Cooperating Mobile Agents for Dynamic Network Routing

Nelson Minar - Kwindla Hultman Kramer - Pattie Maes
MIT Media Lab, E15-305 20 Ames St, Cambridge MA 02139, USA
(nelson,khkramer,pattie)@media.mit.edu
http://www.media.mit.edu/~nelson/research/routes/

1 Introduction

Contemporary computer networks are heterogeneous; even a single network consists of many kinds of processors and communications channels. Networks are also inherently decentralized; capability is scattered across the system. But few system design methodologies embrace or even acknowledge these complexities. New methods and approaches are required if next-generation networks are to be configured, administered and utilized to their full potentials. In our research at the MIT Media Laboratory we are building systems that use mobile software agents to manage complex real-world networks. In this chapter we describe a strategy for using a collection of cooperating mobile agents to solve routing problems for dynamic, peer-to-peer networks.

Information networks continue to increase in size, complexity and importance. More and more devices are connected to computer networks, from desktop computers to cellular telephones, Web servers to pagers, television set-top boxes to smoke detectors. As connections proliferate, network topologies necessarily become more and more dynamic. Devices may move from place to place, or maintain intermittent connections, or change their relationships to the network and their peers on the fly. Networks must be flexible enough to allow these devices to communicate with each other in a variety of ways and across a variety of substrates. For example, physical links currently in widespread use include Ethernet, cellular radio, short-range infrared, and analog modem (to name a few).

The problem of configuring such dynamic and heterogeneous networks is difficult at several levels. Researchers in this domain face hard problems of packet routing, service coordination, and infrastructure security. In many cases, conventional centralized approaches to network management are inappropriate, unable to serve large, diverse, mutable collections of computers.

The routing tables of conventional network systems, for example, are usually generated in a centralized (and often human-mediated) manner. In this paper we present a contrasting model, a dynamic, wireless, peer to peer network with routing tasks performed in a decentralized and distributed fashion by mobile software agents that cooperate to accumulate and distribute connectivity information. Our agents determine system topology by exploring the network, then store this information in the nodes on the network. Other agents use this stored information to derive multi-hop routes across the network. We study these algorithms in simulation as an example of using populations of mobile agents to manage networks.

2 Managing Networks via Populations of Mobile Agents

2.1 The Importance of Decentralization

We believe that complex networks require decentralized management structures. The aggregation of control inherent to centralized systems makes it very difficult for such networks to scale upwards in size. Centralized management tools depend, by definition, on restricting important decisions to one or a few nodes. These special nodes become performance bottlenecks as they are required to serve an increasing numbers of clients. In addition, failure of the controlling nodes (or the inability, for whatever reason, of other nodes to communicate with them) poses serious difficulties to participants in centrally managed networks.

For similar reasons, designing centralized architectures that can readily adapt to changes in network usage is at least as difficult as designing them to scale well. Because the "intelligence" in a centralized network resides in a small number of specialized nodes, most devices on the network will be capable of only a limited range of behaviors. Asking devices with limited, hard-wired behavior to adjust themselves in response to changing circumstances – to use a different back-off algorithm, perhaps, or to structure their packets in a new way – will usually be impossible.

Finally, there is a fundamental incompatibility between centralized design frameworks and inherently distributed real-world networks. A cellular telephone system serves as a good example. The nodes in a cell-phone network are numerous, mobile, and have constantly-changing service requirements. Both the topology of connectivity and the local relationships between nodes are highly fluid. Yet, current cellular networks ignore this complexity as much as possible: all communications are managed by a static grid of monolithic servers (cell towers). Contemporary cellular systems are highly centralized.

All of the interesting dynamics in a cellular system revolve around the behavior of individual telephones, yet almost all of the investment, intelligence, and overhead in these systems is concentrated in the cell tower infrastructure, exterior to the phones themselves. As a result, cellular systems can only offer a very limited range of services, remain expensive to scale (with regards to both coverage area and device density), and make relatively poor use of bandwidth.

We suggest a different model of network design, a decentralized approach to managing networks. Instead of a centralized infrastructure for managing connectivity, we propose using populations of cooperating, mobile software agents to maintain routing information across dynamic networks. Our approach is to push the intelligence traditionally centralized in a few controlling nodes out into the network as a whole, and to embed that intelligence into a flexible, adaptive software framework.

2.2 Mobile Agents

Mobile agents are a novel way of building distributed software systems. Traditional distributed systems are built out of stationary programs that pass data back and forth

across a network. Mobile agents, by contrast, are programs that themselves move from node to node: the computation moves, not just the attendant data. Mobile agents in our work are defined by five important properties:

1. Agents encapsulate a thread of execution along with a bundle of code and data. Each agent runs independently of all others, is self-contained from a programmatic perspective, and preserves all of its state when it moves from one network node to another. This is "strong mobility" as defined in (Baumann, 1997)
2. Any agent can move easily across the network. The underlying infrastructure provides a language-level primitive that an agent can call to move itself to a neighboring node.
3. Agents must be small in size. Because there is some cost associated with hosting and transporting an agent, agents are designed to be as minimal as possible. Simple agents serve as building blocks for complex aggregate behavior.
4. An agent is able to cooperate with other agents in order to perform complex or dynamic tasks. Agents may read from and write to a shared block of memory on each node, and can use this facility both to coordinate with other agents executing on that node and to leave information behind for subsequent visitors.
5. An agent is able to identify and use resources specific to any node on which it finds itself. In the simulation presented in this chapter, the nodes are differentiated only by who their neighbors are (and agents do make use of this information). In a more heterogeneous network, certain nodes might have access to particular kinds of information – such as absolute location derived from a global positioning system receiver – that agents could leverage.

