeled and simulated as well.

BACKGROUND

In the domain of complex networks, being the new kid on the block, pervasive computing applications do not have many dedicated tools at their disposal, unlike the P2P and wireless network M&S communities. These tools range from general-purpose tools such as Opnet, OMNET++, and NS 2 to customized tools for particular domains such as Tiny OS Simulator (TOSSIM) for wireless sensor networks. Designed from the ground up for modeling computer networks, these specialized tools satisfy the basic requirements of M&S. However, the main focus of these tools is to model and simulate computer networks only. On one hand, we have complex problems where humans and mobile systems/users are involved, and on the other, we have pervasive computing and large-

0163-6804/09/$25.00 © 2009 IEEE

scale global networks on an Internet scale. Pervasive computing, being a set of immersive technologies, always has a need for tools with stronger modeling capabilities. The modeler needs the ability to work at a higher level of abstraction instead of just being able to tweak the network parameters. Tools are needed that are flexible enough to conveniently model and simulate complex interactions and protocols with ease. Such tools need to have inherent strengths, and should have demonstrated abilities to model self-organization techniques, engineered emergence, and other socio- and bio-inspired techniques including domains such as ambient intelligence [4].

Another area of application of such tools and techniques is large networks such as unstructured P2P systems, which tend to grow in unpredicted dimensions. A recent example is Gnutella, where strong emergence and unexpected/unexplored graphs have been discovered quite recently.[1] Although in these domains tools primarily designed for modeling of computer networks can occasionally be used to model and simulate, the norm is to have various levels of add-ons; this, however, is not exactly an easy marriage. The designers, being human, feel over-constrained by the limitations of having to keep tweaking and tracking low-level network parameters when their primary focus is on modeling more abstract concepts. It is important to note that the problem emerges because these concepts are natural for the designer but alien for the tool; modern complex network M&S specialists not only require the ability to change sensor ranges and other basic parameters, they also need abilities to model and simulate a myriad of entities and techniques: humans, mobile robots, animals, habitat interaction, as well as self-* techniques (self-organization, self-healing, self-adaptive, etc.). This could eventually be extended to any conceivable concept that could be expected to form a part of the system. So in this domain, even if existing network oriented tools can somehow be used to model these concepts, the concept is typically an add-on and an afterthought to the design. Tools that are not specifically developed to cater for complex simulations and situations make the M&S specialist work longer to fulfill these requirements.

In contrast to this, in another part of the M&S landscape, there are a set of tools focusing on agent-based M&S techniques. These are a mature set of tools whose utility has been demonstrated in various domains. They are used primarily for M&S of complex systems' simulation models ranging from social networks and groups to self-organizing systems and bio-inspired models.

The important question we address in this article is the following: Is there any value addition to the use of generic and highly flexible agent-based modeling tools (which are already known to be very effective in modeling self-organization and complex systems) in the modeling and simulation of communication networks such as P2P, wireless, and pervasive networks?

The rest of the article is organized as follows. The next section briefly reviews the background and distinction of agent-based M&S tools. We then introduce NetLogo, and evaluate its usefulness using a number of M&S experiments. Finally, some concluding remarks are given.

Review of Agent-Based Tools

Are There Any Existing Tools?

Before we start our detailed analysis on the use of NetLogo in this particular domain, it is pertinent to mention some of the previous tools in use by network designers. The literature mentions a large set of tools, which range anywhere from customized scripts such as GnutellaSim,[2] which runs on general purpose tools (e.g., NS-2), to packet-level discrete event simulators for simulation of similar networks as in Structella.

Agent-Based Tools

In the agent-based arena, a number of popular tools are available. These range from Java-based tools such as Mason to Logo-based tools such as StarLogo and NetLogo [5]. Each of these tools has different strengths and weaknesses. In the rest of the article we focus on just one of these tools, NetLogo, as a representative of this set. We attempt to critically evaluate NetLogo and see how it fares when compared to other standard tools that are already in use for M&S of communication networks. Building on the experience of previous tools based on the Logo language, NetLogo has been developed from grounds up for complex systems research. Historically, NetLogo has been used for modeling and simulation of complex systems, including multi-agent systems, social simulations, and biological systems, on account of its ability to model problems based on human abstractions rather than purely technical aspects. However, it has not been widely used to model computer networks, especially by any mainstream practitioners in the computer network M&S domain.

