IEEE distributed systems ONLINE
Expert-authored articles and resources

Future Directions for Mobile Agent Research

David Kotz, Robert Gray, and Daniela Rus • Dartmouth College

The field of mobile agents should shift its emphasis toward mobile code, in all its forms, rather than continue focusing on mobile agents. The development of modular components will help application designers take advantage of code mobility without having to rewrite their applications to fit in monolithic, mobile agent systems.

Last year, several mobile agent researchers gathered for dinner to discuss the future directions of mobile agent research (see the "Discussion Participants" sidebar). The event was the most recent in our series of Dartmouth Workshops in Transportable Agents and was held in Zurich immediately following the joint conference on Agent Systems and Applications and Mobile Agents in September 2000. This article represents an amalgamation of the comments made during the discussion, not necessarily the opinions of the authors or a unanimous consensus among those attending.

The purpose of our discussion was to identify key research directions that will let mobile agent research have an impact—beyond the immediate mobile agent research community—into other research areas in computer science and the commercial world.

Mobile-code concept

A mobile agent is a running program that can move from host to host in a network when and where it chooses. Mobile agents are one form of mobile code. In its simplest form, the concept of mobile code involves dynamically installing code on a remote host. In Web applications, applets and servlets are a common form of mobile code. The mobile code concept also appears in *remote evaluation* systems that extend the notion of remote procedure calls to transport the procedure to the server along with the call.

Many researchers extend the mobile code concept to *mobile objects*, in which an object (code and data) moves from one host to another. The mobile agent abstraction extends this notion further by moving code, data, and a thread from one host to another. A mobile agent runs in one location, moves to another host, and continues at that host. An external agent usually moves mobile code and mobile objects; mobile agents usually have migration autonomy.

Mobile agents offer many potential advantages over traditional approaches. By moving the computation to another host, it is often possible to collocate the computation with an important database, allowing high throughput, low latency access to that database. Compared to more traditional client server approaches, mobile agents can avoid transmitting a large amount of data across the network, which is of particular value when the network is slow or unreliable. The mobile agent can move, with partial results, from one server to another until it has accomplished its task, then return to the originating host.

In addition to speed and reliability improvements, mobile agents can also help structure distributed applications. A service designed to relocate itself in the network to accommodate changing network conditions or the changing location of its clients can easily be written as a mobile agent. A client designed to trace an intruder's path through a network of hosts can run as a mobile agent that scans host logs, identifying the intrusion's source and jumping to that host for further tracking.

"We're trying to separate the logical design of the system from the placement of its components." — *M. Ranganathan*

There are many existing mobile agent systems, but few, if any, will fully meet the needs of those who program large, complex applications. There are few commercial mobile agent systems and even fewer standards. If mobile agents are to have significant impact, one of the tasks ahead is to identify the key features that make mobile agents successful, extract those features into a coherent, flexible set of composable software tools, and arrive at a standard interface.

Monolithic systems

Throughout our discussion, there was a general sense that the mobile agent community should be shifting its emphasis away from mobile agent systems. For the past seven or eight years, our nascent research field has seen the development of initial technology for mobile agents. Researchers and developers have put together the basic technology and tested how well the subsystems interact with each other.

This initial research phase was quite successful. Consider the work published in the Mobile Agents series of conferences, an incomplete but significant portion of the relevant research. There was much promising work on specific features in mobile agent systems, including persistence,[1] resource allocation,[2,3] orphan detection,[4] state capture,[5] security,[6] communication,[7] coordination,[7] and languages.[8] These conference proceedings also present several robust and efficient, albeit monolithic, mobile agent systems, such as AgentSpace,[9] Ara,[10] Concordia,[11] and Nomads.[12] One system, called *m*Code,[13] is a modular toolkit that supports several mobility paradigms.