2.3 Flexible Systems

Using small, self-directed, mobile agents as building blocks allows us to design a network architecture that is flexible in several ways. First, because of the fundamentally distributed nature of collections of agents, our architecture can scale upwards in size quite gracefully. Second, because agent populations can change over time, new usage contexts and models can be accommodated. Finally, because all system interaction is mediated by agents, multiple network management strategies can coexist and co-evolve.

2.3.1 Network Size

Mobile agents serve as simple distributed building blocks for constructing system-level functionality. It is possible, at least in theory, to design a system using a small number of simple agents so that the interactions between agents is well-specified and so that the pattern of interactions between agents remains stable even as more blocks of the same kind are added. The development of design heuristics that are applicable to mobile agent systems is one long-term goal of our research.

One specific rule of thumb (or hypothesis about system design) that has emerged from our research is the following: explicit localization of interaction makes a networked system easier to understand and can reduce or eliminate performance bottlenecks and common points of failure. For example, individual agents in our model are simple and self-contained, and their interaction is restricted to local

manipulation of shared memory. The simplicity of our building blocks makes it easy to conceptualize systems in which large numbers of agents work together across a network of many nodes.

While an analysis of scalability is not the main focus of the experiments presented in this chapter, we have observed that the systems we design using restricted, localized communications models are more comprehensible and stable than those we have designed around more open, "sockets-and-messages" frameworks. It is worth noting that a common way of organizing a large networked system is to partition it into hierarchies smaller sub-networks. By contrast, our approach can be thought of as structuring communications using a large number of limited, overlapping spheres of activity, rather than a few discrete partitions.

2.3.2 Usage Models

It is very difficult to design a conventional network that adjusts well to either changing usage patterns over short time scales or to evolving needs and circumstances over the long haul. In contrast, a mobile agent based system can be reconfigured on the fly, in response to new situations or demands, and with or without human intervention.

System-level behavior can be incrementally altered by varying the size and composition of an agent population, adding new agents dynamically to reinforce weaknesses or balance priorities. Human managers can adjust the network, and agents themselves can adapt to changing circumstances. Well-written agents can be self-regulating, dying off or spawning copies of themselves as the situation dictates. Specific sets of agents can be delegated to monitor aspects of the system, altering the mix of other agents in the population, as needed. Finally, drastic revision of capability, such as patching a security hole or rolling out a new feature all across the network, can be accomplished by flushing the system and flooding it with a new population of agents.

To elaborate, populations of network agents dedicated to maintaining routing and connectivity information can specialize in several distinct ways:

1. Agents can specialize with regard to usage patterns across the network. For example, management agents could enact economic incentives to encourage certain traffic patterns (Gibney, 1998). Or specific agents might dedicate themselves to maintaining very low-latency routes, though they would need to use a disproportionate amount of bandwidth to do so.
2. Specialized agents could work to manage requirements in specific areas of a network. For example, a certain part of a LAN with a great deal of video traffic could deploy agents that construct and maintain high bandwidth connections.
3. Agents can adapt the network infrastructure to changing needs over time. For example, the cost of using a relatively dormant gateway could be reduced so that it handles more traffic. Or at times of peak traffic, the population could be weighted towards less-complex, low-overhead routing agents that mostly "harvest" connectivity information from the environment, rather than explicitly constructing connections (Poor, 1997).
4. Agents can specialize on behalf of specific users. An individual is likely to have a pretty good idea of his or her communications needs, and should be able to carry

around a collection of agents optimized to negotiate or create personal connectivity.

2.3.3 Network Management

As several of the above examples suggest, various management models integrate quite elegantly into systems built around mobile agents. For example, applying simple economic metrics to each agent's movement (or, more subtlely, thinking of information exchanges that enable communications as transactions) provides a decentralized framework within which particular kinds of communication can be encouraged, required, restricted or accounted for accurately.

Each agent must be given tools that allow it to make contextually-specific decisions about how to use finite resources. Cost-based routing algorithms, for example (Davie, 1996) are a well-understood way to optimize multi-hop traffic across a network. Other metaphors have also been proposed, such as the pheromone following of social insects (Schoonderwoerd, 1997) (Bonabeau, 1998) (Schoonderwoerd, 1999, chapter 13 of this book). The flexibility of a mobile agents architecture would allow multiple models such as these to coexist; perhaps messaging agents with different needs would route themselves using information from different management agents, and the owners of those messages would be billed accordingly by the owners of the various management ecologies.

2.4 Cognitive Tools For Systems Design

It is worth trying to separate the purely technical advantages of a system built out of mobile agents from the cognitive leverage that the mobile agents metaphor can provide to designers and programmers. Certain attributes – most notably the extreme run-time flexibility – of mobile-code architectures stand on their own as advances in systems capabilities. Most of the attributes described above, however, are as much arguments about systems designers as about systems design.

As networks become more and more complex, finding abstractions that allow us to think about them in useful ways becomes more and more important. Mobile agents, as discrete pieces of code with clearly-defined functionality and privileges, are an appropriate and powerful tool for use in thinking about, specifying, and writing programs for networks of computers.

The mobile agents approach can be thought of as a metaphoric extension of object-oriented programming, useful to engineers designing network systems in the same way that component software tools are useful to developers of complex desktop applications. A mobile agent encapsulates not only data and code, but an "agenda" – some intentionality, an unfolding thread of execution – into a small package. This encapsulation gives systems designers a way of creating building blocks for networks. With these building blocks it is possible to find ways of thinking about systemic behaviors that embraces changes in network size, congestion, usage patterns and user needs.

