Designing Dependable Agent Systems for Mobile Wireless Networks

Vincent Cicirello, Maxim Peysakhov, Gustave Anderson, Gaurav Naik, Kenneth Tsang, William Regli, and Moshe Kam, *Drexel University*

A mobile ad hoc network (MANET) is a wireless network of mobile devices—such as PDAs, laptops, cell phones, and other lightweight, easily transportable computing devices—in which each node can act as a router for network traffic rather than relying on fixed networking infrastructure.[1] As mobile computing becomes ubiquitous, MANETS will become increasingly important. For example, MANETS can provide a bring-your-own-network solution to the problems inherent in communications between rescue workers at the location of a natural disaster, where a traditional networking infrastructure isn't likely to exist or at best will likely be inoperable. Drexel University is studying these problems as part of the Philadelphia Area Urban Wireless Network Testbed (PA-UWNT), an effort linking the university with local and regional government to study the needs of such first responders.

Mobile ad hoc networks offer a bring-your-own-network solution to rescue workers at natural-disaster locations, where a traditional networking infrastructure isn't likely to exist. As a design paradigm, multiagent systems can help support management and coordination in such environments.

As a design paradigm, multiagent systems (MASs) can help facilitate and coordinate ad hoc scenarios that might include security personnel, rescue workers, police officers, firefighters, and paramedics. Setting the groundwork for this vision, Drexel University has been working on integrating mobile agent systems and MANETS. Drexel University's live MANET consists of dozens of mobile-computing nodes—PDAs, tablets, and laptops—on an 802.11b wireless network with ad hoc routing. On this network, mobile agents perform critical functions that include delivering messages, monitoring resource usage on constrained mobile devices, assessing network traffic patterns, analyzing host behaviors, and revoking access rights for suspicious hosts and agents. Figure 1 shows a few members of the Drexel University research group with the MANET's hardware infrastructure.

In such complex and dynamic environments, the behavior of the agent population comprising the MAS must be self-stabilizing. It would be far too costly—if possible at all—to attempt to exert any central control over the system because the MANET's environment is inherently far too dynamic and the global state information is far too distributed. Agents can effectively operate in such environments if they are *environment aware*—if they can sense and reason about their complex and dynamic environments. Altogether, agents living on a MANET must be network, information, and performance aware. This article fleshes out how we apply this approach to populations of mobile agents on a live MANET.

MAS on MANET: The challenges

For a dependable MAS to operate on a MANET, it must first overcome several challenges, one of which is the dynamic network environment's inherent complexity and uncertainty. Agents operating in a MANET environment must reason about complex network information that can include dynamically changing topology, resource-constrained computing devices, multihop routes, signal strength, packet loss, jitter, and bandwidth limitations.

An agent might be involved in communications with another agent located on a host to which a route ceases to exist—or perhaps the host itself has left the network. The MANET might become severed, possibly denying agents access to key services. Agents also must deal with uncertain communications. Packets might be lost in transit, signal strength between any two hosts can change at any moment, and the number of hops or even the existence of routes between hosts can change dynamically.

1541-1672/04/$20.00 © 2004 IEEE
Published by the IEEE Computer Society

Figure 1. Members of the Drexel University research group and pieces of the MANET's hardware infrastructure.

Another challenge is communications integrity and information assurance. A given scenario might require a high degree of information assurance. Agents might have to detect and react to network intruders and malicious insiders, which can include requiring agents to reason not only about the posture of the MANET in terms of information integrity and assurance but also about the interoperation of network dynamics and MANET information assurance. So an agent might have to reason about MANET route redundancies to avoid transmitting data through a suspected compromised host. Simply removing the compromised host isn't always the solution, because this could sever necessary communications channels and adversely affect information availability.

Another challenge is measuring and evaluating how a MAS operating on a MANET performs. The MANET environment only amplifies the usual issues of evaluating the large amount of data the MAS generates and coordinating the data gathered and composed from a distributed system.[2] The network's dynamics add to the already massive volume of data to collect, and resource-constrained computing devices are particularly challenging when storing performance data. Furthermore, resource-constrained devices and the bandwidth limitations of MANETs amplify the impact of performance measurement. For example, limited computing cycles must be allocated to the MAS and to the computation of performance metrics.

To face these challenges effectively, a dependable MAS operating on a MANET must be adaptive, stable, and redundant. The MANET environment is constantly changing. What might be a good behavioral policy for an agent at one moment can become ineffective and suboptimal over time and even mislead an agent into making bad decisions. Agents living on a MANET must be capable of adapting their behavior as the environment changes.