During our discussion, Gian Pietro Picco noted that this monolithic approach to mobile agent systems is harming the spread and acceptance of mobile code. Developers are hesitant to create applications that require use of a new, large, monolithic system. In addition, because the monolithic systems are usually not tailored to specific application domains, and thus include significant overheads, the performance benefits of mobile code have not yet been made completely clear. Few quantitative performance results have been published about mobile agents, and there has been little motivation for developers to use mobile agents.

"Code mobility is a new way to look at how you structure an application. You're thinking in a different way from mainstream distributed computing, and that's the power. The power is not in the technology, it's in the architecture." — *Gian Pietro Picco*

The mobile agent community must therefore make a concerted effort to move away from monolithic systems—in fact, to move away from the idea of large, completely autonomous mobile agents—and instead apply the idea of mobile code to specific applications using whatever form of mobile code the application demands. Many of the subsystems developed as part of mobile agent systems will find their way into these applications, but it is no longer productive to try to take a monolithic system that has all the subsystems and make it fit the application. In our discussion, we considered component based architectures at length. Existing research suggests the provision of mobility components in a toolkit. Others proposed constructing mobile agents out of existing mobile or nonmobile components.

Furthermore, to justify the claims that we make as mobile agent researchers, research should—whenever possible—include a detailed quantitative evaluation of mobility's value. The recent Mobile Agents conference series includes few such papers. Quantitative justification for improved performance is necessary to convince others about when and how mobility can help.

Architecture and components

During the discussion, it became clear that it is important to distinguish between the use of mobile agent concepts as an architecture for applications and the use of a specific mobile agent system to implement applications. Despite the value of mobile agents as a programming abstraction, it is often unexpectedly difficult for a programmer to consider adding mobility to an existing application. To use a mobile agent system, programmers must typically significantly change their application so that it conforms to the mobile agent system's language and constraints.

"Most mobile agent systems try to solve ten problems at the same time. They tend to be monolithic. People who want to only use one little slice of the system have to install the whole thing." — *Gian Pietro Picco*

Furthermore, most mobile agent systems provide only one form of mobility, although there are several possible forms, such as applets, servlets, mobile agents with weak mobility, and mobile agents with strong mobility. Each form of mobility is useful in different situations.

"You have client-server. You have mobile agents. There's a lot of stuff in between that has not been addressed." — *Gian Pietro Picco*

Picco pointed out that the software industry has evolved a component architecture for many large software systems, and he proposed structuring mobile agent systems as a set of mobility components. At least mobile code and mobile agent systems should interoperate with existing mechanisms, such as remote method invocation, wherever possible. Furthermore, the current variety of features in mobile agent systems would be available as a set of orthogonal components so that programmers could plug in components as needed for their application. One component might provide mobility, and another security, another a certain form of communication.

"You should have some way to add mobility only when it's needed in your application." — *Gian Pietro Picco*

This component-based approach would allow much more flexibility for the programmer than we have seen in most mobile agent systems so far. The challenge is to distill the ideas that have evolved in the mobile agent community into a set of standard, reusable, orthogonal components that can be combined as needed. Not everyone was convinced that the component approach would work. Some thought that it might be too much work to build up an application out of small components because many programmers want a monolithic integrated system as a starting point. Others thought that it might be quite difficult to produce orthogonal, reusable components.

"One research direction is to look for a continuum of different abstractions from simple mobile code to mobile agents. From a simple base you could keep adding abstractions as you need them." — *David Kotz*

Christian Tschudin noted that there are many layers in which mobile agents and mobile code might be useful. Mobile code is most frequently discussed in the application layer, but in many cases it might be more valuable inside a middleware layer, supporting more conventional applications. In the active networking field, of course, mobile code is used in the network and routing layers. There was some interest in using mobile code to program software radios—to install a new radio protocol needed to communicate with towers in the geographic vicinity.

Recently, the mobile agent community has focused on Java as a base language for implementation. Although convenient, Java does not have all of the necessary features. Ideally, a mobile agent system should support multiple languages to

accommodate different application needs. The community will need to determine which features to push into languages as well as find ways to make the components work across multiple languages.