2.5 Related Work

Mobile agents are an active and exciting research topic, especially as contemporary tools such as Java make mobile agent systems relatively simple to implement. A number of general arguments exist as to why the mobile agent approach is potentially useful (Chess, 1997) (White, 1996). Much research has focused on performance enhancement from resource distribution, but we believe that the most interesting and novel possibilities of mobile agent architectures lie in their adaptive nature and inherent flexibility (Baldi, 1997) (Halls, 1997).

Much of the work applying mobile agents to systems infrastructure has taken place under the umbrella of active networking (Tennenhouse, 1997). Our work is closely related to that of the active networks researchers; we share a concern with the dynamic characteristics of network systems and a desire to explore the possibilities of movable bits of code in a network context. Our simple message agents behave much like active packets, using computational resources of each successive node on which they find themselves to choose a route.

There are several threads of research on using mobile agents situated in a telecommunications network to manage connectivity and load balancing. Appleby and Steward's work is an early paper suggesting using mobile agents with AI-like strategies to dynamically load-balance a telecommunication network (Appleby, 1994). Follow-up work using a design inspired by ant behavior (Bonabeau, 1998) extends these ideas in a new direction, using markers in the network environment as a means of indirect inter-agent communication ("stigmergy").

2.6 Why Not Mobile Agents?

A few potential disadvantages to designing networks with mobile software agents should be noted. We see this short list as a blueprint for future research rather than a set insurmountable obstacles.

2.6.1 Efficiency

Computational cycles are expensive and network bandwidth is precious. Messages which route themselves will consume more computational resources than statically-routed packets. And management and infrastructure agents will consume a share of bandwidth as well. Given this, it is perhaps difficult to justify these newer approaches as they seem more costly than those we use now.

However, contemporary engineering approaches already have difficulty dealing with dynamic networks (Perkins, 1997), and this will only become more apparent as our networks become increasingly complex and heterogeneous. Efficiency, from this perspective, is largely determined by how well an architecture deals with a broad range of ever-changing demands. A mobile agents approach offers a rich set of tools at design time and great flexibility during the *in situ* lifetime of a system, stacking the deck in favor of architects, implementers and maintainers faced with the difficult task of engineering reliable, dynamic networks.

Another way to think about network efficiency is as a set of necessary tradeoffs between finite resources. For example, because compute time has fallen in price more

rapidly than connect time it is common to trade cpu cycles for bandwidth by compressing data before sending it across a network and decompressing it upon receipt. Architectures in which packets route themselves across our current multi-hop networks are too expensive to see widespread use at the moment. But if cheaply available silicon continues to outpace fiber, the benefits of self-routing messaging agents will come to outweigh their diminishing relative costs. Similarly, if mobile management and infrastructure agents can do a good enough job maintaining dynamic connectivity in a wireless network, then such a system could prove more efficient than a rigid, centralized system that must waste a certain amount of bandwidth in order to provide a certain level of service. The infrastructure agents could be thought of as harvesting bandwidth, rather than simply consuming it!

2.6.2 Security

Second, moving bits of executing code from computer to computer raises a number of serious questions about security. We are very concerned with building secure systems, and are working to understand the particularities of securing our networks.

Mobile agents present three broad classes of security problems: protecting hosts from agents, protecting agents from hosts, and protecting agents from each other. The problem of protecting hosts from agents has attracted the most attention from researchers. We believe that standard cryptographic techniques (such as code signing) combined with sandbox- and permission-based models of security (such as those present in Java 1.2). (Oakes, 1998) are sufficient to design an execution environment that protects a computational host.

The other two problems of mobile agent security – protecting agents from their hosts and from each other – have enjoyed less attention. We are actively involved in research to protect agents from each other by extending existing host security models. And there is a growing body of research on protecting agents from hosts by executing encrypted code (Tschudin 1997). Although these constitute open areas of research, progress is being made. Furthermore, in some scenarios it may be possible to avoid these security problems entirely, by carefully delineating trusted terrain within a network.

Finally, it is worth noting that the fundamental difficulties we all face regarding information security are inherent in the spread of relatively open networks. As the advantages to having access to such networks generally outweigh the risks, we must all deal with questions of how to protect ourselves from both malicious attackers and poorly-written software, no matter what kinds of systems we design.

2.6.3 Provability

A third concern is that the “ecological” approach of building network behaviors from collections of cooperating agents is difficult to model mathematically, and as a result is difficult to reason about deterministically. We might call this the “unprovability” problem, and given how important contemporary software can be not only to productivity but to life and limb, it is a serious one. We typically do not want to build systems that are unreliable or unpredictable.

However, it should be noted that it is extremely difficult to reason about any real-world networked system, no matter what its design, because networks of computers

are irreducibly unreliable. Whenever it is possible that any participant in a network is potentially unreachable, is malicious, or is pathological (any of which are always possible), it becomes impossible to reason provably about the behavior of that system (Fischer, 1985) We believe that the mobile agents frameworks we are developing will become aids to reasoning about robustness and fault-tolerance, rather than additional impediments to such analysis.

The remainder of this chapter presents results obtained by simulating mobile agent populations within a wireless, peer to peer network. These results support the contention that using mobile agents to manage networks has merit, as well as provide a thorough analysis of one particular solution to an important network management problem, maintaining connectivity of routing maps.

