DESIGNING AN ECOLOGY OF DISTRIBUTED AGENTS

Master's Thesis Proposal
November 5, 1997
Revised December 1, 1997

Nelson Minar
<nelson@media.mit.edu>
http://www.media.mit.edu/~nelson/

Pattie Maes
Associate Professor of Media Arts and Sciences
MIT Media Lab

Mitchel Resnick
Associate Professor of Media Arts and Sciences
MIT Media Lab

Judith Donath
Assistant Professor of Media Arts and Sciences
MIT Media Lab

Contents

Abstract

Computers have become commonly connected to the Internet. Ubiquitous networking will enable powerful new types of computational systems, but software designers do not currently have good methods for building large distributed computer systems. In this thesis *ecologies of distributed agents* will be explored as a way of designing and understanding programs for complex networked systems. An application will be built using an ecology of agents to represent the presence and activity of people on the Internet.

1 Introduction

Unnetworked computers are an anachronism. The Internet has radically changed our understanding of computation; computers used to be calculating machines, boxes sitting alone waiting for a user to come along and use their functions. But now a computer is much more than a number tool. It is a communication device, a gateway into an environment of data, services, other computers and other people. Similarly, computers are no longer passive objects waiting to be told what to do. They are active peers in a network, ready to respond to requests from other computers and initiating requests on their own.

But while the uses of computers have expanded with the Internet, the way we design computer systems has largely remained the same. The programs we use daily are still mostly designed to be single processes running on one machine, maybe fetching a thin stream of data through a network interface but otherwise largely independent of the vast sea of computation out on the Internet. The World Wide Web expanded the realm of data from isolated document stores to an interconnected, distributed mesh of information. A similar transition is happening for computation; computer processes will increasingly occur *in* the Internet, not just make use of it. Programs on the network will interact in complex and sophisticated ways.

The Internet is not just a network in which computer programs operate, it is a social environment, a place where people are present every day carrying out their activities. The application of this thesis research is to develop a distributed system to represent the presence of people in the Internet, to reveal the social aspects of the network. The application will be built with a new design paradigm, an ecology of distributed agents that work together to make people's presence on the Internet visible. In addition to building a useful application, this thesis research will explore the metaphor of computational systems as complex ecologies of computer programs.

1.1 Ecologies of Programs

Distributed computation is not a new research topic. But most of the work in distributed programming has been from a very particular point of view; systems where a single designer creates programs that interact in precisely predetermined ways. While such top-down engineering practices are valued because they give predictable results with exact control, the resulting systems are often brittle, inflexible, and limited in scale.

Another way to approach distributed computation is to consider the communication of programs on networks as analogous to the interaction between organisms in an ecosystem. An ecological metaphor can be a useful way to understand what happens on computer networks. For example, the interaction of computer viruses and infected computers is understood by analogy to real viruses infecting hosts. The very name "computer virus" is an ecological metaphor. Andrew Leonard's book *Bots: The Origin of New Species* [13] takes ecological metaphors further, presenting a phenomenology of the behavior of active computer programs ("bots") in such diverse environments as MUDs, IRC, and the Web. In Leonard's narrative, bots interact in complex ways reminiscent of natural ecology: fighting, stealing, cooperating. And ecology is not just a metaphor for understanding existing systems: Tom Ray's Network Tierra system [17] creates a computational ecology by building an artificial environment for computer programs to live in, breed, grow, die, and evolve.

An ecological metaphor carries with it a new understanding of computation. A natural ecology is the sum total of the interaction of many organisms; a computational ecology is the result of the interaction of many independent programs. Ecosystem interactions can take many forms; competition, cooperation, symbiosis. In addition, ecology implies the idea of *diversity*, many different species of programs coexisting. Instead of being limited to systems that a team of engineers can design and fully understand, a computational ecology is a system that is open-ended, capable of novelty and surprise.

Interactions between programs on the Internet are often complicated precisely because of the diversity and open-endedness of the network. New programs are created every day that interact with other programs in ways that their authors never intend or understand. At its best, in the Internet ecology computer programs interact in synergistic ways. For example, the interaction between Web spiders and Web servers has made it possible to search an unimaginable amount of information without the spider authors ever directly coordinating with webmasters. But our intuitions about the complex interactions between programs are poor; complex network systems often fail and software designers often miss opportunities for beneficial interactions between disparate programs.