"To make mobility routine we need to address the language foundation. In addition to mobility, Java is missing fine-grain resource control and security." — *Jeff Bradshaw*

As M. Ranganathan suggested, in many application domains it might be better not to use a general purpose programming language. The idea is to give the user a set of components that the network administrator has tested and certified, and a simple language for composing components into mobile agents. For simple composition languages, the system might be able to derive guarantees on the mobile agent's size, resource consumption, and activity that would not be possible in a more general purpose programming language.

Educating other fields

To influence the world outside the mobile agent research community, we, as researchers, must educate those in other fields—including those in application development—that mobile agents are valuable. We must

- Encourage more quantitative studies on mobility's value and the component based approach to mobile application development
- Develop real applications that demonstrate the value of mobility and components
- Develop standard interfaces and protocols to encourage compatibility and reuse
- Develop practical solutions to the security and resource control challenges of mobile agents
- Encourage faculty to teach about mobile code in courses on distributed systems

Perhaps one of the most significant ways to demonstrate mobile code's value is to develop prototype applications in which the value becomes apparent.

"People don't use mobile code because they don't realize that it's useful. A lot of things come down to education, to awareness." — *Amy Murphy*

The Mobile Agents conferences provide several good examples, including using mobile agents to negotiate and monitor quality of service for multimedia applications, manage networks, videoconference, or disseminate information. Some papers have evaluated multiple applications or implemented the same application in multiple ways.

During our discussion, we identified a range of other applications, including Web servers (migrating to optimize their network position), network edge servers (dynamically installed to cache content), active documents, spacecraft (due to the high interplanetary latency, mobile code might offer better performance), network management (dynamically deployed monitoring tools), and network games.

"We believe that Web hosting is a common application for mobile code. Imagine a Web server that moves itself around in the network to optimize its position relative to the people currently using it." — *Colin Harrison*

Amy Murphy pointed out that it is important to remember that many of these applications might be implemented in closed networks, such as intranets, where a single developer controls all of the hosts and agents. A university might let mobile agents circulate to allow large scientific calculations to run during idle moments on university owned workstations. A telecommunications company might choose to use mobile code in its network management application.

"In IP telephony I see a real application for mobility, for example, a call-processing program that handles calls while you are disconnected." — *M. Ranganathan*

Tschudin mentioned that the mobile-code community should aim for something that is invisible, notably middleware. It might often be more appropriate to use mobile code inside the middleware for an application domain rather than in the applications themselves. The application might not move or be aware of the use of mobility, but some of the abstract services its middleware provides might take advantage of mobility. An application using a database, for example, might not know that its queries are encoded in numerous bits of mobile code sent to remote data warehouses.

Ideal applications

It was clear in our discussion that there is no killer app for mobile agents. Indeed, the participants were skeptical that any would be found. If, as Picco believes, mobile agents are a design technique, not an enabling technology, the designer of any distributed application should consider using mobile code in some form.

Still, Günter Karjoth encouraged everyone to look for applications with *sequential matchmaking*, in which the application must perform a sequence of activities involving data or services on multiple hosts. Of course, the same application could be implemented with a sequence of remote procedure calls, but under the right circumstances this sort of application might demonstrate the value of multihop mobile agents.

"People want to support mobile devices with scalable services located at the edge of the network. The ability to migrate functionality right into these services is going to be very important." — *Colin Harrison*

Jeff Bradshaw and Colin Harrison suggested that, in the future, software might not be distributed to end user computer systems. Instead, the software author would send it to an Application Service Provider (ASP) host, where the software registers itself as a service. End users would then use that service over the Internet. Any components that need to be on the end user machine would be installed when first needed. It is possible that mobile code could be helpful in both installation procedures. Others at the table were concerned about the potential security problems as well as the potential for unforeseen interactions between services or components.

An alternative that Niranjan Suri suggested was *service composition*. Although ASPs might not be comfortable letting users upload complex code, they might be willing to let users run simple code, such as code that makes calls to existing services on the ASP systems.