3 Experimental Model

For our experiments we have defined a simple model of a dynamic wireless network and implemented a simulator to study that model. Our model is motivated by real-world wireless networks we are designing at the Media Lab, which consist of large numbers of small, low-power nodes scattered throughout our building. These nodes vary a great deal with respect to computational power, capabilities and usage – some are wearable computers, some environmental sensors or actuators, some are "gateways" attached to desktop computers, and some simply provide location-specific information to their peers. But they all share the characteristic that in order to reduce power consumption and maximize the use of a single channel they are limited to relatively short-range communication. This means that any message intended for a recipient outside the immediate vicinity will require multiple hops to reach its destination.

The simulation we describe here is perforce quite simplified compared to a real, hardware implementation of such a network. However, we believe our model is realistic enough to provide guidance for designing real systems, and that our experiments have general applicability to dynamically changing, decentralized networks. This chapter extends a previous set of experiments in using mobile agent populations to perform routing, reported in (Minar, 1998). The model presented here is more complex than that presented in our earlier work; we expect to be able to directly apply the lessons from this set of experiments in the construction of real dynamic, multi-hop, RF networks.

3.1 Nodes

The results we present derive from a simulation of a dynamic, peer to peer network. For our experiments, we chose a scenario consistent with related work by our colleagues at the Media Lab (Poor, 1997). The nodes in our system are conceived as low-power, relatively short-range, radio-frequency transceivers distributed throughout a two-dimensional space. In our simulation the total network diameter is roughly 20

hops, so nodes must cooperate in a peer to peer fashion to route packets across the network. The average packet in such a system requires multiple hops to travel from source to destination; therefore, resident mobile agents need to move around the network in order to effectively gather data about the whole system.

Individual nodes move slowly through simulated space, following random vectors, so that radio links form and break as the nodes move in and out of range of each other. As a result, the network topology is quite dynamic. We do assume, however, that physical links are reliable, bidirectional, and easily detectable. Every node knows who its neighbors are.

Each node in our system owns a simple routing table. Each node keeps a list of per-node routing information. For each other node in the world, the table stores what node to first send a packet to. So to route a packet to an arbitrary node, a message only needs to do a lookup in its current node for the destination, and then move to the neighbor that is indicated, repeating as necessary. It is important to note that these routing tables are not updated by the nodes themselves. The nodes are completely passive; they rely on mobile agents to update their tables.

We chose a network density to match our best estimates of the capabilities of low-power, single-channel RF devices, such as might be found in future personal accessories or as part of home wireless networks. Network size was chosen as large as was feasible for our data collection, 250 nodes. All of our experiments were performed with the same configuration and movement paths of nodes. A snapshot of the node placements at a typical time is presented in Figure 1.

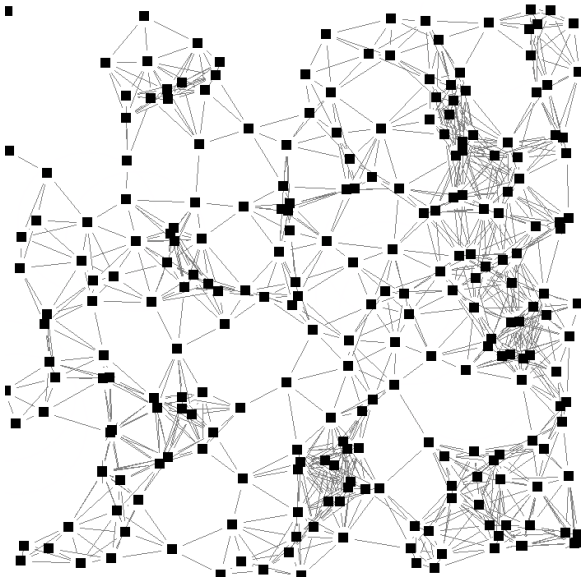


Figure 1: Snapshot of Simulated Network

3.2 Agents

In our system the nodes are “dumb”: they run no programs of their own, they simply host agents and provide a place to store a database of routing information. The mobile agents embody the “intelligence” in the system, moving from node to node and updating routing information as they go. Routing agents have one goal: to explore the network, updating every node they visit with what they have learned in their travels.

Routing agents discover edges in the network by traversing them. Each routing agent keeps a history of where it has been. When an agent lands on a node it uses the information in its history to update the routing table on its host as to what possible routes might be, by writing the best routes the agent knows about into the node's table. Each agent's memory for history information is quite small: our baseline is 25 entries. Because the agents must carry their histories with them as they move, the size of the history window is an important parameter: the longer the history, the higher the overhead of moving the agent. Our experiments investigate tradeoffs between history size and system performance.

The system as a whole relies on the cooperative behavior of a population of agents. The population size is an important parameter: the more routing agents, the higher the overhead. Agents in a population don't communicate directly with one another. The algorithms presented here don't even read information from the node's routing tables – they only write to them – making their own decisions about where to go based on their personal history (or, in the case of the random agent, on “whim”). Though system performance is dependent on the behavior of the all of the agents, the individual routing agents are blind to each other. The subject of inter-agent cooperation was treated in our previous paper describing this system (Minar, 1998), and we will return to this work for future experiments.

Our model is implemented as a simple, discrete event, time-step based simulation. Every step of simulated time an agent does three things. First, the agent looks at all the neighbors of the node it is on and makes a decision about to where to go next. Second, the agent moves itself to the new node, learning about the edge it travels. Third, it updates the routing table of the node it now occupies, using its own recent knowledge of the network.

We test two algorithms for how an agent chooses to move. One algorithm is a “random” agent that simply moves to a randomly chosen reachable node at each opportunity. This agent provides a base of comparison for more directed algorithms. We also tested an “oldest-node” agent that preferentially visits the adjacent node it last visited longest ago (or never visited, or doesn't remember visiting). This agent uses its history to try to avoid backtracking. Intuitively, it performs its task more efficiently by not repeating its own work.