Biologists consider a system ecologically stable if it can resist change while maintaining function and can recover to normal levels of function after a disturbance.[3] For a MAS operating on a MANET, ecological stability is important. As the MANET changes dynamically, the MAS must continue to perform its intended functions dependably. At the same time, if a large enough disturbance occurs in the MANET to alter the MAS's functioning significantly, the MAS must be able to recover to a desirable steady state of operation.

A MAS operating on a MANET can't rely on any single agent or host or assume that the network is dependable. A host can disappear without warning—for example, its hardware might fail or a malicious user could take it over. Agents or services executing on that host might cease to exist if they can't migrate to another host before the failure occurs. In addition, the network topology can become severed, potentially stranding mobile agents on outlying network nodes or preventing communications between agents or users at different locations on the network.

Environment-aware agents

Building a dependable MAS on a MANET requires that the agents be aware of the environment in which they operate. This means developing agents that can reason about a constant flow of complex, dynamic data, such as dynamic network data, information assurance, and performance data (see Figure 2).

The network state's dynamic and uncertain nature, if ignored, can result in inefficiencies or even complete failures. For example, a mobile agent might need to migrate to several hosts sequentially to complete a given task. An agent tasked with monitoring the battery levels of the hosts on the network might need to traverse the network, collect the data, and return to its source—perhaps a network management tool. For any given MANET topology, one or more itineraries might exist that are more efficient than others.

This is essentially the traveling salesperson problem (TSP). An agent that considers the MANET's topology as well as other data, such as signal strengths of the links in selecting its itinerary, is likely to require less time to complete its data-collection task. The problem is further complicated by the fact that a MANET's topology and other network characteristics are dynamic. An itinerary planned ahead of time using a TSP heuristic can become obsolete quickly as the network's link-state changes.

An agent that considers such dynamics and dynamically reasons about the MANET's uncertainty—perhaps predicting future topological change—can further enhance its data-collection performance, possibly returning more timely results and preventing a costly failure. Figure 3 illustrates network awareness for mobile agents on a MANET. Agents at the agent system layer must reason about the MANET layer's dynamics.

In a critical application of a MAS on a MANET, agents must be able to reason about malicious agents and hosts. The lightweight nature of the computing devices in such scenarios can increase the risk of compromise. For example, just as you can easily carry a PDA in your pocket, a PDA can easily fall out of your pocket. If an adversary later finds your PDA, you'd want the MAS to detect this host compromise. You'd also want its agents to react to knowledge of the malicious host and react to agents originating from it.

An agent can benefit from the ability to reason about performance data of the MAS on the MANET. For example, an agent of the MAS might require a particular service. It might send out a child agent to a host to obtain some required information. An agent that can reason about the result's timeliness might be able to detect when the MAS should spawn another instance of this service on some other host or migrate an existing instance from a less-accessible to a more-accessible host. Perhaps the service is located on a host that was formerly in a central position in the MANET's topology but over time has moved to the fringe of the MANET.

Do all agents need to reason about all types of data to be dependable? Of course not. The spectrum of data types from which a MAS operating on a MANET can benefit is quite large, and the reasoning problems are complex. If an agent considered all of the available data when reasoning about its actions, the agent would never be able to do anything. The data an agent should reason about and the methods used for that reasoning are application specific. Not all agents must be network, information, or performance aware.

Integrating mobile agents

In PA-UWNT, mobile agents assess network traffic patterns, analyze host behaviors, revoke access rights for suspicious hosts or agents, adaptively reroute traffic at the network layer to improve the overall system's information integrity, and provide the implementation framework for several decentralized user applications. The agent architecture within which we have implemented our mobile agents is the Extendable Mobile Agent Architecture of Lockheed Martin's Advanced Technology Laboratories.