Although we did not discuss any applications in detail, there was clear interest in seeing more researchers implement these sort of applications in multiple ways (without mobile code and with a variety of forms of mobile code), analyze the different solutions quantitatively and qualitatively, and publish articles on the advantages and disadvantages of mobility in those applications.

Protecting hosts and agents

In our discussion of security, two things became clear: security is hard and is not always important. Many—if not most—applications for mobile code and mobile agents do not need these problems solved. For example, many applications occur inside closed systems. The security challenges here are much more limited than in the general open system case. As a community, we should encourage the development of applications or middleware based on the concept of mobile code, wherever their security requirements fit within our current limitations.

That said, the remaining security problems fit into two categories:

- Protecting host systems and networks from malicious agents
- Protecting agents from malicious hosts

There are many mechanisms to protect a host against malicious agents. Digital signatures and trust management approaches might help identify the agent and identify how much it should be trusted. But what, then, is the policy to determine whether an agent receives access, and how much access it should receive? This area requires more research.

"I think that we have quite well identified the threats against mobile-agent systems. Unfortunately for the most interesting ones we don't have answers." — *Günter Karjoth*

The malicious host problem, in which a malicious host attacks a visiting mobile agent, is the most difficult problem. To execute the agent and update its state, the host must of course be able to read and write the agent. A malicious host might steal private information from the agent or modify the agent to compute the wrong result or misbehave when it jumps to another site.

Karjoth and Tschudin have both addressed this problem, for example, by encrypting the agent's code in such a way that it can compute on encrypted input and produce encrypted output, which, when decrypted, is the output that would have been produced by running the original input through the original code. Unfortunately, this approach currently works only for certain classes of polynomials, not for general code.

Future directions

There is an unfortunate tendency to reinvent ideas from earlier research in distributed computing, relabel the ideas in a mobile computing or mobile agent context, and republish. We should embrace ideas from other fields and identify which ones are useful in the context of mobility, but we should avoid "reinventing" these ideas.

We should also avoid overselling mobility's value. Mobility is useful in some but not all situations. Our research's goal should be to help the broader community understand when, and how much, mobility might be of use.

We could summarize the primary theme of our discussion in one sentence: The future of mobile agents is not specifically as mobile agents. The concepts developed in the mobile agent community have value in many situations, but the monolithic mobile agent systems developed in the past decade are not necessarily the vehicle for those concepts to have an impact. The subthemes in our discussion centered on components, education and awareness, applications, standards, and security.

Recent years have seen the development of many mobile agent systems based on several slightly different semantics for mobility, security, and communication. The community now needs to start distilling the best of these ideas from all of the proposed approaches and identifying the situations where those approaches best apply.

We need quantitative measurements of the value of each form of mobility, communication, and so forth. We need qualitative analysis of the value of these ideas in helping structure distributed applications. Specifically, when, where, and why are different forms of mobility useful? We need to remember that Java is not the only language and that multilanguage support is important as well as isolating the value of our ideas from their implementation in a particular language platform.

Once we have identified and quantified the key concepts and ideas, we need to develop an infrastructure that would encourage the development of orthogonal, reusable components that implement those ideas. The infrastructure would begin with a minimal base and let the programmer add only those components—those abstractions—that are needed for that application.

As a community, we need to reach out to other research and development communities, both to spread awareness of mobile code's value (without overselling it) and to increase our awareness of the types of applications that might benefit from our ideas. We can monitor other research communities, such as the distributed computing, programming, security, and software engineering communities, adapting their ideas to the world of mobile computers and mobile code. Eventually, we can form many of the distilled ideas into a set of standards that can encourage the incorporation of the ideas into more applications.

It still seems too early to begin that process, however. We need implementations of real applications that demonstrate the value, with meaningful analysis, of the benefits of mobile code. In the discussion, two specific ideas emerged that might encourage real application development: Create a contest to award a prize to the best network game based on mobile code, or encourage all mobile agent research groups to enhance their Web sites with a mobile agent platform so that mobile agents could visit their sites. The resulting set of sites could be an interesting proving ground for mobile agent applications.

