Abstract

During a discussion in September 2000 the authors examined the future of research on mobile agents and mobile code. (A mobile agent is a running program that can move from host to host in network at times and to places of its own choosing.) In this paper we summarize and reflect on that discussion. It became clear that the field should shift its emphasis toward mobile code, in all its forms, rather than to continue its narrow focus on mobile agents. Furthermore, we encourage the development of modular components, so that application designers may take advantage of code mobility without needing to rewrite their application to fit in a monolithic mobile-agent system. There are many potential applications that may productively use mobile code, but there is no “killer application” for mobile agents. Finally, we note that although security is an important and challenging problem, there are many applications and environments with security requirements well within the capability of existing mobile-code and mobile-agent frameworks.

1 Introduction

Last year several mobile-agent researchers gathered for a dinner and discussion of the future directions in mobile-agent research. The September 2000 event was the most recent in our series of Dartmouth Workshops in Transportable Agents, and was held in Zurich immediately following ASA/MA 2000 (the joint conference on Agent Systems and Applications and on Mobile Agents). Biographies of the participants appear in the appendix.

Our purpose was to identify key research directions that will allow mobile-agent research to have an impact beyond the immediate mobile-agent research community, to other research areas in computer science and to the commercial world. This article summarizes and reflects on the most important themes in that discussion.

The text of this article represents an amalgamation of the comments made during the discussion, not necessarily the opinions of the authors and not necessarily a unanimous consensus of those attending. In summarizing, we attempt to present the main ideas and the general sense of the discussion.

2 Background

A mobile agent is a running program that can move from host to host in network when and where it chooses. Mobile agents are one form of mobile code. In its simplest form, the concept of mobile code involves the dynamic installation of 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 [SG90], which extend the notion of remote procedure call 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) are moved 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 (with its state) to another host, and continues at that host. Mobile code and mobile objects are normally moved by an external entity; mobile agents usually have migration autonomy.

Mobile agents have many potential advantages. By moving the computation to another host it is often pos-
sible to co-locate 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 as much
data across the network, which is of particular value when
the network is slow or unreliable (as in the Internet, or
wireless networks). The mobile agent can move, with par-
tial results, from one server to another until it has accom-
plished its task, and then return to the originating host,
which may freely be disconnected during the agent’s trav-
els.

In addition to speed and reliability improvements, mo-
bile agents can also be helpful in structuring many dis-
tributed applications. A service that wishes to reloca-
t itself in the network to accommodate changing network con-
ditions or the changing location of its clients can easily
be written as a mobile agent [RASS97, APGG00, BPR98]. A
client that wishes to trace the path of an intruder through
a network of hosts can be written as a mobile agent that
scans host logs, identifying the source of the intrusion 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 programmers
of large, complex applications. There are few commer-
cial mobile-agent systems, and fewer standards, available.
Clearly one of the tasks ahead, if mobile agents are to
have significant impact, 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.

3 Monolithic systems

Throughout the discussion there was a general sense that
the mobile-agent community should be shifting its em-
phasis away from mobile-agent systems.

For the last seven or eight years, our nascent research
field has seen the development of initial technology for
mobile agents. Researchers and developers put together
the basic technology and tested how well the (many)
needed subsystems interacted with each other.

This initial research phase was quite successful. Con-
sider the work published in the “Mobile Agent” se-
ries of conferences (MA97, MA98, ASA/MA99, and
MA2000), an incomplete but significant portion of
the relevant research. There was a lot of promis-
ing work on specific features in mobile-agent sys-
tems, including persistence [dSdS97], resource alloca-
tion [Tsc97, LP99], orphan detection [BR98], state cap-
ture [Fin98, WBDF98, SBB+00a, SSY00, TRV+00], se-
curity [BMW98, KAG98, Fin99, BV99, Kar00], com-
munication [CLZ00, MP99, RSM99, RBM00], coordi-
nation [CLZ00], and languages [WS99, DELM00].
These conference proceedings also present several ro-
bust and efficient, albeit monolithic, mobile-agent sys-
tems, such as AgentSpace [SdD98], Ara [PS97], Con-
cordia [WPW+97], and NOMADS [SBB+00b]. One,
µCode [Pic98], is a modular toolkit that supports a variety of
mobility paradigms.