3.3 Experimental measurements

Our simulation system consists of 1600 lines of Java code implementing a discrete event scheduler, a graphical view, a data-collection system, and the simulated objects themselves: network nodes and mobile agents. For our experiments, we repeatedly run

this simulator over various parameters. We measure system performance for the two different agent algorithms at varying history and population sizes.

Our measure of system performance is the average connectivity of the network after it has converged to its mean behavior. This number is calculated as follows. In our model, we designate 9 of the 250 nodes to be special "gateway" nodes. In a wireless network, these might be nodes that are wired to a larger network, maybe the local LAN or an Internet link. To measure connectivity we count the fraction of nodes in our system that have a valid route to at least one gateway. This measure is a reasonable aggregate of overall connectivity at any given time, as it models how many nodes have access to the "outside" world.

We run each of our iterations for 300 time steps. For all of our parameter settings, the simulation converges to its mean behavior at time 100 or well before. Therefore, we take as a measure of performance the average fraction of connectivity for all nodes from time 150 to 300. In addition, we report the maximum and minimum connectivity observed over that time period in order to convey an idea of the stability of the system. To factor out randomness in the initial placements of the agents, we report these numbers averaged over a set of 238 different runs of the same parameter set. All of our experiments were conducted with the same initial node placement and node movements.

4 Results

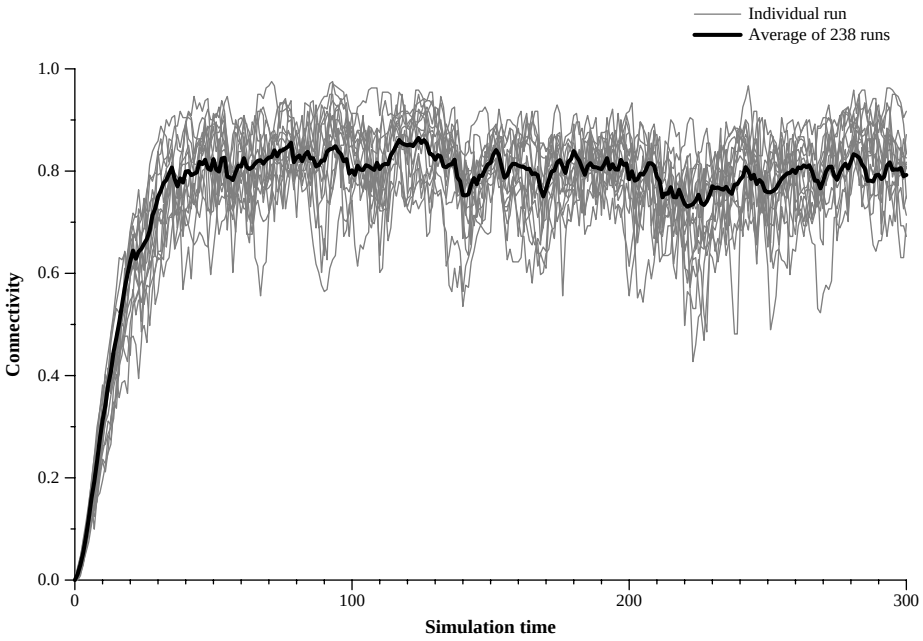
A summary of our results is printed in the appendix; what follows is a detailed description of our analysis.

4.1 First result: performance of a system over time

The first question we investigated is how does the connectivity of the network change during the course of a single simulation. We examined our baseline system – 100 oldest node agents with history length 25 – measuring the connectivity of nodes at each time step. The time series in Figure 2 shows the result: the heavy line is the average time series over all 238 runs, grey lines are a randomly selected set of 16 individual runs.

As the reader can see, the system starts out at zero connectivity: the initial condition is no knowledge. However, the network quickly gains connectivity as the agents move around collecting information. There is a small hiccough around time 20 (presumably the result of a major change in graph topology) but the agents quickly work around that and rise up to 0.8 connectivity by time 40. From then until the end of our simulation at time 300, the connectivity fluctuates around 0.8 as the network changes dynamically, and the agents adjust the routing tables accordingly.

We define the convergence point for the purposes of further measurements to be time 150; this is well after the time series of connectivity flattens out for all of our runs. The mean connectivity from time 150 to 300, then, is calculated to be roughly 0.792 – we take this mean as the characteristic performance of the system consisting



Connectivity for system over time. 100 Oldest Node agents, history size 25.

Figure 2

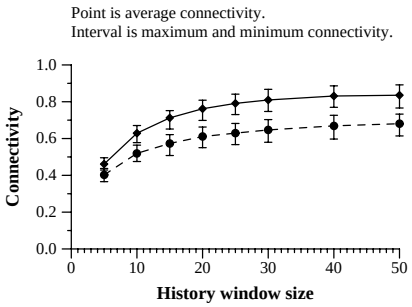
of 100 oldest-node-agents with a history size of 25. In addition, the average connectivity remained between 0.73 and 0.84 during that time span; this is the performance interval of this system.

This particular example is representative of our time series data in general. All of our runs start with 0 connectivity, quickly climb to some characteristic equilibrium, and then remain there until the end of the run. Therefore, in the rest of this chapter we will report simply the final mean connectivity number, along with maximum and minimum connectivity values registered after convergence.