How do we improve our understanding of ecologies of computation? By experimenting with them, by building systems that deliberately embrace the power and complexity of ecological interactions between programs. The core research for this thesis is to make an ecology of computer programs manifest, to build a system where agents travel around the Internet, visit people's machines, and act in interesting and useful ways. The agents themselves have complex interactions with each other and suffer life-or-death selection at the hands of users. By building a computational ecosystem I will explore the ecological metaphor and study the methods it inspires for designing and understanding complex distributed programs.

1.2 Distributed Agents

In order to build a computational ecology a technology is needed to support programs living natively in a network environment. The key requirement is to integrate communication into the computational framework, to make it as simple and natural for programs to work over the network as they operate on a single machine. One promising paradigm for networked computation is *distributed software agents*: programs that are autonomous, network aware, and (potentially) move from machine to machine. Distributed agents open up new possibilities for the way that we design computational systems.

The term "agent" is used to mean many different things; for the purpose of this thesis, the idea of autonomy is the most important part of the definition of a software agent. Briefly, an autonomous agent is a program that has sensors, actuators, and its own goals and purposes. Software agents are capable of independently acting in their environment. Agents are a natural component of computational ecologies — they are the computational analogy of biological organisms.

Natural ecologies have complex natural environments; the environment for a computational ecology is the network itself. *Distributed* agents are agents that are native to their particular environment, the network. Part of being Internet-native is being able to easily talk to other agents across the network. In particular, distributed agents require high-level interagent communication in order to make it easy for agents to find each other and meaningfully interact.

One particular distributed agent facility, mobile agents [5] [21], is also an important part of a computational ecology. Mobile agents can move themselves across the network, picking up from one machine and moving to another in order to get better access to resources. Mobility for natural organisms is common but not required — there are animals that move, but there are also stationary plants. Similarly, some (but not all) agents living in networks can benefit from being able navigate the environment, from moving from computer to computer.

Mobility and distribution are clearly needed if we are to remain true to the ecological metaphor; we do not want an ecosystem consisting entirely of isolated plants. But there are many questions about the usefulness of distributed and mobile agent systems. The foremost criticism of distributed agents is the complexity of the software technology. Building distributed object oriented systems is difficult; until Java there were not many widespread tools for programming network-native applications. Fortunately, better tools for building distributed systems are becoming available.

Mobility is a more controversial technology. There are few (if any) examples today of an application built with mobile agents that could not also be built by some simpler, better understood technology. But mobile agents are a powerful way of designing solutions for problems in some scenarios including limited bandwidth, unreliable networks, and itinerant computing. And most importantly, mobile agents make computers more flexible — with a system based on mobile code, servers can be extended with unforeseen functionality by simply sending in a new agent to provide a new service. One conclusion is that even if mobile agents do not do anything new, they do provide better ways to do the things networked systems need [9].

1.3 Visualizing Activity and Presence in the Network

To give focus to the research in ecologies of distributed agents, this thesis will concentrate on building one particular application — representing the presence and activity of people online. This topic has been popular both in practice and in research. Basic systems have been reinvented many times, from the BSD finger protocol to IRC and the recently popular ICQ system [19]. There is also a large amount of research in representing activity and presence more expressively; Judith Donath's Ph.D. research at the Media Lab, for example, in particular the Visual Who system [7].

A natural way to build a system that represents people online is to write distributed software agents to collect and display information about people's activities. Agents in this application can serve several roles. They can themselves represent people, either graphically or through other modalities. They can also mediate communication of information between computers; people can have agents living on their computer that manage their own data profile and communicate it to other agents for display.

There are many reasons an ecology of distributed agents is attractive for this application. Presence is inherently a distributed application; distributed agents can naturally manage communication between users' computers. In addition, there are many different ways to represent people. An ecology of agents allows for a diversity of different approaches. Users are not limited to any fixed set of possible representations: mobile agents are free to come and go between machines, bringing in new types of display and functionality. Users can make personal choices about which agents they like. The ecology as a whole is a complex expression of how well individual agents are matching people's desires.

1.4 Other Applications of Computational Ecologies

At the MIT Media Lab there are several projects underway that are related to distributed agents. Kwin Kramer's "Tiles" thesis proposal and Tom White's "Buzz" project both have at their core the idea of programs that live in and move around a computer network. The interaction between the programs and the network environment is what makes them interesting.