During our discussion Gian Pietro Picco noted that this
monolithic approach to mobile-agent systems is hurting
the spread and acceptance of mobile code. Developers
are hesitant to develop applications that require them to
use a new, large, monolithic system. In addition, since
the monolithic systems are usually not tailored to specific
application domains, and thus include significant over-
heads that are not needed for many applications, the per-
formance benefits of mobile code in general have not yet
been made completely clear. Indeed, there few quan-
titative performance results yet published about mobile
agents. So far, there has been little motivation for devel-
opers to use mobile agents.

Thus the mobile-agent community must make a co-
certed effort to move away from the 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 suggests or demands.
Many of the subsystems developed as part of the 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 try to make that
system fit the application (or more precisely, bend the
application to fit the system). In our discussion we consid-
ered component-based architectures at length (see the next
section). Existing research suggests the provision of mo-
tility components in a toolkit [Pic98]. Others propose to
construct mobile agents out of existing mobile [GFP99]
or non-mobile [QV99] components.

Furthermore, to justify the claims that we make as
mobile-agent researchers, research should whenever pos-
sible include a detailed quantitative evaluation of the value
of mobility. In the recent Mobile Agent conference se-
ries, there are few such papers [Tsc97, Fin98, WBDF98,
Jobh98, KP99, SH99, DSDL99, LP99, TR99]. Other no-
table papers include [BPR98, FPV98, BGM+99]. Quanti-
tative justification for improved performance is necessary
to convince others about when and how mobility can help
them.
4 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 your application, and the use of a specific mobile-agent system to implement your application.

“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

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 make significant changes to their application so that it conforms with the language and constraints of the mobile-agent system.

“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 actually several possible forms (such as applets, servlets (remote evaluation), mobile agents with weak mobility, and mobile agents with strong mobility). Each form of mobility is useful in different situations [FPV98].

“You have client-server... you have mobile agents. There’s a lot of stuff in between that has not been addressed.”
— Gian Pietro Picco

Gian Pietro Picco pointed out that the software industry has evolved a component architecture for many large software systems, and proposed that mobile-agent systems be structured as a set of mobility components. At least, mobile-code and mobile-agent systems should interoperate with existing mechanisms, such as RMI communication, 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, 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, then, 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 may be too much work to build up an application out of small components, that many programmers want a monolithic integrated system as a starting point. Others thought that it may be quite difficult to actually 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 may be useful. It is most frequently discussed in the application layer, but in many cases it may 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. Even lower, there was some mention of using mobile code to program software radios, e.g., to install a new radio protocol that is needed to communicate with towers in the geographic vicinity.

Recently the mobile-agent community has focused on Java as a base language for implementing its ideas. Although convenient, Java does not have all of the necessary features. Ideally, a mobile-agent system should support multiple languages to accommodate the different needs of a variety of applications. Challenges for the community, then, are to determine which features need to be pushed into languages, and to find ways to make the components mentioned above work across multiple languages.

Indeed, as M. Ranganathan suggested, in many application domains it may be better not to use a general-purpose programming language. Give the user a set of compo-
“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

To have an impact on the world outside of the mobile-agent research community we, as researchers, must educate those in other fields of research and those in application development that mobile agents are a valuable tool. We discussed several steps that would help achieve that goal: 1) encourage more quantitative studies on the value of mobility and of the component-based approach to mobile-application development, 2) develop real applications that demonstrate the value of mobility and components, 3) develop standard interfaces and protocols to encourage compatibility and re-use, 4) develop solid practical solutions to the security and resource-control challenges of mobile agents, and 5) encourage faculty to teach about mobile code in courses on distributed systems.

5.1 Applications

Perhaps one of the most significant ways to demonstrate the value of mobile code is to develop prototype applications in which the value becomes apparent. In the Mobile Agent conferences there are several good examples, including the use of mobile agents in negotiating and monitoring quality of service for multimedia applications [OOC97], network management [BGP97, PPC99], videoconferencing [BPR98], or information dissemination [TR99]. Some papers evaluated multiple applications or implemented the same application in multiple ways [GV97, Joh98, BJP’00].

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), spacecraft (due to the high inter-planetary latency, mobile code may 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 from 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 may be implemented in “closed” networks, such as “intranets,” where a single developer controls all of the hosts and all of the agents. A university may allow mobile agents to circulate to allow large scientific calculations to run during idle moments on university-owned workstations. A telecommunications company may 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

As mentioned earlier in a comment from Christian Tschudin, “we should aim for something that is very invisible,” notably the middleware. It may often be more appropriate to use mobile-code inside the middleware for an application domain, rather than in the applications themselves. The application itself may not move, or be aware of the use of mobility, but some of the abstract services provided by its middleware may take advantage of mobility. An application using a database, for example, may be unaware that its queries are encoded into numerous bits of mobile code sent to remote data warehouses.