Agent-Based Thinking — In a distributed environment with large-scale complex interacting entities, sometimes it makes more sense to work on a simulation based on local parameters of each network node and assuming intelligence at a local level. Now, for the network modeling specialist to model a system in terms of agents, the nodes, humans or robots or any other entities that might be part of the model, need to be directly addressable. The designer of the system should be free to address one or more types or breed of objects/entities/nodes of the system. This kind of addressability is the norm in agent-based modeling and simulation.

As an example, it is fairly easy to address one or more types of entities (called agents or turtles in Logo-speak) just by their name. In other words, programs in NetLogo[3] are produced at a higher level of abstraction. Instead of the regular paradigms of looping through objects and executing functions, the designer can ask entities/nodes to move or change colors, or create links with other entities without worrying about the low-level details of how the animation or actual interaction is going to be executed. This paradigm actually produces small yet functional programs. An important benefit of small pro-

In a distributed environment with large scale complex interacting entities, sometimes it makes more sense to work on a simulation based on local parameters of each network node and assuming intelligence at a local level.

[1] *http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/topology.html*

[2] *http://www.cc.gatech.edu/computing/compass/gnutella/usage.html*

[3] *NetLogo is based on the Logo language.*

In NetLogo, modeling complex protocols does not have to be limited to the simulation of networks alone; it can readily be used to model human users, intelligent agents, mobile robots interacting with the system or virtually any concept that the M&S designer feels worthwhile having in the model.

grams is their ability to greatly reduce the tweak-test-analyze cycle. This age-old paradigm of development (coming from the heritage of interpreted languages) where paper-based models are assisted by quick remodeling of the model to fit the given parameters for verification and validation is very obvious in NetLogo. Thus, it is more likely to model complex paradigms within a short time without worrying about the lower layers of other parameters unless they are important for the particular application. As we shall see in this article, it also has great expressive power and, most of all, is fun to use. Although used very frequently to model complex and self-organizing systems, it has not previously been used extensively to model computer networks.

Distinction of Agent-Based M&S Tools — In this section we distinguish the power of agent-based techniques as compared to previous tools in general and NetLogo in particular. There are certain metrics by which we can evaluate agent-based tools and compare them with their counterparts in the domain of computer network M&S. These metrics, however, are of a qualitative nature. The existing tools for network-based M&S are not specifically tailored to developing self-organizing or complex system simulations. The M&S tools from the agent-based M&S domain have several strong features such as direct addressability of nodes, ease of implementation and evaluation of self-organization, and emergence and bio-inspired algorithms as well as the capability of being understandable from the human perspective (having their background rooted in social simulation), all of which make them extremely useful for application in the domain of ad hoc, P2P, and pervasive systems. One way of looking at this is that originally, the object oriented programming features of the C++ programming language were developed entirely in C in the form of a front-end that would compile to C code (as an academic exercise); however, it was too cumbersome and counterproductive for programmers to follow the same practice in their regular development. It was infeasible for companies to develop programs with object-oriented features using pure C in general. However, with the advent of modern languages with built-in features of object orientation, it has become much easier and "natural" for software engineers to start with a template based on an object-oriented paradigm. Using the same analogy, self-organization and emergence techniques are even more abstract than object orientation, but they do entice and fit well with human nature.[4] So the significance of agent-based tools is to attain a level of comfort for the M&S specialist in designing complex paradigms such as self-organization.

Complex Interaction Protocols Modeling — Another point to note here is that in NetLogo, modeling complex protocols does not have to be limited to the simulation of networks alone; it can readily be used to model human users, intelligent agents, mobile robots interacting with the system, or virtually any concept the M&S designer feels worthwhile having in the model. NetLogo in particular has the advantage of LISP-like list processing features (with LISP being one of the most commonly used languages in the AI literature). Thus, modeled entities can be shown to be interacting with the computer networks all at the same time and with the same ease that is there for modeling networks. Alternatively, the simulationist can interact and create runtime agents to interact with the network to experiment with complex protocols that are not otherwise straightforward to conceive in terms of programs.

As an example, let us suppose that we were to model the number of human network managers (e.g., from 10 to 100) attempting to manage a network of 10,000 nodes by working on workgroups the size of n nodes (e.g., ranging from 5 to 100) at one time while giving a total of 8-hour shifts with network attacks coming in as a Poisson distribution; this could all probably be modeled in less than a few hours with only a little experience in NetLogo per se. The simulation can then be used to evaluate policies of shifts to minimize network attacks.