The applications of ecologies are not limited to conventional computers on the Internet. Distributed agents could also be a natural way to coordinate the various computational devices being designed by participants in the Things That Think consortium. Individual TTT devices might not have much computational power, but agents that live in the environment of many TTT devices networked together could make great use of the diverse resources available.

1.5 Summary of Expected Contributions

These various projects are only hints at what distributed agents can do. A major goal of this thesis is to explore distributed agent technologies, including mobile agents, to try to understand what these techniques are best applied to. Distributed agents are a natural way to design a computational ecology. Furthermore, distributed agents are interesting independent of an ecological metaphor; they are a powerful way to describe and design computational systems. An ecology of agents is particularly appropriate for designing an application that represents presence online. It is time to experiment with distributed agents, to design specific systems with this paradigm and see what is possible, what is easy, and what is hard.

The core research contribution of this work will come from the experience of building and understanding of an ecology of agents. The process of creating the system itself will provide valuable insight into designing systems with distributed agents, in particular what languages, tools, and technologies are appropriate for designing and understanding ecologies of distributed agents. Once the basic system is built, an iterative process of implementation, use, and evaluation will yield an understanding of how complex software ecologies operate and change. In summary, this thesis research will make contributions in the following areas:

- **Visualizing Activity and Presence**: an application that improves the social experience of the Internet by making people aware of each other's presence.
- **Distributed Agents**: research in creating distributed software systems with software agents.
- **Ecologies of Programs**: results of exploring the metaphor of computer systems as complex ecologies of interacting agents.

2 Application Scenario

To represent presence, an application will be built on a framework that allows an ecology of agents to live on users' computers on the Internet. Participants run a server on their machine that will host a local population of agents. Each server has a window open on the user's screen; agents render themselves in the window to display information about the activities of other people. Agents communicate with each other to obtain information, across the network if necessary, and also move and copy themselves from machine to machine. Users express their preferences about agents by rewarding or punishing individual agents as they see fit. The system will be designed to work for as many users as can be recruited; for the purposes of this thesis research the network will consist of about fifteen users with a total of roughly two hundred agents of twenty different species.

2.1 Server Environments

The primary component of the system is a distribute agent server, a program that resides on each user's computer and accepts agents. The servers are the environment of the ecology — they are the homes for the agents, the world they experience. Servers control the senses of the agents, the effects the agents can have, and the topology of the environment. These properties are made specific in the API that the server exposes to the agents for acquiring data, being visible to users, and moving themselves to other servers. The agents themselves are simply small programs that are written to the server API.

The server provides the agents with access to various kinds of data. It defines what agents are capable of knowing. The server will provide data about whether a user is present at the moment, how busy he is, and what he is doing. The server will also tell agents information about the status of the server itself; how busy the machine and the local network are, what other agents are present. This information about machine and user activity will give agents a rich data stream as input.

Agents take in data from their sensors, process it, and present the information to the user. A variety of actuators for agents can be imagined. For the purposes of this thesis the output available to the agent will be quite simple — just access to a window and graphic drawing commands. The system could be expanded to allow all sorts of other interesting output modalities such as sound or ambient displays, but that is a topic outside the scope of this thesis.

The world of the agents is the network of interconnected servers. An agent initially will only know about other agents on its home machine. To learn about agents on another machine, an agent must first either move itself or send out a messenger. Once two agents have met, they can move away and continue contact over the Internet. Agent mobility is limited by the servers; agents can only move to a server's neighbors. Limited communications brings structure to the network and will hopefully induce interesting agent dynamics.

2.2 Agents Representing Presence and Activity

The system will contain three kinds of agents. The most visible genus of agents will be the representation agents, those that present presence data with animations and interpretive graphics. Multiple species of display agents will live in the network, embodying a diversity of different visualization strategies. A second type of agents are informational agents: programs that collect and process data and make it available to representation agents. Informational agents broker information about users, such as whether someone is online, how busy she is, what she is currently working on, her interaction with the Web. A third genus of agents will act as messengers, moving from machine to machine to create communication links between informational agents and display agents.

All of these types of agents work together, cooperating to provide representations of presence and activity. Agents will communicate via simple Internet sockets. For example, a representation agent could have the goal of telling me how busy my friend is at her keyboard. This agent would first send out a messenger agent to my friend's computer to locate an informational agent there that will provide data about my friend's keyboard activity. The messenger would then return home and introduce the display agent to the informational agent. Contact established, the representation agent can now render my friend's keystroke data with a graphical display. The agent network as a whole will be a complicated system of agents producing and consuming information. While agents will often cooperate to fulfill a task, they will also be competing for the the right to live on machines. Agents live or die on the whims of users.