In the end, the mobile agent research community should strive to contribute by improving our understanding of the value of mobility, distilling our ideas into a core set of concepts, encouraging the construction of those concepts as a set of software components, educating those outside our community about the value of mobility, and demonstrating its value through the its use in real applications and middleware.

Acknowledgments
The event was supported by DoD MURI contract F49620-97-1-03821. Many thanks to Jeff Bradshaw, Colin Harrison, Günter Karjoth, Amy Murphy, Gian Pietro Picco, M. Ranganathan, Niranjan Suri, and Christian Tschudin for their willingness to attend our conversation and comment on drafts of this paper. Thanks to Wanda Bachmann for transcribing the tapes of the conversation, and thanks to the Hotel zum Storchen in Zurich for the excellent dinner.

References

1. M.M. da Silva and A.R. da Silva, "Insisting on Persistent Mobile Agent Systems," *Proc. 1st Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1219, 1997, pp. 174–185.
2. F. Christian Tschudin, "Open Resource Allocation for Mobile Code," *Proc. 1st Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1219, 1997, pp. 186–197.
3. M. Lal and R. Pandey, "CPU Resource Control for Mobile Programs," *Proc. 1st Int'l Symp. Agent Systems and Applications and Third Int'l Symp. Mobile Agents (ASA/MA99)*, IEEE CS Press, Los Alamitos, Calif. 1999, pp. 74–88.
4. J, Baumann and K, Rothermel, "The Shadow Approach: An Orphan Detection Protocol for Mobile Agents," *Proc. 2nd Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1477, 1998, pp. 2–13.
5. S. Fünfrocken, "Transparent Migration of Java Based Mobile Agents: Capturing and Re Establishing the State of Java Programs," *Proc. 2nd Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1477, 1998, pp. 26–37.
6. I. Biehl, B. Meyer, and S. Wetzel, "Ensuring the Integrity of Agent Based Computations by Short Proofs," *Proc. 2nd Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1477, 1998, pp. 183–194.
7. G. Cabri, L. Leonardi, and F. Zambonelli, "MARS: A Programmable Coordination Architecture for Mobile Agents," *IEEE Internet Computing*, vol. 4, no. 4, July 2000, pp. 26–35.
8. P.T. Wojciechowski and P. Sewell, "Nomadic Pict: Language and Infrastructure Design for Mobile Agents," *Proc. First Int'l Symp. Agent Systems and Applications and Third Int'l Symp. Mobile Agents* (ASA/MA99), IEEE CS Press, Los Alamitos, Calif., 1999, pp. 2–12.
9. A. Silva, M.M. da Silva, and J. Delgado, "An Overview of AgentSpace: A Next Generation Mobile Agent System," *Proc. 2nd Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1477, 1998, pp. 148–159.
10. H. Peine and T. Stolpmann, "The Architecture of the Ara Platform for Mobile Agents," *Proc. 1st Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1219, 1997, pp. 50–61.
11. D. Wong et al., "Concordia: An Infrastructure for Collaborating Mobile Agents," *Proc. 1st Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1219,1997, pp. 86–97.
12. N. Suri et al., "NOMADS: Toward a Strong and Safe Mobile System," *Proc. 4th Int'l Conf. Autonomous Agents*, ACM Press, New York, 2000, pp. 163–164.
13. G.P. Picco, "Code: A Lightweight and Flexible Mobile Code Toolkit," *Proc. 2nd Int'l Workshop on Mobile Agents*, Springer-Verlag, Berlin, Germany, vol. 1477, 1998, pp. 160–171.

David Kotz is an associate professor of Computer Science at Dartmouth College. His research interests include mobile agents, ubiquitous computing, wireless networks, parallel I/O, and computer ethics. He received a PhD in computer science from Duke University. Contact him at the Department of Computer Science, Dartmouth College, Hanover, New Hampshire, 03755 3510; david.kotz@dartmouth.edu.