Another example could be the modeling and simulation of link backup policies in case of communication link failures in a complex network of 10,000 nodes along with network management policies based on part-time network managers carrying mobile phones for notification and management vs. full-time network managers working in shifts, all in one simulation model. And to really make things interesting, we could try these out in reality by connecting the NetLogo model to an actual J2ME-based application in phones using a Java extension; so the J2ME device sends updates using General Packet Radio Service (GPRS) to a Web server that is polled by the NetLogo program to get updates while the simulation is updated in a user interface provided by NetLogo. Again, although the same could be done by a team of developers in a man-year or so of effort using different technologies, NetLogo provides for coding these almost right out of the box, and the learning curve is not steep either.

This is the expressive power of NetLogo, which lies in the sense of modeling even non-network concepts such as pervasive computing where human mobile users (e.g., in the formation of ad hoc networks for location of injured humans) or body area networks come in play along with the network. Now, it is important to note here that simulation would have been incomplete without effective modeling of all related concepts that come into play. Depending on the application, these could vary from ambulances, doctors, and nurses to concepts such as laptops and injured humans. in addition to readily available connectivity to GIS data provided by NetLogo extensions.

Range of Input Values — Being a general-purpose tool, the abstraction level of NetLogo is specifically much higher. As such, the concepts of nodes, antenna patterns, and propagation modeling are all user-dependent. On one hand, this may look burdensome to the user accustomed to using these on a regular basis, as it might appear that he or she will be working a little extra to code these in NetLogo modeling. On

[4] *As is clear from their prevalence in social simulation literature.*

the other hand, NetLogo allows for the creation of completely new paradigms of network modeling, wherein the M&S specialist can focus on, for example, purely self-organization aspects or developing antenna patterns and propagation modeling — directly in NetLogo, a relatively trivial task per se.

Range of Statistics — NetLogo is quite flexible in terms of statistics and measurements. Any variable of interest can be added as a global variable, and statistics can be generated based on a single run or multiple runs. Plots can be automatically generated for these variables as well.

Handling Complex Metrics — Measurements of complex terms in NetLogo programs are limited only by the imagination of the M&S specialist. Almost any concept that can be conceived as important can easily be added to the model. As an example, if it is required to have complex statistics such as network assembly time, global counters can be used easily for this. Similarly, statistics such as network throughput, network configuration time, and throughput delay can easily be modeled by means of similar counters (which need not be integral). By default, NetLogo provides for real-time analysis. Variables or reporters (functions that return values) can be used to measure real-time parameters, and the analyst can actually have an interactive session with the modeled system without modifying the code using the command window.

NetLogo Tutorial

In this section we introduce NetLogo, and demonstrate its usefulness using a number of modeling and simulation experiments.

What NetLogo Is and How to Get It

NetLogo is a popular tool based on the Logo language with a strong user base and an active publicly supported mailing list. It provides visual simulation and is freely available for download [5], and has been used considerably in multiagent systems literature [6]. It has also been used considerably in social simulation and complex adaptive networks [7]. One thing that distinguishes NetLogo from other tools is its very strong user support community. You can usually get a reply from someone in the community in less than a day. The current version of NetLogo is 4.0.4; the higher number actually demonstrates its stability and active development. NetLogo also contains an enormous number of code samples and examples. Most of the time, it is rare to find a problem for which there is no similar sample freely available either within NetLogo's model library or elsewhere from the NetLogo M&S community.

Basic Structure of a NetLogo Program

The NetLogo World — Based on the Logo language, the NetLogo world consists of an interface that is made up of "patches." The inhabitants of this world can range from turtles to links. In addition, one can have user interface elements such as buttons, sliders, monitors, and plots.

The NetLogo Interface — NetLogo is a visual tool and is extremely suitable for interactive simulations. When one first opens up a NetLogo screen, an interface with a black screen is visible. There are three main tabs and a box called the command center. Briefly, the interface tab is used to work on the user interface manually, and theinformation tab is used to write the documentation for the model. Finally, the procedures tab is where the code is actually written. The command center is a sort of interactive tool for working directly with the simulation. It can be used for debugging as well as trying out commands similar to the interpreter model which, if successful, can be incorporated in one's program. The commands of a NetLogo procedure can be executed in the following main contexts.

Turtle — The key inhabitants of the Logo world are the turtles, which, from our perspective of designing networks, can be used to easily model network nodes. The concept of agents/turtles is to provide a layer of abstraction similar to the layer of abstraction object-oriented programming adds to structured programming paradigms. In short, the simulation can address much more complex paradigms, including pervasive models, environment or terrain models, or indeed any model the M&S specialist can conceive of, without requiring many additional add-on modules. However, the tool is extensible and can be directly connected to Java-based modules. By writing modules using Java, the tool can potentially be used as the front-end of a real-time monitoring or interacting simulation. For example, we could have a Java-based distributed file synchronization systemthat reports results to the NetLogo interface and vice versa; the NetLogo interface could be used by the user to set up the simulation at the back-end (e.g., how many machines, how many files to synchronize), and subsequently, with the help of the simulation, the user could simply monitor the results. Although the same can be done with a lot of other tools and technologies, the difference is that NetLogo offers these facilities almost out of the box and requires minimal coding besides being noncommercial, free, and easy to install.