2.3 User Interaction

People are a part of this artificial ecosystem, both actively and passively. Passive participation is implicit in the design — because the agents themselves are active, a person can use the system by simply sitting back and watching what the agents are doing. Agents will come into the server without a user's deliberate request and will constantly be sensing data.

However users will also actively affect the ecology by expressing their preferences about individual agents. For example, a user of the system might decide that a new representational agent that just came in is annoying or is wasting CPU time providing a function the user is not interested in. In this case the user could express disapproval by simply killing the agent off his server. Similarly, if a representation agent is well-liked by a user it can be rewarded, allowed to survive on the user's system and spawn children on neighboring computers. Users will be explicitly aware of informational agents that are collecting data on their machines and messenger agents traveling through; if an agent is revealing too much information or is taking up too much bandwidth, the user can kill it. Agents will compete for the acceptance of users; user feedback will create a selective pressure on the agent ecology, thereby shaping the global population according to people's tastes.

The system will also be open-ended. Motivated users can write their own agents and load them on servers, adding new inhabitants into the ecosystem. Time permitting, a simple genetic algorithm will be implemented so that new subspecies of agents can arise and evolve spontaneously.

2.4 Summary of Application Scenario

A constantly active agent ecology will operate to make people's presence on the Internet visible. The ecological metaphor will create a system that is open-ended and adaptable, allowing a diversity of agents to exist while also giving users the ability to express their individual preferences. The agents themselves will live in a complex environment, constantly seeking out new people to impress, new data to acquire, new machines to live on. The system will be both entertaining and useful; the process of building and studying it will give insights into several important research areas. Creating the system will require coordinating three things:

- **Distributed Agent Servers** to provide the environment for agents
- **Agents** to represent the activity of people, collect data, mediate communication between other agents.
- **Users** to serve as data sources and observe, reward, and punish agents.

3 Research Activities

The major research activities for this thesis work fall into three categories. The fundamental work will be in building a system that enables an ecology of distributed agents. On top of this framework the application itself will be built, a set of agents that coordinate to represent people's presence on the Internet. Once the system is built and deployed, the appropriateness of the ecological metaphor will be studied by examining how elegantly the system is constructed and how well it operates.

3.1 Building an Ecology of Distributed Agents

Designing an infrastructure for a computational ecology is the major research activity for this thesis. Distributed agents are an appropriate technology for the task but tools for building distributed agent systems are relatively new. Fortunately, Java 1.1 provides many of the basic functions. On top of the Java core, several third party packages such as IBM's aglets [12], CalTech's Infospheres

[3] [4], and ObjectSpace's Voyager [20] system provide higher level distributed object and mobile agent facilities. In particular, they package up details of naming objects, handling network class loading, and presenting a coherent distributed objects API. An early requirement for the thesis project is evaluating the various architectures and choosing the most appropriate for building ecologies. Voyager appears to be the most well-suited platform, although I intend to stay in close contact with Caltech's Infospheres research group as well.

Existing software will provide some of the base functionality needed for distributed agents but a lot will need to be done to build an ecology on top of the core. Current distributed object systems do not have a very strong notion of the *environment* that agents have access to. There is little in the way of a well-defined limited API for incoming agents and nothing as high level as the sensors and actuators that the agents inhabiting the thesis system will need. Defining an environment interface is a major research task; the results will have broader applicability towards any systems that want to define a service API for mobile agents.

There is also the task of defining the structure of the network environment as a whole — what servers are available, how an agent can find and navigate them. The preliminary plan is to have a simple locally connected topology; each server is connected to only a few others, but the system as a whole will be a connected graph. The suitability of this topology for an artificial ecosystem is an open question. Time permitting, the issues of global naming (how to construct a reference to an agent that is valid anywhere) will also be studied.

Agent intercommunication is essential for the system to work; agents have to share information with each other to fulfill the application. Coordinating agent interaction will touch on many deep research issues, including questions of ontology and credit assignment. A description of agent services will be needed so that when two agents meet they can quickly learn what capabilities each other has. Defining ontologies for software is a major research topic in its own right; for this thesis, an ad hoc solution will be created possibly using XML [2]. A related issue is the credit assignment problem — if one agent is rewarded or punished by a user, the agents that it communicates with should also receive part of the credit or blame. Credit assignment in distributed systems is also a major research topic; for the purpose of this thesis only a very basic system will be implemented.