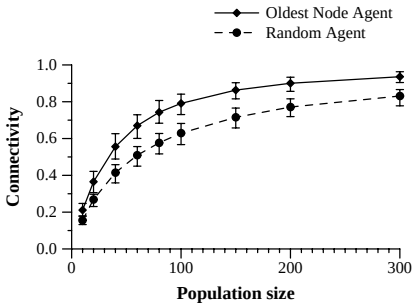
There are two significant results in this first experiment. First, it demonstrates a basic concept: a population of agents can maintain a reasonable rate of connectivity (in this case, 80%) across the simulated network. Second, the population maintains this connectivity rate with reasonable stability. The system converges quickly, after which there is some fluctuation in connectivity but never large swings.

4.2 Analysis of agent algorithm and parameters

With a basic framework for analyzing the behavior of our system in place, we can explore the effects of various choices for system parameters on performance. We alter three main variables independently: the number of agents, the history size of each agent, and the type of routing agents used (random or oldest node). We varied these



Connectivity for different history sizes.
Population size 100 agents.



Connectivity for different population sizes.
History size 25 hops.

Figure 3

results around a baseline of 100 agents with history size 25. The results are presented in Figure 3.

4.2.1 Agent type

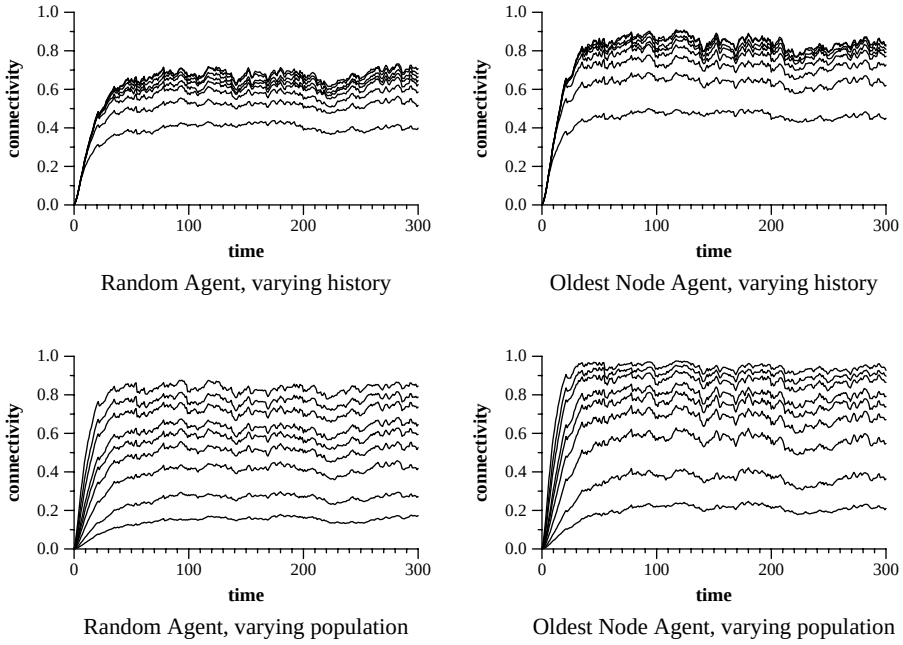
The most apparent result is the effect of agent type on the performance of the system. In both graphs and at every parameter setting, oldest node agents perform better than random agents. This should come as no surprise: the oldest node agents are able to search the network more effectively by minimizing back-tracking.

However, it is interesting to see that often the random agent does not do as much worse than the oldest-node agent as we might expect. For example, in our baseline of 100 agents with a history size of 25, the average connectivity for the oldest-node agent system is 0.792, while the average connectivity for random agents is 0.646. Though a 23% gain in performance is palpable, random agents might still be preferable for some network situations. In particular, the random algorithm might reduce the computational load on each node in the system. Whether or not this load is important depends on a variety of factors, not least of which is how quickly the network topology changes out from underneath the agents. An algorithm that makes a careful choice about where to visit is much less useful relative to a random choice if the links between nodes are no longer what the agent thinks believes them to be.

4.2.2 History size

The second parameter we studied is the effect of agent history size on system performance. Reading the history graph in Figure 3 from left to right, we see that for both types of algorithms having more history is better. This is to be expected: the agents have longer memories, and so provide more data to the nodes and (in the case of the oldest node agent) operate more efficiently.

However, the performance improvements diminish as the history size grows bigger. Intuitively, it is always better for agents to have longer memories. But at some point those memories are either so old as to be out of date and useless, or else redundant across multiple agents. In addition, larger history sizes still have high



Random Agent, varying history | Oldest Node Agent, varying history

Random Agent, varying population | Oldest Node Agent, varying population

Figure 4

spreads between maximum and minimum connectivity. Adding more memory to agents does make the system perform better on average, but has little effect on stability. Again, the movement of nodes explains this characteristic. Long agent memories allow information that remains valid for several time steps to be widely-distributed, but do not help the system recover from large topological changes.

4.2.3 Population size

Finally, we also looked at the effect of population size on system performance. As with history size, we found that having more agents is better. This result is unsurprising, since a larger population means there are more agents to look for routes. And as with history size, the gain from adding more agents declines as the population size gets larger, presumably because agents begin to duplicate each other's work. However, more agents also narrows the spread between maximum and minimum connectivity; having some often-redundant agents can help lend stability to the network. In other words, distributing the workload across the network (so, necessarily, across many agents) reduces the time it takes for information about newly-formed routes to propagate outwards.

4.2.4 Summary of basic system analysis

A different view of the above data is presented in the four time series graphs in Figure 4. Each graph represents the effect of changing one variable (either history size or population size) while holding the others (agent algorithm, and population size or history size) fixed. The time series data confirms the summary presented above; a

larger population is better, agents with larger memories perform better, and oldest node agents are better than random agents.