The EMAA framework includes autonomous and asynchronous agents as well as management mechanisms for agent mobility, agent events, and interagent communication.[4] Architecturally, EMAA consists of three core layers: docks, servers, and agents. Docks provide the execution environment on a host in which all other components are executed. Servers provide heavy-weight or fixed services, while the agents are lighter and more mobile, each having task lists and itineraries. While highly autonomous, EMAA agents can still receive and respond to commands from a controlling authority, thus balancing between totally autonomous behavior and cooperation to command centers that human users can control.[5]

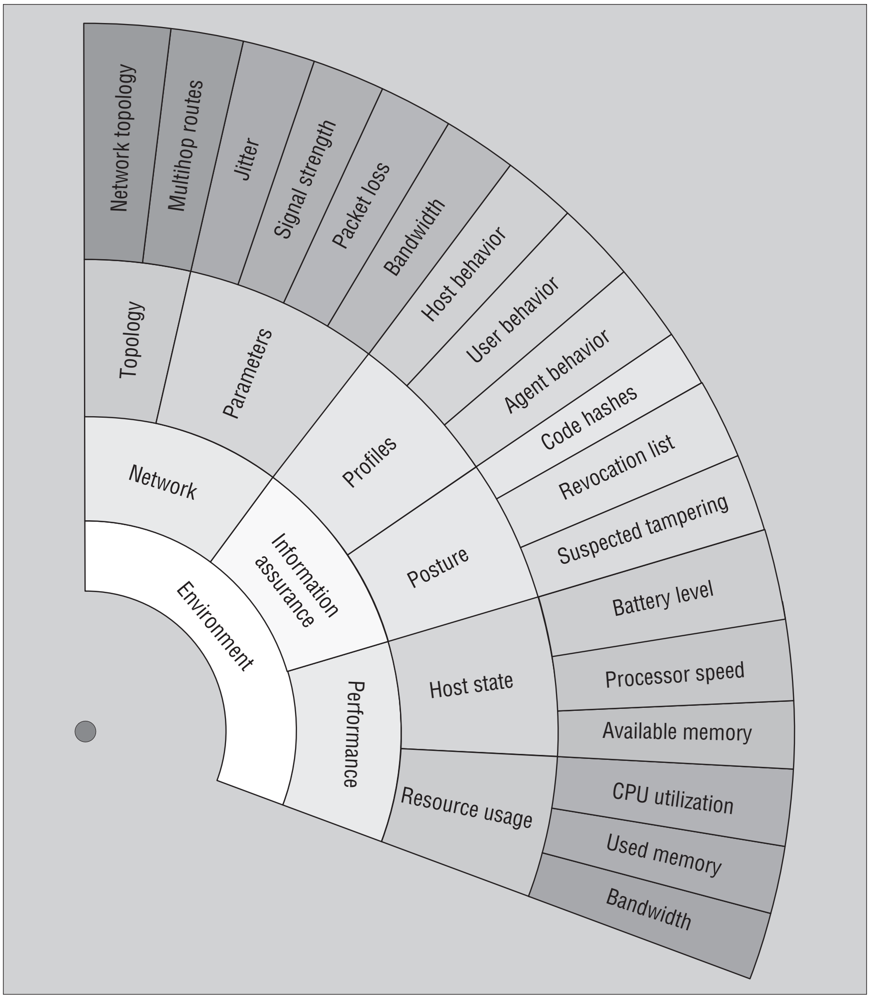


Figure 2. Hierarchical view of the agent's operating environment. Agents might need to be network aware, information assurance aware, and performance aware. Each requires reasoning with a spectrum of complex, dynamic data.

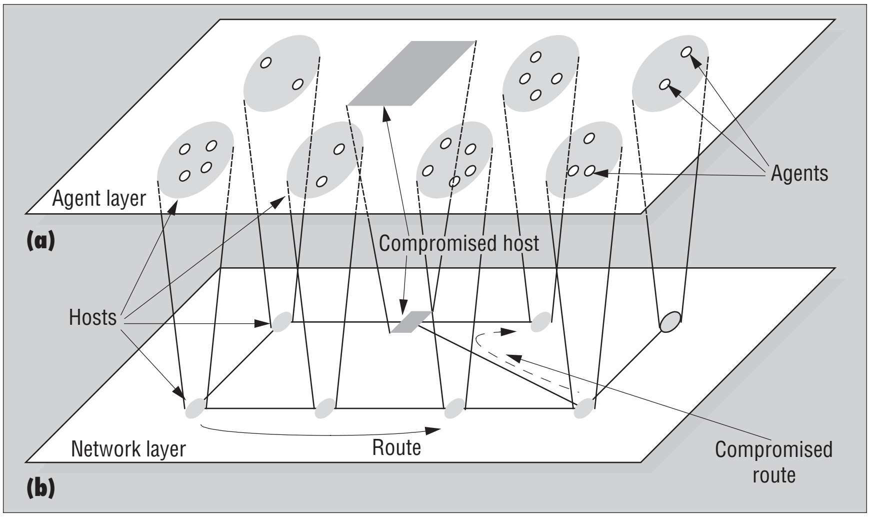


Figure 3. Example of network-aware agents showing a two-layer view of the MAS on the MANET: (a) the agent system layer, showing agents on the hosts (the agents at the agent system layer reason about the network layer to make more intelligent decisions); (b) the network layer, showing the direct connections between hosts on the MANET.