Patch — A single place where the turtle exists is a patch.

Observer — This is a context that can be used in general without relating to either a patch or a turtle. The NetLogo user manual, which comes prepackaged with NetLogo, says: "Only the observer can ask all turtles or all patches. This prevents you from inadvertently having all turtles ask all turtles or all patches ask all patches, which is a common mistake to make if you're not careful about which agents will run the code you are writing."

Explanation of NetLogo Nomenclature — Inside the NetLogo world, we have the concept of agents. Coming from the domain of complex systems, all agents inside the world can be addressed in any conceivable way the user can think of; for example, if we want to change the color of all nodes with communication power

NetLogo is quite flexible in terms of statistics and measurements. Any variable of interest can be added as a global variable and statistics can be generated based on single or multiple-runs. Plots can be automatically generated for these variables as well.

(a)

```
1.  to setup
2.  ca ; Clears everything so if we call setup again, it won't make a mess
3.  crt 100 ;This means we are creating a 100 turtles
4.  [
5.  setxy random-pxcor random-pycor    ;These 100 turtles, we want them to be spaced out at random
6.  ; patch x and y co-ordinates
7.  ]
8.  let mycolor random 140 ; Randomly select a color value from 0 to 139
9.  ask patches
10. [
11. set pcolor mycolor ; Ask all patches to set their color to this random color
12. ]
13. end
```

(b)

```
1.  to go
2.  ask turtles
3.  [
4.  fd 0.001 ; ask each turtle to move a small step
5.  ]
6.  end
```

Figure 1. *a) Code for setup; b) code for go.*

less than 0.5 W, a user can simply say: ask nodes with [power < 0.5] [set color green]; or if a user wants to check nodes with two link neighbors only, this can also be done easily, and so on.

The context of each command is one of three. The observer object is the context when the context is neither turtle nor the patch. It is called the observer because this can be used in an interactive simulation where the simulation user can interact in this or another context using the command window.

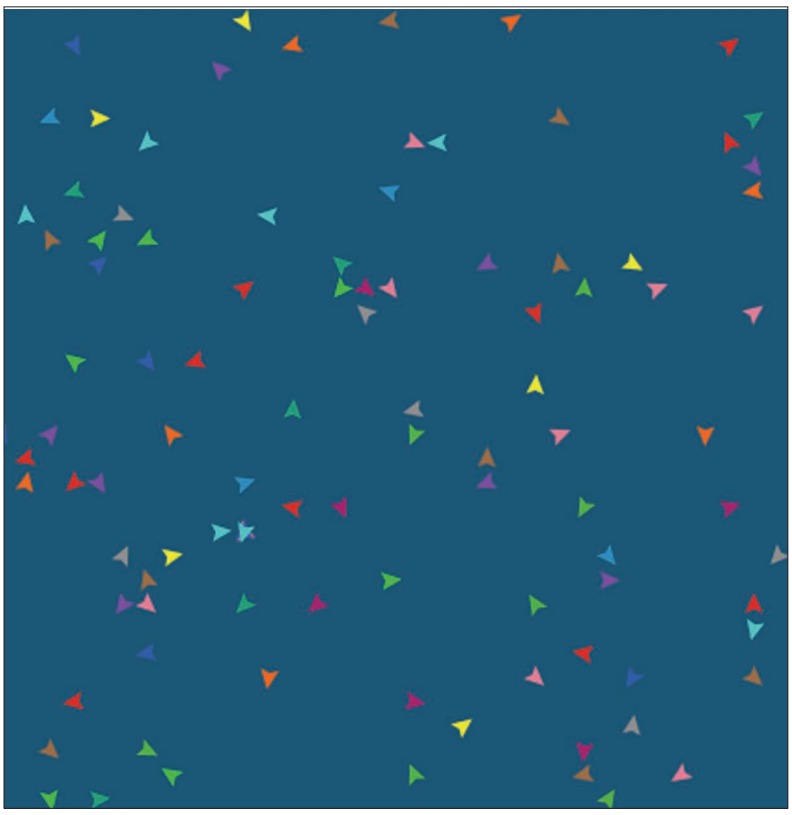

Figure 2. *Mobile nodes.*

The Setup and Go Buttons — Although there are no real rules to creating a NetLogo program, one could design a program to have a set of procedures that can be called directly from the command center. However, in most cases it suffices to have user interface buttons to call procedures. For the sake of this article, we shall use the standard technique of buttons.