An optional topic of research is the design of a security manager to ensure that agents do not escape the constructed environment. Access controls are an important topic for all mobile computation; security of the servers needs to be ensured. Java provides a foundation for enforcing security policy, but higher level work is needed to make security management simple and practical for this kind of application. In addition, while protecting servers from agents is relatively well understood there is little research in the problem of protecting agents from each other. If time permits a security manager for agent ecologies would make a nice side project, but it is not an essential part of the thesis.

3.2 Presenting Activity with Distributed Agents

Once the environment is created, the next step is to build agents to populate it. While much of my effort will necessarily focus on building and understanding the infrastructure, the application of representing presence is itself a major research topic that this thesis will address.

The power of an agent ecosystem is that multiple solutions to a problem can be implemented. The ecosystem will be populated with a variety of agents that show the activities of people on the network. In order to design visualization agents, I will rely on my graphical design experience from John Maeda's MAS course "Principles of Visual Interface Design," [14] along with consulting existing research in representation of presence. Several species of data providing providing agents will be created to provide representation agents with data from various sources. Coarse data such as a user's activity at the keyboard will be provided, as well as fine detail about specific activities such as what Web page a user is looking at. The goal is to build enough agents to have many alternatives for addressing the problem of making the Internet a more social space.

3.3 Exploration of the Ecological Metaphor

Why use an ecology of agents to represent presence? The application is fundamentally distributed: agent ecologies are a powerful way to build distributed applications. A major evaluation of this thesis research will be a critique of the design of the implemented application. How hard is it to build a system of cooperating distributed agents? Is this a natural way to build applications, does it afford greater power and flexibility as predicted? The experience of building a practical application will enable me to at least partially answer these questions.

Another reason to build a presence application with an ecology of agents is to give users a diverse population of solutions, to have a variety of different ways to represent presence. The reward and punishment of agents gives a user the ability to choose individual agents for himself and influence the entire network population. One goal for the thesis is to begin to understand the phenomenology of computational ecosystems, to study how agent populations are affected in response to users' feedback and selection. In a large-scale version of the system (thousands of users), interesting dynamics would emerge with clusters of agents moving across the network in complicated ways, embodiments of trends or fashions. In the scope of a one year project, results from the experience of smaller user communities will give basic insight into ecologies of agents.

3.4 Summary of Research Topics

This thesis work will be an investigation into a new way of designing computer systems, one that embodies the understanding that the Internet is a crucial environment for software. Toward that end, there are several related research topics that will be covered in the course of the master's work:

- **Designing an Environment for Distributed Agents**: studying the architectural issues behind building a framework for distributed agents.
- **Designing Agents to Reflect Presence**: working in the application domain of showing the presence and activity of people online.
- **Understanding Computational Ecologies**: studying the behavior of complex systems of distributed agents in a network environment and learning how to build computational ecologies.

A Resources

The main resource requirement for this work is easy to fill; several computers running Java. Existing computers inside the lab are more than adequate. A more complicated requirement is having multiple users to experiment with the system. The plan is to make the system simple to run so anyone interested in the project can use it with little effort. Even a small group of users should be sufficient to provide experience about the behavior of the agent ecology.

B Deliverables

B.1 Software

- An environment for ecologies of distributed agents.
- Distributed agents that live in the network environment
- Agents that represent the presence and activity of people.

B.2 Writing

- Documentation of the design of the agent servers
- Description of the behavior of software ecologies
- Distillation of experience designing systems with distributed agents

C Plan of Study

December 15 Have a first pass at the basic framework in place. Install several servers in a simulation network on one machine. Choose a distributed agent technology, build preliminary APIs for the server environment and write first test agents.

February 1 Round out server environment API, reevaluate and document (as a thesis chapter) the distributed agent framework. Build basic suite of agents; presence of people and activity of people and machines. Recruit first users.

March 1 Evaluate the use of existing agents. Build more agents from experience with first set. Incorporate user feedback and agent design experience into the structure of the server framework.

April 1 Document the agents, their design and use. Examine the behavior of the ecosystem. Discover interesting synergies between ecosystem and application and explore them.

May 8 Thesis due. Write up results: experiences with the ecosystem, general introduction and conclusions. Take readers out for a nice dinner.