Table 1. Results of the experiment

Agent type	Visited	Visited/second	Percentage of agents lost	Percentage of agents compromised
Naive	7.3	0.027	31.1	97.8
MANET aware	7.5	0.037	11.8	91.8
Information assurance aware	6.6	0.052	24.7	12.3

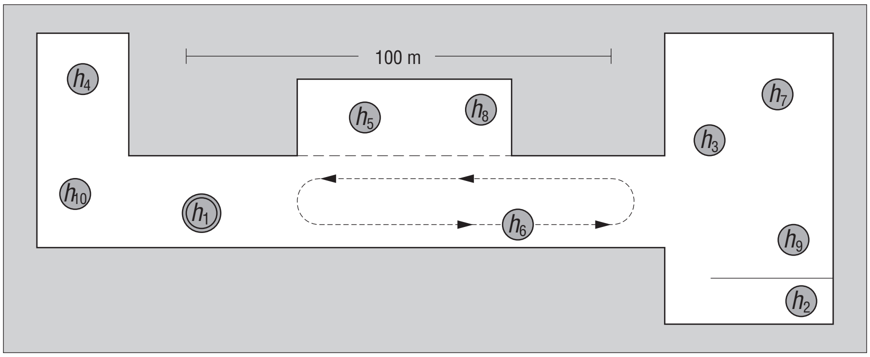


Figure 4. The experimental setup inside of a building with a compromised host in the middle.

One type of mobile agent that we implemented performs a rudimentary level of intrusion detection. This agent is mobile and wanders the MANET, examining hosts for one particular type of intrusion. Specifically, its job is to check for modifications to certain predetermined segments of host-level code critical to the safe operation of the other mobile agents. It performs this check, not by directly comparing the server code to known uncompromised code, but by comparing a one-way hash of the server code segments to a hashed value of the known uncompromised version. This intrusion-detection agent then can report intrusions to system components that can react to such intrusions—for example, revoking a suspect host's access rights.

Other mobile agents collect data regarding network characteristics, resource use, bandwidth use, and so forth. These mobile agents deliver such data to services and other agents interested in it. For example, an intrusion-detection server or agent might examine this gathered data for anomalous patterns of network and resource use. Thus, this information-aware service relies on reasoning processes that require network awareness and performance awareness. It must reason about the system's normal behavior from a network-aware and performance-aware perspective, detect abnormalities, and inform the appropriate system components about possible intrusions.

When provided with information regarding hosts that might be compromised—detected by the intrusion-detection service—an information-aware mobile agent can plan its itinerary across the MANET accordingly, avoiding migrating to a suspect host if the possible threat from such a visit is deemed severe enough to warrant such action.[6] An agent makes this information-aware decision by considering an information-assurance metric $V(N, \alpha, h_i, h_j)$ that we define as

$$V(N, \alpha, h_i, h_j) = \varpi_K(h_i, h_j)((\alpha)\mathit{integrity}(N, h_i, h_j) + (1-\alpha)\mathit{availability}(N, h_i, h_j)),$$

where α trades off integrity and availability and h_i, h_j is a pair of hosts for which the metric is being computed. The *integrity* rating is the product of the integrity levels of the hosts along the route. We obtain the *availability* rating by the link weightings and the length of the route.[6] This approach offers a formal method of balancing the risk inherent with an agent traveling through a possibly compromised host and the timeliness of an agent completing its assigned tasks.

Table 1 shows the results of an indoor experiment conducted on a live MANET. As Figure 4 shows, a compromised mobile agent moves in a specified area in the middle of the hallway. In the experiment, users of mobile hosts are in continuous motion throughout the facility. Host h_1 spawns an agent of three types each second. Each of these agents has an itinerary of hosts to visit before returning to h_1. The naive agent follows the given itinerary. The MANET-aware agent reasons about network topology to optimize its route. The information-assurance-aware agent reasons about network topology and host compromises to avoid migrating through compromised hosts while optimizing its route.

The results show that network awareness increases the efficiency with which the tasks are performed and that information assurance awareness results in significantly fewer agents traveling through the suspected compromised host—maintaining a higher level of information integrity and assurance.