In our program, to start with, we shall have two key buttons: setup and go. The idea is that the setup button is called only once, and the go button is called multiple times (automatically). These can be inserted easily by right clicking anywhere on the interface screen and selecting buttons. So just to start with NetLogo, the user will need to insert these two buttons in his or her model, remembering to write the names of the buttons in the commands. For the go, we shall make it a forever button. A forever button is a button that calls the code behind itself repeatedly.

Now, the buttons show up in red text. This is actually NetLogo's way of telling us that the commands here do not yet have any code associated with them. So let us create two procedures named setup and go.

The procedures are written, as shown in Fig. 1a, in the procedures tab, and the comments (which come after a semi-colon on any line in NetLogo) explain what the commands do.

The code in Fig. 1a creates 100 turtles (nodes in our case) on the screen. However, the shape is a peculiar triangle by default, and colors are assigned at random. Notice that we have written code here to have the patches colored randomly.

To create the procedure for go, write the code shown in Fig. 1b.

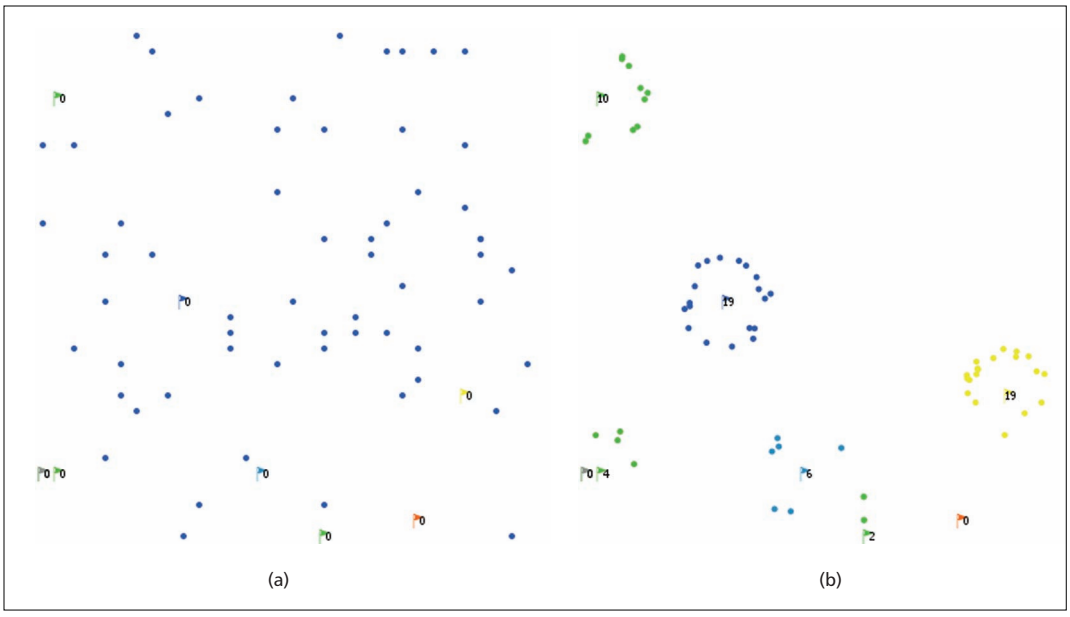

Figure 3. *a) Towers (flags) and nodes; b) nodes after flocking.*

There are two types of nodes; one type is spatially fixed and corresponds to the transmission towers. The second type of nodes perform a random walk until they come within a certain (pre-specified) radius of one of the towers, when they stop moving and change their color to the transmission tower's color.

Now, if we press setup and then go, we see turtles walking slowly in a forward direction on the screen, a snapshot of which is shown in Fig. 2.

Modeling Self-Organization

Experiment I: Modeling Flocking of Mobile Nodes — In this model flocking of mobile wireless nodes within a certain radius of a transmission tower is simulated. There are two types of nodes. One type is spatially fixed and corresponds to the transmission towers. The second type performs a random walk until they come within a certain (prespecified) radius of one of the towers, when they stop moving and change their color to the transmission tower's color. We are using this behavior to model flocking of wireless nodes within a certain radius of a tower. These are illustrated in Fig. 3.

Modeling Complex Paradigms in P2P Systems

In the next set of experiments we demonstrate the use of NetLogo for modeling and simulation of complex paradigms in P2P systems.