This series of experiments confirms our basic expectations of the model. An analysis of the overhead of these agent populations, to establish some basis for determining optimum parameters in this kind of system, remains.

4.3 Overhead analysis

The results presented above validate our model, but the basic conclusions are a bit dull. Of course more agents or more memory is better, we expect that! But, in the real world, these attributes come at some cost. The more agents there are in the system, or the more memory the agents have, the more overhead there will be for the network to support the routing agent population. A sensible analysis must account for agent overhead when measuring system performance.

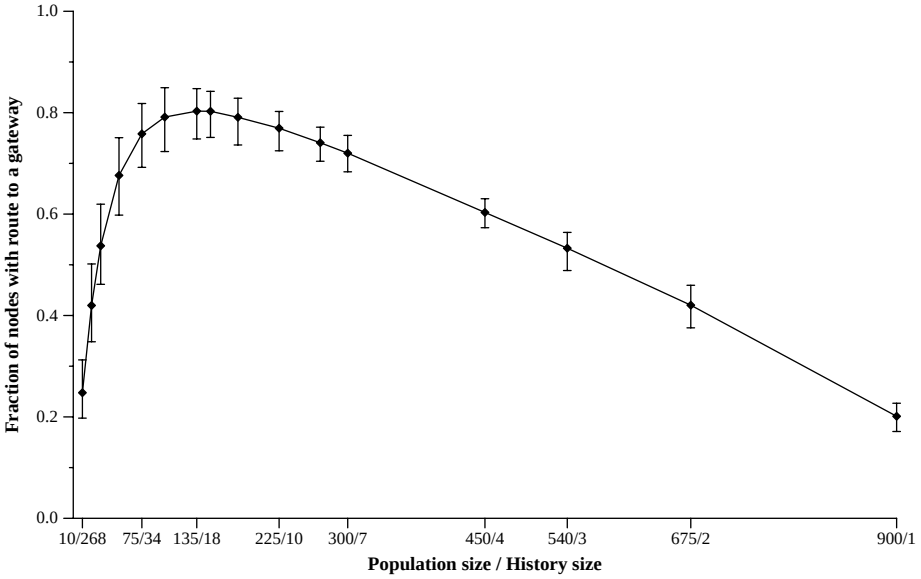
In order to specify the overhead characteristic of an agent population, we make some rough estimates on the cost of transmitting an agent. If nodes have a 48 bit ID, time stamps are kept as 64 bit quantities, and hop counts are kept as 16 bit quantities, then for every node an agent remembers it needs to keep 48 + 64 + 16 = 128 bits of data in its history table. Therefore, the cost for a particular agent is 128 * h, where h is the history size. Furthermore, we estimate that the cost of sending the agent itself is roughly 256 bits – 128 bits to cover a digital signature on the agent's contents, and 128 bits of data for the agent owner field, the agent type, and other bookkeeping data. (The size of the agent's code is not factored in, because in these homogeneous systems it would be sensible to cache the agent's code. In more open systems where there are many different agent species, the agent's code size will become an important factor.)

Taking these estimates together, the overhead of the system is defined by the equation $O = N * (128 * h + 256)$ It is important to note that the particular values of these constraints are somewhat arbitrary. However, the general form of the overhead measurement is important: a simple linear function dominated by the product of the number of agents and their history size.

With this overhead calculation in place, it is possible to compare different systems with equal overhead. In our baseline, the overhead for the entire population of 100 agents with history 25 is 100 * (128 * 25 + 256) = 345600 bits/step. Many other parameter settings also produce the same overhead: for example, a system with 30 agents of history size 88 also has an overhead of 345600, as does a system with 450 agents of history size 4.

Given the fixed overhead cost of 345600, what is the optimal tradeoff between number of agents and history size? Figure 5 shows the performance of a system of oldest node agents at varying settings with constant overhead. This graph shows that for our experimental parameters, a system with 135 agents of history size 18 is optimal.

We do not wish to argue for the significance of these specific numbers; they are dependent on the particular parameters we chose for the world. However, we believe these results are suggestive of the performance of these and similar algorithms across characteristic multi-hop systems.



Average connectivity for oldest node Agent
Varying history size and population size with constant overhead.
Interval represents max and min connectivity, point is average.

Figure 5

In particular, our results show that with this kind of system one has to strike a balance between having a large number of agents and having each agent carry around a large amount of data. Putting too few agents into the system causes unacceptable performance degradation, but there is a bit more leeway in giving individual agents shorter memories. Intuitively, this implies that for routing on a dynamic network, a decentralized approach using a great many individual autonomous agents is more effective than a more-centralized solution with a few very smart agents.

5 Conclusions & Future Work

This chapter has presented an approach to managing service in a dynamic network. The particular network we have chosen to simulate, consisting of a large number of highly-mobile nodes with relatively short-range transceivers, is one example of a system that is poorly served by traditional, centralized architectures. We have attempted to show that a population of mobile, cooperating software agents are able to build and maintain routing tables in such a system, and that these tables remain reasonably accurate even as the topology of the network changes over time.

At a more general level, we have described a framework for the design and management of unruly networks. Our framework is based on the idea of an ecology of mobile agents living inside a network, both serving as and servicing infrastructure. This approach has promise as an inherently decentralized technique, appropriate for

managing networks which are themselves inherently decentralized. To borrow the language of electrical engineering, there is no “impedance mismatch” between system and software. The result is a design theory that is aesthetically appealing, conceptually clean, and significantly flexible.