Managing services

Mobile agents living on a MANET require services, some of which might be provided by other mobile agents. Other services might be statically located for one reason or another, perhaps because it would be too bandwidth intensive to mobilize a particular service. The benefit to having a mobile agent provide the service might be outweighed by the cost of agent mobility. A complexity associated with using fixed network locations for services on a MANET is determining where those locations should reside. The optimal locations will be in constant motion as the MANET topology changes. Also, if the demand for the service changes, then a different number of instances of a particular service might be required. Given the limited computational resources of mobile devices on a MANET, it might not be feasible to replicate such a service on every host.

The problem can be summarized as follows: We have a computationally intensive and critical service that would require far too much bandwidth and time to encapsulate in a mobile agent. We thus need to maintain a minimal set of these services on the MANET to fulfill the demand placed on the service by the MAS's agents. The appropriate locations of these services might also change because of the changing MANET topology and the importance of mobile agents efficiently finding their way to one of the instances. For example, the code-hashing agent we discussed earlier might deliver its hashes to an intrusion-detection service, monitoring agents might deliver data about the behavior of or actions taken by hosts, and so forth. This problem is equivalent to the NP-Hard problem known as dynamic servers.[7]

To maintain the MANET's integrity, agents must detect compromised hosts as quickly as possible. The intrusion-detection service will be increasingly effective with the timeliness of the data with which it reasons. Our approach to service availability is to provide all hosts with the ability to provide a service. A population of mobile agents, coordinated by the quorum sensing behavior of ants for nest relocation, wanders the MANET with the ability to start and stop instances of the service.[8] Figure 5 illustrates this ant-inspired approach to service availability management.

The approach works as follows:

- The internal state variable v of an ant-inspired service availability management agent is initialized by system to a desired target time interval t. The time interval t is the desired average time a randomly wandering agent requires to find an instance of the service.
- The variable v is then reset by the system to t each time the ant takes the action of starting or shutting down a service.
- As the ant traverses the network randomly, it keeps track of the time elapsed since it has last seen the service. It does this by decrementing v by one for each millisecond of elapsed real time.
- Every time the ant encounters the service, its state variable v is increased by a value of $t(v = v + t)$.
- If v falls below a bound called the lower threshold, the ant starts another instance of the service on the current host. If v exceeds a bound called the upper threshold, the ant will shut down the service on that host.

In this way, network-aware mobile agents that explore the MANET's changing topology can manage a set of service instances by maintaining the location of the service that is capable of handling the demand imposed upon it by the MAS without overburdening the MANET's resources with an excessive number of instances. Other researchers have worked on an ant-based approach to this problem.[9]

Agent ecosystems

In the preceding examples, we have a population of code-hashing agents, a population of mobile agents managing service availability, and perhaps additional populations of mobile agents gathering other data needed by the intrusion-detection services. All these mobile agents use MANET bandwidth as well as host resources, such as CPU and memory. It's important to not have unnecessarily large populations of these agents, but at the same time, we need enough service availability management agents to maintain a cost-effective distribution of intrusion-detection services. We also need enough code-hashing and monitoring agents to provide the necessary data to these services for early intrusion detection.

Our approach to the problem of determining the number of agents appropriate for a task is inspired by large biological ecosystems—an agent ecosystem.[10] Large ecosystems usually have several attractive qualities—such as dynamic decentralized control, self-regulation, redundancy, robustness, and stability—that are required for maintaining these populations of mobile agents in our system. Figure 6 illustrates this approach.

In this approach, each task that a mobile agent can perform in our system will be associated with food. Code hash data is one type of food; we associate other types of data that services require with other food types. Agents consume food points over time by successfully completing tasks, thereby sustaining their existence.

For example, when a code-hashing agent or a monitoring agent delivers data to a service, it receives artificial food proportional to the length of time the service has been waiting for this type of agent to visit. If the agent goes to a recently visited service , then the agent will receive less of a reward than if it had visited a service starving for required data. The service-availability management agents receive rewards whenever they start or stop a service.

Food points correspond to an agent's level of health. Over time, accumulated points decay, and agents that exhaust their food supply die—that is, their execution is terminated. This corresponds to an insufficient demand to support an agent population's current size and thus reduces the number of agents that are taxing the MANET with bandwidth and CPU usage. In contrast, if an agent collects enough food points to push its health level up to some predetermined level, then it spawns a copy of itself, distributing its health level between the two copies of the agent. This process corresponds to a demand for a task that the particular agent type provides that the current agent population can't satisfy.