D Bibliography

[1] Michael L. Best. An ecology of the net: message morphology and evolution in NetNews. Master's thesis, MIT Media Lab, 1996. Machine Understanding Technical Report 96-001.
Ecological metaphors for the dynamics of memes on Usenet.

[2] Jon Bosak. XML, Java, and the future of the Web, March 1997.
http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.html
A summary of XML with a concentration on adding dynamic behavior to the Web.

[3] K. Mani Chandy, Joseph Kiniry, Adam Rifkin, and Daniel Zimmerman. Framework for structured distributed object computing. submitted to Parallel Computing, 1997.
http://www.infospheres.caltech.edu/papers/framework/framework.html
Requirements and architecture of Infospheres, a distributed objects framework.

[4] K. Mani Chandy, Adam Rifkin, Paolo A.G. Sivilotti, Jacob Mandelson, Matthew Richardson, Wesley Tanaka, and Luke Weisman. A world-wide distributed system using Java and the Internet. Technical report, Caltech Computer Science, 1996. Caltech CS Technical Report CS-TR-96-0.

http://www.infospheres.caltech.edu/papers/chandy_etal/hpdc.html
Requirements and architecture of Infospheres, a distributed objects framework.

[5] Daniel Chang and Danny Lange. Mobile agents: A new paradigm for distributed object computing on the WWW, December 1995. Submitted to OOPSLA '96.
http://www.trl.ibm.co.jp/aglets/ma.html
Mobile agents for distributed computation.

[6] dist-obj FAQ and mailing list archives.
http://www.infospheres.caltech.edu/mailing_lists/dist-obj/distobjgroup.html
A community of researchers working on radical distributed object computing.

[7] Judith Donath. *Inhabiting the virtual city: The design of social environments for electronic communities*. PhD thesis, MIT Media Arts and Sciences, 1997.
http://judith.www.media.mit.edu/Thesis/
Issues of representing presence and representing community online.

[8] David Gelernter. *Mirror Worlds*. Oxford University Press, 1991.
Conceptions of complex systems of distributed computation.

[9] Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile agents: Are they a good idea? Technical report, IBM TJ Watson, 1995.
http://www.research.ibm.com/massive/mobag.ps
Presents various arguments for mobile agents: while no one claim is compelling, the collection is persuasive.

[10] B. A. Huberman, editor. *The Ecology of Computation*. Elsevier Science Publishers, 1988.
Collection of papers about applying ecological metaphors to computation.

[11] Kevin Kelley. *Out of Control*. Addison-Wesley, 1994.
Ecological metaphors for complex systems of distributed computation.

[12] Danny B. Lange and Daniel T. Chang. IBM Aglets workbench: Programming mobile agents in Java, September 1996.
http://www.trl.ibm.co.jp/aglets/whitepaper.htm
White paper: an overview of the aglets architecture.

[13] Andrew Leonard. *Bots: The Origin of New Species*. Hardwired, 1997.
Stories about the complex interactions of programs on the Internet.

[14] MAS 964: Principles of visual interface design, Spring 1997.
http://acg.media.mit.edu/courses/spring97/mas964/
Media Lab course on graphic design in Java.

[15] Nelson Minar. Floyd, a global network of communicating agents. 1995.
http://www.santafe.edu/~nelson/ccr/floyd/floyd/floyd.html
A paper describing earlier research in representing presence with distributed agents.

[16] Nelson Minar. Computational media for mobile agents. December 1996.
http://nelson.www/people/nelson/research/dc/
Design of a system for using mobile agents to use idle CPU cycles.

[17] Tom S. Ray. A proposal to create a network-wide biodiversity reserve for digital organisms. Technical report, ATR, 1995.
http://www.hip.atr.co.jp/˜ray/pubs/reserves/reserves.html
A plan for building a networked ecology of evolving programs.

[18] Mitchel Resnick. *Turtles, Turmites, and Traffic Jams*. MIT Press, 1994.
Ecological computing and complicated interactions between multiple software agents.

[19] Mirabilis Software. ICQ.
http://www.icq.com/
An Internet chat and presence system.

[20] Objectspace Voyager core package version 1.0 technical overview, 1997.
http://www.objectspace.com/voyager/VoyagerTechOview.pdf
Description of the Voyager distributed objects framework.

[21] J.E. White. Telescript technology: The foundation for the electronic marketplace, 1996.
http://www.genmagic.com/Telescript/Whitepapers/wp1/whitepaper-1.html
Overview of Telescript infrastructure, a very early mobile agent system.