This work is part of a broader research program focusing on the development of infrastructure for complex, large-scale heterogeneous networks. We are just beginning to understand the possibilities and implications of systems built from mobile code. Much work remains to be done in several important areas: provisions for security and safety across networks; tools for design, modeling and inspection; methods for analysis of resource tradeoffs, system overhead and protocol strengths and weaknesses; and application-level frameworks that allow new uses of networked devices.

Our next series of experiments will study the design of specialty sub-populations of agents that manage particular routing tasks. In addition, there are a number of parameters that are held constant in the particular system presented here, but that we would like to explore, such as the speed with which nodes move, the density of nodes, the size of transmit/receive radii, and the assumption of link reliability. Incorporating these variables will increase the complexity and realism of our simulation, giving us a better picture of the demands upon such networks and the possibilities of the mobile code approach. Finally, we are in the process of building a real-world implementation of the short-range, low-power RF network modeled in this chapter, in order to apply the lessons learned so far, and to test our hypotheses outside the safe confines of simulation.

Bibliography

Appleby S., Steward, S. (1994) 'Mobile software agents for control in telecommunications networks.' BT Technology Journal, Vol. 12. No. 2. April 1994. pp.104-113. Chapter 11 of this book.

Mario B., et al. (1997) 'Exploiting code mobility in decentralized and flexible network management.' Proceedings, First International Workshop on Mobile Agents, Berlin, April 1997. http://www.polito.it/~picco/papers/ma97.ps.gz

Baumann, J. (1997) 'Mobility in the mobile-agent-system Mole.' CaberNet: 3rd Plenary Workshop, 1997. http://www.informatik.uni-stuttgart.de/ipvr/vs/Publications/Publications.html#1997-baumann-05

Bonabeau E., et. al. (1998) 'Routing in telecommunications networks with "smart" ant-like agents.' Santa Fe Institute Publications. January, 1998. Submitted to: Intelligent Agents for Telecommunications Applications '98.

Di Caro, G., Dorigo, M. (1998) 'Mobile agents for adaptive routing.' Proceedings: Thirty-first Hawaii International Conference on Systems, January 1998. ftp://iridia.ulb.ac.be/pub/dorigo/conferences/IC.22-HICSS31.ps.gz

Chess, D., et al. (1997) 'Mobile agents: are they a good idea?' In *Mobile Object Systems: Towards the Programmable Internet*, volume 1222 of *Lecture Notes in Computer Science*. 1997. http://www.research.ibm.com/massive/mobag.ps

Davie, B., Peterson, L. (1996) *Computer Networks: A Systems Approach*. ISBN 1558603689.

Fischer, M., et al. (1995) 'Impossibility of distributed computing with one faulty process.' Journal of the ACM 32:2 (April 1985): 374-382.

Garijo, M. et al. (1996) 'A multi-agent system for cooperative network-fault management.' Proceedings: First International Conference and Exhibition on the practical applications of intelligent agents and multi-agent technology, pages 279-294, London.

Gibney, M., Jennings, N. (1998) 'Dynamic resource allocation by market-based routing in telecommunications networks.' To appear in: Intelligent Agents for Telecommunications Applications 98. http://www.elec.qmw.ac.uk/dai/projects/agentCAC/IATA.pdf

Halls, D. (1997) 'Applying Mobile Code to Distributed Systems.' PhD thesis, Computer Laboratory, University of Cambridge, June 1997. http://www.cl.cam.ac.uk/users/dah28/

Huberman, B. (1990) 'The performance of cooperative processes.' Physica D, 42:38-47, 1990.

Minar, N., et al. (1998) 'Cooperating mobile agents for mapping networks.' Proceedings: First Hungarian Conference on Agent Based Computation. 1998. http://www.media.mit.edu/~nelson/research/routes-coopagents/

Oakes, S. (1998) *Java Security*. ISBN 1565924037.

Perkins, C., Woolf, B. (1997) *Mobile IP: Design Principles and Practices*. Addison-Wesley Wireless Communications Series. ISBN 0201634694

Poor, R. (1997) 'Hyphos – A Self-Organizing, Wireless Network.' Master's thesis, MIT Media Lab, 1997. http://ttt.media.mit.edu/pia/Research/Hyphos/

Rosenschein, J., Zlotkin, G. (1994) *Rules of Encounter*. ISBN 0262181592

Schoonderwoerd, R., et al. (1997) 'Ant-based load balancing in telecommunications networks.' Adaptive Behavior, 5(2):169-207, 1997. http://www-uk.hpl.hp.com/people/ruud/abc.html

Schoonderwoerd, R, Holland, O. (1999) 'Minimal agents for communications networks routing: The social insect paradigm' Chapter 13 of this book.

Somers. F. (1996) 'Intelligent agents for high-speed network management.' Proceedings: First International Conference and Exhibition on the practical applications of intelligent agents and multi-agent technology, pages 909-921. London.

Sycara, K (1989). ‘Multi-agent compromise via negotiation.’ Distributed Artificial Intelligence, volume 2. ISBN 1558600922.

Tennenhouse, D., et al. (1997) ‘A survey of active network research.’ IEEE Communications Magazine, 35(1):80-86, January 1997. http://www.tns.lcs.mit.edu/publications/ieeecomms97.html

Tschudin, C. (1997) ‘Protecting mobile agents against malicious hosts.’ *Mobile Agents and Security (Lecture Notes in Computer Science, 1419)* ISBN 3540647929 http://www.icsi.berkeley.edu/~tschudin/

White, J. (1996) ‘Telescript technology: Mobile agents.’ *Software Agents*. ISBN 0262522349.