Experiment II: Clustering of Nodes in a P2P System — Clustering is an important technique extensively used in the literature [8]. In this example we demonstrate the use of NetLogo for clustering in P2P systems. To generate this self-organizing behavior, nodes initiate messaging between nodes. The nodes first discover other nodes within their sensing radius. Next, nodes perform an election algorithm looking for a node with the lowest ID (as is the norm in election algorithms). Figures 4a and 4b illustrate , the initial unclustered nodes and clustered nodes after applying self-organization, respectively.

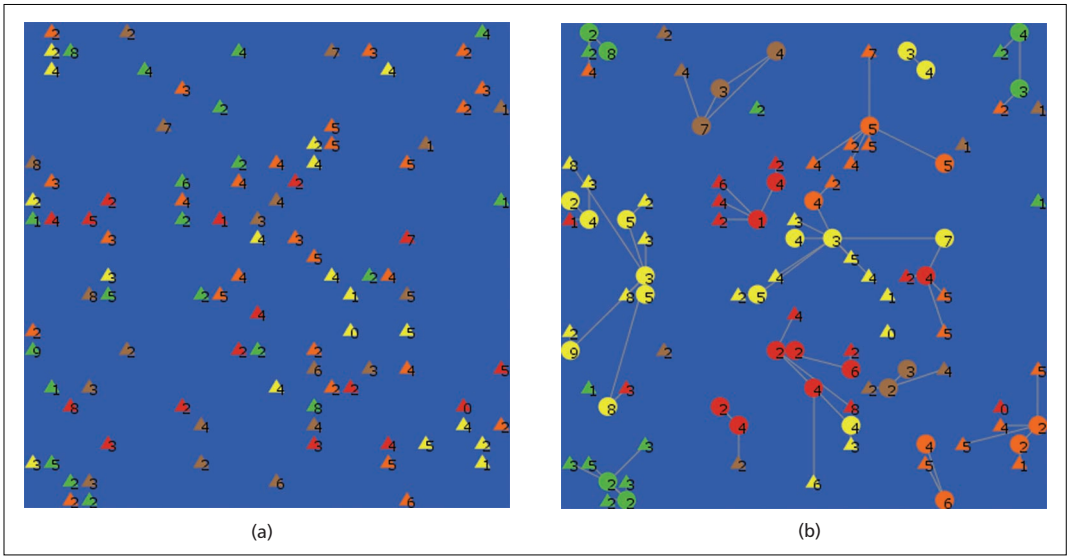

Figure 4. *a) Initial unclustered nodes; b) clustered nodes after applying self-organization.*

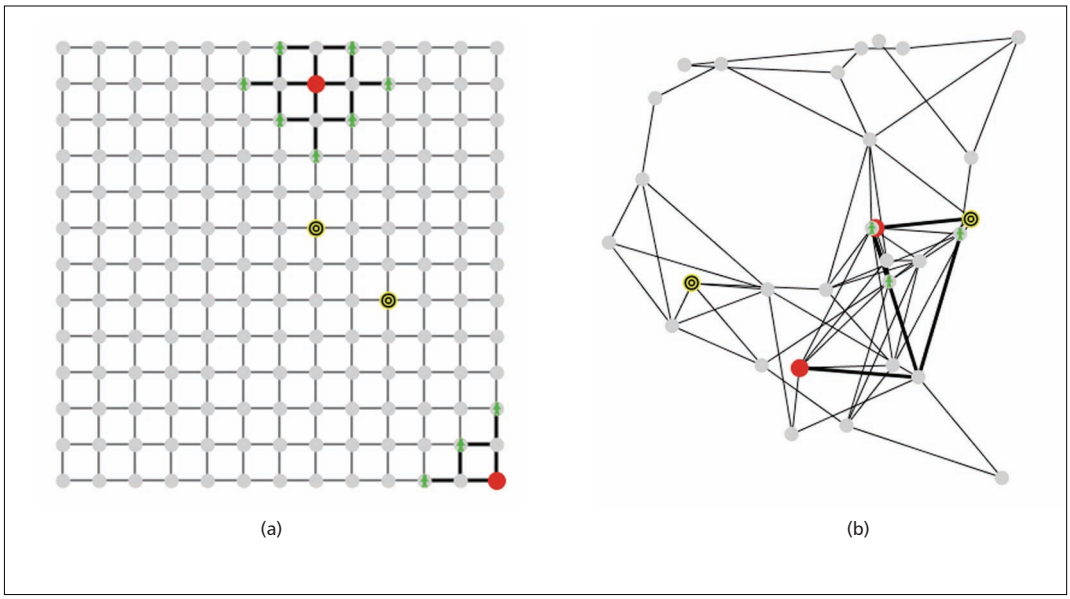

Figure 5. *a) Expanding ring algorithm in a lattice; b) k-random walk with check on a randomly connected graph.*