Ecosystem stability

We can describe and formally analyze the collective behavior of a population of these ecologically inspired agents. The set H denotes the set of producers. That is, each $h \in H$ is a host on the MANET for which some resource must be monitored by the population of mobile agents. The function $F_h(t)$ is

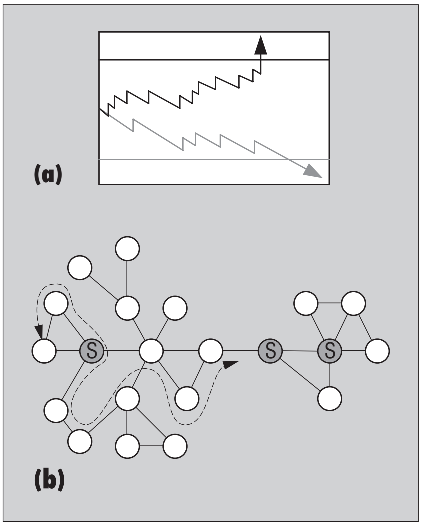


Figure 5. (a) The internal state of a wandering service-management agent. If the agent crosses the upper threshold, an instance of the service is stopped. If the lower threshold is crossed, a new instance of the service is started. (b) The dashed line illustrates a wandering service management agent. The S indicates active services.

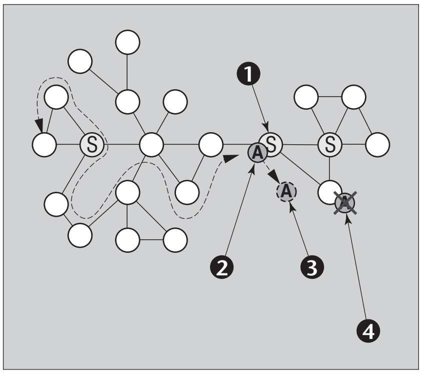


Figure 6. Controlling the size of populations of mobile information-gathering agents: (1) An instance of an intrusion-detection service interested in resource usage, network usage, code hashes, and so on; a wandering resource monitoring agent (2) receiving a "food" reward as it delivers required resource usage data to an intrusion-detection service, (3) being cloned as its health (food level) exceeds a threshold, and (4) dying as its health (food level) declines to 0.

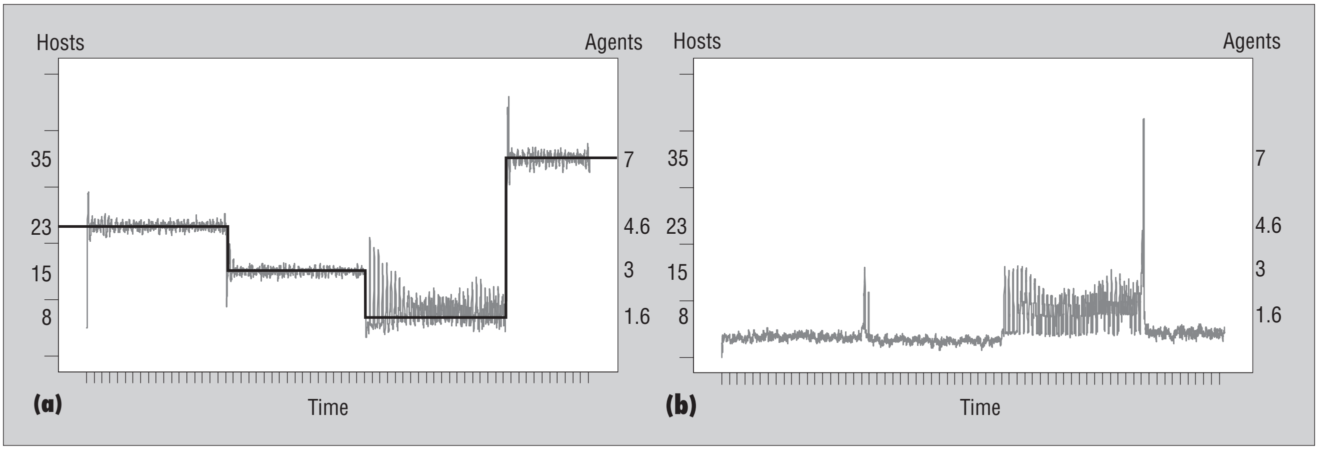


Figure 7. System behavior over time. The number of hosts changed from 23 to 15 to 8 to 35. (a) Horizontal lines show the equilibrium number of agents. Superimposed on this is a tracking of the actual number of agents over time. (b) The standard deviation over time.

the rate at which host *h* produces food for the mobile agents. The set *A* defines the set of mobile consumer agents.