It was clear that there is no “killer app” for mobile agents. Indeed, the participants were skeptical that any would be found; if, as Gian Pietro Picco believes, mobile agents are a design technique, not an enabling technology, the designer of any distributed application should consider the use of 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 ser-
vices 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 may demonstrate the value of multi-hop mobile agents.²

“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 may 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 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 suggested by Niranjan Suri was Service Composition. Although an ASP may not be comfortable allowing you to upload complex code to their site, they may be willing to let you run simple code, e.g., code with no loops, that makes calls to existing services on the ASP.

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 papers on the advantages and disadvantages of mobility in those applications.

6 Security

In our discussion of security, two things became very clear: 1) security is hard, and 2) security is not always important. Indeed, many if not most applications for mobile code and mobile agents do not need these problems to be 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, and protecting agents from malicious hosts.

There are many mechanisms to protect a host against malicious agents. Digital signatures and trust-management approaches may help to identify the agent and how much it should be trusted. But what then is the policy, what access and how much access do certain agents receive? This area needs a lot more work.

The “malicious host problem,” in which a malicious host attacks a visiting mobile agent, is the most difficult and largely unsolved. Consider that, 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 to misbehave when it jumps to another site. Günter Karjoth and Christian 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 [Tsc99]. Unfortunately this approach currently works only for certain classes of polynomials, not general code.

7 What should we not do?

There is an unfortunate tendency to reinvent ideas from earlier research in distributed computing, relabel the ideas in the mobile-computing or mobile-agent context, and republish. We should of course embrace ideas from other fields, identify which ideas are useful and which are not in the context of mobility, but avoid reinventing these ideas.

It is definitely not a good idea to over-sell the value of mobility. Mobility is useful in some but not all situations. The goal of our research should be to help the broader community understand when, and how much, mobility might be of use.

² Single-hop mobile agents move from a client to a server, and perhaps back to the client. Multi-hop mobile agents visit more than one server before returning to the client.
Summary

The primary theme of our discussion could perhaps be summarized 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 sub-themes in the 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, communication, and the like. The community now needs to begin to distill the best of these ideas from all of the proposed approaches, and identify 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 to 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, that multi-language support is important, and that it is important to isolate 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 allow the programmer to 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 the value of mobile code (without over-selling!) and to become more aware ourselves of the types of applications that may benefit from our ideas. We can monitor other research communities, such as distributed computing, programming language, security, and software-engineering communities, adapting their ideas to the world of mobile computers and mobile code.

Eventually, many of the distilled ideas can be formed 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.

We need implementations of real applications that demonstrate the value, with meaningful analyses of the benefits of mobile code in those applications. There were two specific ideas that might encourage real application development. One was to create a contest to award a prize to the best network game based on mobile code. Another was to encourage all mobile-agent research groups to enhance their web site with a mobile-agent platform, so that mobile agents could visit their web site. 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, by distilling our ideas into a core set of concepts, by encouraging the construction of those concepts as a set of composable software components, by educating those outside our community about the value of mobility, and by demonstrating its value through the its use in real applications and middleware.
Biographies

Jeff Bradshaw is a research scientist at the Institute for Human and Machine Cognition at the University of West Florida. Previously, Dr. Bradshaw led the intelligent agent technology group at The Boeing Company, where he was responsible for the development of the KAoS agent framework and its use in applications such as the NASA-sponsored Aviation Extranet and the DARPA control of agent-based systems (CoABe) program. He is a member of the NASA team developing agent-based software for the Personal Satellite Assistant (PSA), a softball-sized flying robot for use by astronauts in the shuttle and the International Space Station. Jeff is chair of the RIACS Science Council for NASA Ames Research Center and chair of ACM SIGART.

Bob 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 current research, which is funded by the Department of Defense, DARPA, and the National Institute of Justice, focuses on the performance and scalability of mobile agents and other forms of mobile code, and on information-retrieval tools for law enforcement personnel. Bob received a B.S. in Computer Science from the University of Vermont in 1993 and a Ph.D in Computer Science from Dartmouth College in 1997.

Colin Harrison is Director of Global Services Research at the IBM T. J. Watson Research Center in New York. Following his thesis work in materials science at Imperial College, London, he spent several years at CERN developing the real-time control system for the SPS accelerator. He then returned to EMI Central Research Laboratories in London, and lead the development of the world's first clinically-useful MRI system. He joined IBM in San Jose and has enjoyed a career leading from magetics, to medical imaging, parallel computing, mobile networking, intelligent agents, telecommunications services, and knowledge management. He now has broad responsibility for the application of research skills in IBM's services businesses and is leading the establishment of new program in e-Learning.