Experiment III: Modeling Unstructured Overlay Networks — Here we show how NetLogo can be used to model real-world examples of unstructured overlay forming algorithms such as Gnutella, random walk in P2P networks, as well as pervasive environments as shown in [9]. Figure 5a shows the expanding ring algorithm being applied to a lattice. The expanding ring algorithm is an advanced form of flooding algorithm where flooding is first performed with a time-to-live (TTL) value of 1 and then subsequently 3, 5, and so on, until one of the queries reaches the destination. Figure 5b shows the application of an advanced form of random walk, *k*-random walk with check, where there are *k* random walkers. The idea is that every few hops, these random walkers (queries) send a message back to the source node to verify whether or not the query needs to continue its quest.

Experiment IV: Distributed Averaging Using Gradients — In this final experiment we first show, in Fig. 6a, how NetLogo can be used to evaluate distributed consensus formation using gradients. Next, we illustrate gradient formation in wireless sensors (Fig. 6b). Fast self-healing gradients are discussed in [10].

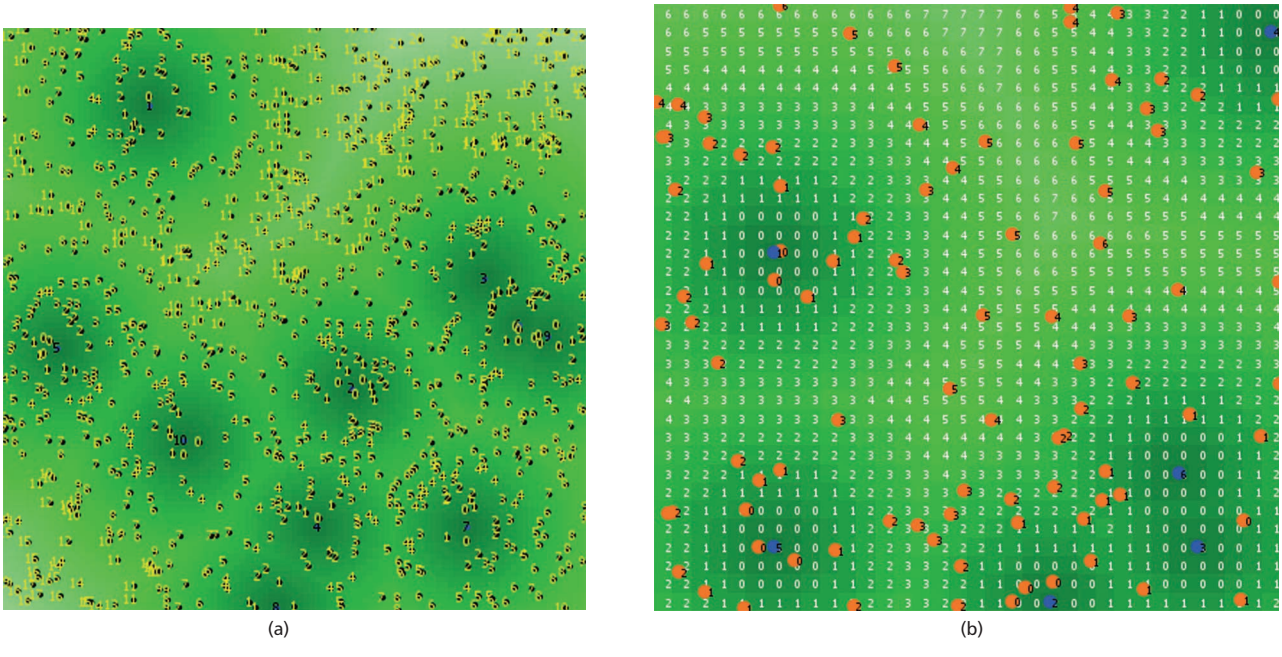

Figure 6. *a) Mobility based-consensus formation in* n = *1000 nodes; b) gradients formed around wireless sensors.*

CONCLUSIONS

In this article we have demonstrated the use of NetLogo, an increasingly popular tool for agent-based modeling in the M&S communities, from the perspective of modeling and simulation of self-organizing mobile and P2P systems. We have illustrated the strengths and distinctive aspects of agent-based modeling tools by employing a number of simple experiments to demonstrate the utility of the NetLogo tool in the important domain of modeling and simulation of P2P and ad hoc networks. In summary, we conclude that NetLogo is not only extremely flexible with a very short learning curve, but its powerful generic capabilities can readily be exploited to model self-organization in any complex system.