Each $a \in A$ has a consumption function $f_a(t)$, which is the rate at which the agent's collected food points decay. The dynamic system of *H* producers and A consumers is considered to be in an equilibrium state over some period of time from t_1 to t_2, if and only if the amount of food produced during that time period equals the amount of food consumed during that same time period. This relationship can be expressed in the following equation:

$$\sum_{h \in H} \int_{t_1}^{t_2} F_h(t)dt = \sum_{a \in A} \int_{t_1}^{t_2} f_a(t)dt$$

In our implemented system, the production and consumption rates, *c* and *d* respectively, are constant. The previous equation and the following equations define the equilibrium state for this example agent population:

$$\sum_{h \in H} \int_{t_1}^{t_2} c\,dt = \sum_{a \in A} \int_{t_1}^{t_2} d\,dt$$

$$|H| \times c \times (t_2 - t_1) = |A| \times d \times (t_2 - t_1)$$

$$|A| = |H| \frac{c}{d}$$

In one experiment, we set the consumption rate *d* for our mobile agents to 5 units of food. Each host produced 1 food point per time unit. We ran 15 trials. Initially, there were 23 hosts. After 30,000 seconds, the number dropped to 15, then to 8 hosts after an additional 30,000 seconds. The number of hosts later increased to 35. Figure 7 shows the behavior of the system over time.

The solid lines show the equilibrium number of agents (4.6, 3.0, 1.6, and 7.0). Superimposed on this is a tracking of the actual number of agents over time. Whenever hosts were shut down, all agents on these hosts and agents traveling to these hosts were terminated. New hosts initially had no food or agents. The agent ecosystem approach closely tracks the equilibrium in all cases except where the equilibrium number of agents is small, where we see significant noise.

This approach also can adapt to changes in the number of hosts. The source of the instability when the system needs few agents is due to the system overcompensating for when that agent population becomes extinct. The system might spawn too many agents, which can then lead to rapid extinction to compensate for the overabundance of agents. This is an issue we plan to address. A more thorough stability analysis is currently underway.

We have taken some of the first steps toward developing dependable, environment-aware MASs on a MANET. Our future research will address many more challenges. For example, what happens if a compromised host can spawn numerous seemingly legitimate resource-monitoring agents? If the agents haven't been tampered with, the ecosystem control strategy will eventually kill off the excess agents. If this behavior of the host is far enough from the norm, then the compromised host will be detected and have its access rights revoked. But what if the instances have been tampered with? An authentication protocol might be used to authenticate the agent's identity. Or a possible solution might incorporate an approach related to the hash agent used to check a host's code.

In most cases, the agent population stabilizes around an equilibrium number of agents. Although the agent population can adapt to changes in the dynamic network state, the population size can vary greatly. We still need refinements to perfect the ecosystem control policy. The advantage of taking such a decentralized approach is that we don't rely on a centralized control system that can serve as a single point of failure. The MANET environment's distributed and dynamic nature puts a centralized approach at a disadvantage in that it is difficult, if not impossible, to maintain an up-to-date global view of the system. Developing a dependable MAS on a MANET is a difficult research challenge indeed. The diversity and complexity of the data that constantly flows through the MANET environment, as well as the MANET's inherent dynamics, creates an intractable problem for centralized control approaches. ■

Acknowledgments

Thanks to the Drexel University Department of Public Safety for their assistance in setting up the PA-UWNT.

References

1. *Computer*, special issue on ad hoc networks, Feb. 2004.

2. A. Helsinger et al., “Tools and Techniques for Performance Measurement of Large Distributed Multi-Agent Systems,” *Proc. Int’l Joint Conf. Autonomous Agents and Multi-Agent Systems*, ACM Press, 2003, pp. 843–850.

3. K.H. Johnson et al., “Biodiversity and the Productivity and Stability of Ecosystems,” *Trends in Ecology and Evolution*, Sept. 1996, pp. 372–377.

4. R.P. Lentini et al., “Emaa: An Extendable Mobile Agent Architecture,” *Proc. AAAI Workshop Software Tools for Developing Agents*, AAAI Press, 1998, pp. 133–134.

5. J. McCormick et al., *A Distributed Event Messaging System for Mobile Agent Communication*, tech. report TR-01-02, Lockheed Martin Advanced Technology Laboratory, 2000; www.atl.external.lmco.com/overview/library.html.