Robert Gray is a research engineer in the Thayer School of Engineering at Dartmouth College, an adjunct professor in the department of computer science, and a research engineer in the Institute for Security Technology Studies. His research interests include the performance and scalability of mobile agents, other forms of mobile code, and information retrieval tools for law enforcement personnel. He received a PhD in computer science from Dartmouth College. Contact him at the Department of Computer Science, Dartmouth College, Hanover, New Hampshire, 03755 3510; robert.gray@dartmouth.edu.

Daniela Rus is an associate professor in the computer science department at Dartmouth College, where she founded and directs the Dartmouth Robotics Laboratory. She also codirects the Transportable Agents Laboratory and the Dartmouth Center for Mobile Computing. Her research interests include distributed manipulation, 3D navigation, self reconfiguring robotics, mobile agents, and information organization. She received a PhD in computer science form Cornell University. Contact her at the
Department of Computer Science, Dartmouth College, Hanover, New Hampshire, 03755 3510; daniela.rus@dartmouth.edu.

Related Links:
(purchase or Digital Library members log in)

- Enhancing Network Management using Mobile Agents, W.J. Buchanan, M. Naylor and A.V. Scott, *Proceedings of the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems*
- System Mechanisms for Partial Rollback of Mobile Agent Execution, M. Strasser and K. Rothermel, *Proceedings of the The 20th International Conference on Distributed Computing Systems* (ICDCS 2000)
- Nomad:Mobile Agent System for an Internet-Based Auction House, Tuomas Sandholm, Qianbo Huai, *IEEE Internet Computing* (March-April 2000)

Discussion Participants

From left to right: Gian Pietro Picco, David Kotz, Amy Murphy, Bob Gray, Niranjan Suri, Christian Tschudin, Jeff Bradshaw, Daniela Rus (front), Colin Harrison (back), Günter Karjoth, and M. Ranganathan.

Jeff Bradshaw is a research scientist at the Institute for Human and Machine Cognition at the University of West Florida. Bradshaw is chair of the RIACS Science Council for NASA Ames Research Center and chair of ACM SIGART. He is a member of the NASA team developing agent based software for the Personal Satellite Assistant, a softball sized flying robot for use by astronauts in the shuttle and the International Space Station. He received a PhD in cognitive science from the University of Washington.

Colin Harrison is director of global services research at the IBM T. J. Watson Research Center. His research interests include magnetics, medical imaging, parallel computing, mobile networking, intelligent agents, telecommunications services, and knowledge management. He received a PhD in material science from Imperial College, London.

Günter Karjoth is a research staff member in the computer science department at the IBM Zurich Research Laboratory. His research interests include modeling, validating, and implementing distributed systems.

Amy Murphy is an assistant professor in the department of Computer Science at the University of Rochester. Her research interests include the development of standard algorithms for mobility and the design, specification, and implementation of mobile middleware systems. She received a DSc from Washington University.

Gian Pietro Picco is an assistant professor in the department of electronics and information at Politecnico di Milano, Italy. His research interests include distributed systems that exhibit mobility, be it logical or physical.

M. Ranganathan is a computer engineer in the advanced networking technologies division of the National Institute of Standards and Technology. His research interests include signaling, service creation, routing, and quality of service for IP telephony. He received an MS in mechanical and electrical engineering from the University of Illinois at Chicago and an MS in computer science from the University of Maryland.

Niranjan Suri is a research scientist at the Institute for Human and Machine Cognition at the University of West Florida. His research interests include virtual machines, distributed computing, autonomous intelligent agents, network security, persistent objects, and Internet based collaboration tools. He received an MS in computer science from the University of West Florida.

Christian Tschudin is an associate professor at Uppsala University, Sweden. His research interests include mobile code, active (and wireless) networks, cryptography, market based control, operating systems and artificial life. He received a PhD in computer science from the University of Geneva.