ACKNOWLEDGMENTS

We would like to thank Prof. Jose Vidal for his suggestion to try out NetLogo and Prof. Michael Huhns for helping in selection of agent-based toolkits, both from the University of South Carolina.

REFERENCES

[1] L. F. Perrone, Y. Yuan, and D. M. Nicol, "Simulation of Large Scale Networks II: Modeling and Simulation Best Practices for Wireless Ad Hoc Networks," *Proc. 35th Winter Simulation Conf.*, 2003, pp. 685–93.

[2] V. Naoumov and T. Gross, "Simulation of Large Ad Hoc Networks," *Proc. 6th ACM Int'l. Wksp. Modeling Analysis Simulation Wireless Mobile Sys.*, 2003, pp. 50–57.

[3] K. Pawlikowski, J. Jeong, and R. Lee, "On Credibility of Simulation Studies of Telecommunications Networks," *IEEE Commun. Mag.*, 2000.

[4] M. Niazi and A. Baig, "Phased Approach to Simulation of Security Algorithms for Ambient Intelligent (AmI) Environments," *Proc. 40th Winter Simulation Conf.*, Ph.D. Student Colloqium, Dec. 7–11, 2007.

[5] U. Wilensky, NetLogo, Center for Connected Learning Comp.-Based Modeling, Northwestern Univ., Evanston, IL, 1999; http://ccl.northwestern.edu/netlogo

[6] J. M. Vidal, P. Buhler, and H. Goradia, "The Past and Future of Multiagent Systems," *AAMAS Wksp. Teaching Multi-Agent Sys.*, 2004.

[7] M. Niazi et al., "Simulation of the Research Process," *Proc. 2008 Winter Simulation Conf.*, 2008, pp. 1326–34.

[8] O. Younis, and S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks,"*IEEE Trans. Mobile Comp.*, vol. 3, no. 4, pp. 366–79.

[9] M. Niazi, "Self-Organized Customized Content Delivery Architecture for Ambient Assisted Environments," *UPGRADE-CN '08*, Boston, MA, June 23, 2008.

[10] J. Beal *et al.*, "Fast Self-Healing Gradients," *ACM SAC '08*, Fortaleza, Ceará, Brazil, Mar. 16–20, 2008.

BIOGRAPHIES

MUAZ NIAZI [M] (man@cs.stir.ac.uk) obtained his B.S. in electrical engineering from NWFP UET, Pakistan, and a Master's degree in computer science from Boston University in 1996 and 2004, respectively. After working in industry for several years, he is currently an assistant professor of software engineering at Foundation University, Pakistan, as well as a doctoral student at the University of Stirling, Scotland. He is a member of the IEEE Communications and Computational Intelligence Societies. He is also a member of the IEEE CIS Task Force on Intelligent Agents and the IEEE CIS Task Force on Organic Computing. He has served on the program committees of a number of conferences and workshops such as Bio-Inspired Algorithms for Distributed Systems and the IEEE Wireless Communications and Networking Conference 2009, as well as the IEEE CIS Symposium on Intelligent Agents, where he is also co-chair of a special session on self-adaptive agents. He is also a reviewer for international journals, including the *Elsevier Journal of Network and Computer Applications*. His areas of research interest are modeling and simulation of self-organizing and self-adaptive systems in the P2P and ad hoc network domain, and novel applications of socially inspired techniques.

AMIR HUSSAIN [SM] (a.hussain@cs.stir.ac.uk) obtained his B.Eng. (with 1st Class Honors) and Ph.D., both in electronic and electrical engineering from the University of Strathclyde in Glasgow, Scotland, in 1992 and 1996, respectively. He is currently a reader in computing science at the University of Stirling in Scotland. His research interests are mainly interdisciplinary, and include machine learning and cognitive computing for modeling and control of complex systems. He holds one international patent in neural computing, is editor of five books, and has published over 100 papers in journals and refereed international conference proceedings. He is Editor-in-Chief of Springer's *Cognitive Computation*, Associate Editor for *IEEE Transactions on Neural Networks*, and serves on the Editorial Boards of a number of other journals. He is IEEE Chapter Chair of the U.K. & RI IEEE Industry Applications Society and is a Fellow of the U.K. Higher Education Academy.

NetLogo is not only extremely flexible with a very short learning curve, but its powerful generic capabilities can be readily exploited to model self-organization in any complex system.