6. M. Peysakhov et al., “Network Awareness for Mobile Agents on Ad Hoc Networks,” *Proc. 3rd Int’l Joint Conf. Autonomous Agents and Multi Agent Systems*, ACM Press, 2004, pp. 368–375.

7. M. Charikar, D. Halperin, and R. Motwani, “The Dynamic Servers Problem,” *Proc. 9th Ann. ACM-SIAM Symp. Discrete Algorithms*, ACM Press, 1998, pp. 410–419.

8. S. Pratt et al., “Quorum Sensing, Recruitment, and Collective Decision-Making During Colony Emigration by the Ant Leptothorax Albipennis,” *Behavioral Ecology and Sociobiology*, vol. 52, no. 2, 2002, pp. 117–127.

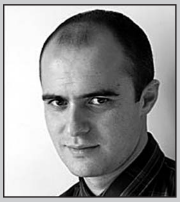
9. H. Van Dyke Parunak and S. A. Brueckner, “Self-Organizing MANET Management,” *Proc. Workshop Eng. Self-Organizing Applications*, LNCS 2977, Springer-Verlag, 2003, pp. 20–35.

10. M. Peysakhov, V.A. Cicirello, and W. Regli, “Ecologically Inspired Agent Control Model,” *Proc. 3rd NASA-Goddard/IEEE Workshop Formal Approaches to Agent-Based Systems*, IEEE Press, 2004, pp. 26–28.

For more on this or any other computing topic, see our Digital Library at www.computer.org/publications/dlib.

The Authors

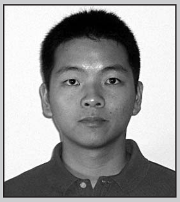
Vincent Cicirello is a research scientist in the College of Engineering at Drexel University. His research interests include applied artificial intelligence, multiagent systems, machine learning, planning and scheduling, biologically inspired computing, and anytime algorithms. He received his PhD in robotics from Carnegie Mellon University. He is a member of the AAAI, ACM, IEEE, and SIAM. Contact him at the Dept. of Computer Science, Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; cicirello@cs.drexel.edu.



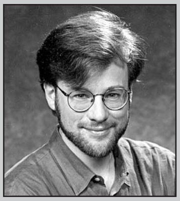
Maxim Peysakhov is a PhD candidate in the Department of Computer Science at Drexel University. His research interests include nature-inspired algorithms, evolutionary computation, automatic design generation, and software agents. He received his MS in software engineering from Drexel University. Contact him at the Dept. of Computer Science, Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; peysakhov@cs.drexel.edu.

Gustave Anderson is a PhD candidate in the Department of Electrical and Computer Engineering at Drexel University. His research interests include network security, software architecture, mobile ad hoc networks, and cryptography. He is a student member of the IEEE. Contact him at the Dept. of Electrical and Computer Eng., Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; gus@minerva.ece.drexel.edu.

Gaurav Naik is an undergraduate student in the Department of Electrical and Computer Engineering at Drexel University. His research interests include distributed systems, discrete event systems, and network security. He is a student member of the IEEE and ACM. Contact him at the Dept. of Electrical and Computer Eng., Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; gnaik@minerva.ece.drexel.edu.



Kenneth Tsang is an undergraduate student in the Department of Electrical and Computer Engineering at Drexel University. His research interests include mobile agents, optimization, and RF interference. He serves as treasurer of Drexel University’s IEEE student branch. Contact him at the Dept. of Electrical and Computer Eng., Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; kt62@drexel.edu.



William Regli is an associate professor in the Department of Computer Science in the College of Engineering at Drexel University. He pursues interdisciplinary research spanning engineering design, graphics, and modeling and intelligent systems. He received his PhD in computer science from the University of Maryland at College Park. He is a member of the ACM, IEEE, AAAI, and Sigma Xi. Contact him at the Dept. of Computer Science, Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; regli@drexel.edu.

Moshe Kam is the Robert Quinn Professor of Electrical and Computer Engineering at Drexel University and is the director of Drexel University NSA Center of Excellence in Information Assurance Education and of Drexel’s Data Fusion Laboratory. His research interests include decision theory, detection and estimation, sensor fusion, dynamic systems, and network security. He received his PhD in electrical and computer engineering from Drexel. He is a fellow of the IEEE. Contact him at the Dept. of Electrical and Computer Eng., Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; kam@minerva.ece.drexel.edu.