Günter Karjoth is a research staff member in the computer science department at the IBM Zurich Research Laboratory. He has done research in the fields of protocol engineering and network security, with a current focus on access control in distributed object systems. His research interest is in the modeling of distributed systems, their validation and implementation.

David Kotz is an Associate Professor of Computer Science at Dartmouth College, Hanover NH. He received the M.S. and Ph.D degrees in computer science from Duke University in 1989 and 1991, respectively. He received the A.B. degree in computer science and physics from Dartmouth College, Hanover NH, in 1986. He rejoined Dartmouth College in 1991 and was promoted with tenure to Associate Professor in 1997. His research interests include mobile agents, parallel and distributed operating systems, multiprocessor file systems, and computer ethics.

Amy Murphy received a B.S. in Computer Science from the University of Tulsa in 1995, M.S. and D.Sc. degrees from Washington University in St. Louis, Missouri in 1997 and 2000 and is currently an Assistant Professor in the department of Computer Science at the University of Rochester in Rochester, New York. Her research interests include the development of standard algorithms for mobility and the design, specification, and implementation of mobile middleware systems. These topics are integrated under the theme of enabling the rapid development of dependable applications for both physically and logically mobile environments.

Gian Pietro Picco is an Assistant Professor at the Department of Electronics and Information at Politecnico di Milano, Italy. Prior to this current appointment, he was a Visiting Assistant Professor at Washington University in St. Louis, Missouri, USA. His research interests are in distributed systems that exhibit mobility, be it logical or physical. His work in this area thus far has investigated several aspects spanning from theoretical models to systems research, and has led to several publications, some of which are widely referenced by the research community.

M. Ranganathan is a Computer Engineer in the Advanced Networking Technologies division of the National Institute of Standards and Technology (NIST). He holds M.S. Degrees in Mechanical and Electrical Engineering from the University of Illinois at Chicago, and an M.S. in Computer Science from the University of Maryland. He holds a B Tech. degree in Mechanical Engineering from the Indian Institute of Technology, Madras. His professional life includes 4 years as a Mechanical Engineer designing heat-transfer and plastics processing equipment and 16 years as a Computer Engineer. As a Computer Engineer, in the IBM Corporation Ranganathan contributed to the design and development of the memory manager component of the OS/2 kernel and worked on workstation- and network-security issues for AIX and OS/2. At the University of Maryland, Ranganathan developed a mobile-agent platform called Sumatra that incorporated thread mobility into the Java Virtual Machine. At NIST, under DARPA funding, Ranganathan developed a re-configurable distributed event-oriented scripting platform called AGNI, applied it to building collaborative environments and distributed test systems, and studied issues relating to peer-to-peer communication in such environments. His current technical interests are in signaling, service creation, routing and quality of service for IP tele-
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 co-founded and co-directs the Transportable Agents Laboratory and the Dartmouth Center for Mobile Computing. She holds a Ph.D degree in computer science form Cornell University. Her research interests include distributed manipulation, 3d navigation, self-reconfiguring robotics, mobile agents, and information organization. She is an Alfred P. Sloan Foundation Fellow and holds an NSF Career award.

Niranjan Suri is a Research Scientist at the Institute for Human and Machine Cognition (IHMC) at the University of West Florida. His recent research led to the development of the NOMADS mobile agent system for Java-based agents with secure execution and anytime and forced mobility. He developed and implemented a Java-compatible Virtual Machine with key extensions to support mobile state and resource control within the Java environment. His research interests include virtual machines, distributed computing, autonomous intelligent agents, network security, distributed, persistent objects, and Internet-based collaboration tools.

Christian Tschudin is an Associate Professor at Uppsala University, Sweden. He graduated in Mathematics, Physics and Sociology at the University of Basel, Switzerland in 1986 and received his PhD in computer science at the University of Geneva in 1993 on a computer network architecture based on mobile code. Before taking the position in Sweden, he was at the International Computer Science Institute, Berkeley, USA, and also worked at the University of Zurich as a senior research associate. His research interests revolve around mobile code and include active (and wireless) networks, cryptography, market-based control, operating systems and artificial life.

Acknowledgements

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 to 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


Christian F. Tschudin. Open resource allocation for mobile code. In *Proceedings of


