

# Mobile Information Agents for Cyberspace - State of the Art and Visions

Todd Papaioannou

DALi, Inc. 1020 Mission Street, South Pasadena, CA 91030, USA,  
toddp@acm.org,  
<http://www.luckyspin.org>

**Abstract.** As ubiquitous computing becomes a reality, the amount of information we are confronted by is becoming overwhelming. In addition, the myriad of devices we can use to access this information is continually increasing. We require new technologies to assist us in locating and filtering this information, which are also able to deliver it at our convenience. One technology that show potential in this area is Mobile Information Agents. In this paper, we review the current state-of-the-art in this field, and suggest key issues that must be addressed if widespread adoption is to happen. Lastly, we look forward to the future and postulate where this new technology may take us.

## 1 Introduction

The last decade has seen an explosion in the growth and use of the Internet. Rapidly evolving network and computer technology, coupled with the exponential growth of services and information available on the Internet, is heralding a new era of ubiquitous computing. Hundreds of millions of people will soon have pervasive access to a huge amount of information, which they will be able to access through a plethora of diverse computational devices. These devices are no longer isolated number crunching machines; rather they are on our desks, on our wrists, in our clothes, embedded in our cars, phones and even washing machines. These computers are constantly communicating with each other via LANs, Intranets, the Internet, and through wireless networks, in which the size and topology of the network is constantly changing. And somewhere in this network, lies the piece of information that we desperately require; lost among the flotsam and jetsam of the Information Age. As our lives change, we are not only being overloaded with information, generated from a myriad of sources, but its topology can actually change from one day to the next. In this shifting sea, we require new techniques and technologies that are able to locate and filter the information on our behalf. One solution to these problems are mobile information agents.

## 2 Why Mobility?

In recent years the agent research field has become one of the most active and vibrant areas in information technology. For many, the question of whether this

technology will realise its potential is not an issue, they are convinced. In certain quarters of the intelligent agent community, however, there has often been a certain degree of dismissal when the topic of agent mobility is raised. Their view has been that every system can be constructed by using static agents, which communicate with each other across the network to achieve their goals. Fortunately, this view is beginning to change, since the position is becoming untenable.

The nature of the network is changing. No longer can we view it as a core and pervasive aspect of our computing architecture. It is no longer the reliable backbone we envisage, accessible from everywhere. The harsh reality is that bandwidth has become a commodity, and a scarce one at best. A plethora of new computing devices continually arriving on the market are going to eat up even more of the precious resource. These devices are no longer isolated number crunching machines; rather they are on our desks, on our wrists, in our clothes, embedded in our cars, phones and even washing machines. Further, these new devices do not enjoy the same connection speeds as say, a corporate network. But while they may be the poorer netizens, their numbers are prolific, and rising.

To understand how to architect information system in this environment, we should stop conceptualising the network as a connection of major arteries, which allow a constant flow of information. Rather we should view it as a lung, which has many differing sizes of information conduit. There are bronchi where large volumes of information can flow freely. These represent the Internet backbone, haven for mega corporations and academic institutions. Then, there are the bronchioles, which represent corporate LANs and well off home users. But at the boundaries of the network, are the minute alveoli, the portable devices and modem users, whose transfer rate is small. Pervasive information access from these different access points requires rethinking how we architect our systems.

Many opponents of mobility point towards a future when there will be enough bandwidth for everyone, and mobility will not be an issue. This argument is flawed in two aspects. Firstly, while there may indeed be plentiful bandwidth at the bronchial level, the many devices at the fringes of the network will never enjoy the types of connection speed users will have come to accept as the norm. Secondly, it is not clear whether we will ever be able to outpace demand and provide enough bandwidth to satisfy everyone's requirements. To illustrate let us imagine our kitchen has one waste bin. Currently, when it is full, we have to empty it or find it overflowing. If we add another bin to double our waste capacity, is it likely that we will think to empty one before the other is full. Or will we simply wait until both are full before taking action? If we look to the road systems around the globe, this is similar to what occurs with traffic systems, demand usually outstrips capacity on the major roads.

In addition, users are no longer satisfied with having their connection to the wider world rooted to a single location. Ubiquitous computing is seducing people with its pervasive nature, and users are becoming accustomed to accessing their information immediately, from wherever they may be at that time. Web-based email has demonstrated that users value the ability to access their email from

any computer. Soon, web terminals will be commonplace in public spaces, such as cafes, airports, taxi's and hotels. In the future, users will expect full access to any of their information from any device. Despite this, mobile devices will proliferate unchecked, since just as with public phones, Web terminals will never be available everywhere that a user might find themselves. WAP enabled cell phones are already becoming increasingly common.

## 2.1 What Does Mobility Offer Us?

The advantages that the mobile agent abstraction bring are extensive. They have been covered in greater detail in many other locations [6] [21] but are briefly summarised below.

**Table 1.** Summary of Advantages of Mobile Information Agents

Advantages	Description
Bandwidth savings	Instead of transferring lots of over the network, move the logic to be local to the data source
Limiting latency	By having the logic and data on the same machine, latency can be reduced
Disconnected operation	If a computer is going to be offline, then it is possible to move the active processes to another host
Stability	By using mobile code, software can be less dependant on the network, and therefore more stable. Mobility can be used to achieve replication for fault tolerance
Server Flexibility	A client isn't limited to the functions a server provides. Code mobility allows clients to upload new (or improved) functionality to the server
Simplicity of installed server base	Servers can become simpler, requiring less functionality pre-engineered from the outset which can help prevent legacy
Enable distributed computation	Mobile agents are inherently distributed, and as such can be a fundamental enabler for distributed computation

More importantly however, it is a software architecture that offers many abstractions to a designer. These include Client/server, Mobile Computation, Remote Evaluation and Mobile Agents and are covered extensively by in [19] and [24].

## 2.2 Research Status

Code mobility is not a completely new idea. There have been several widely used and successful mechanisms for moving code around a network previously employed, perhaps the best known being the PostScript language [1] that is used

to control printers. Recently though, mobility has been examined from a different perspective, and has become a burgeoning topic for discussion in mainstream distributed systems research. Mobility currently boasts a flourishing research community dedicated to investigating the potential of this new paradigm [15]. In this paper we define a mobile agent as:

"a software agent that is able to autonomously migrate from one host to another in a computer network."

The notion of a mobile agent was first established in 1994 with the release of a white paper by White [33] that described a computational environment known as "Telescript" [34]. In this environment, executing programs were able to transport themselves from one node to another in a computer network, in order to interact locally with resources at those nodes. Telescript was never a commercial success, but it did generate a lot of academic interest.

Since that time, this field has exploded in popularity, with a plethora of new frameworks and infrastructures appearing almost continually [16]. This profusion of experimental frameworks is reminiscent of the explosion of new programming languages in the early days of computing and is indicative of a new and immature research field. Many frameworks and platforms spring up, and are continually written about, but unfortunately, these papers remain qualitative and subjective in their nature. The dearth of quantitative results, however, means it has not yet been possible to fully evaluate the potential of either the technology or the paradigm. In the last year a trickle of results is beginning to validate some of the claims [23] [24], and these results are certainly important in establishing the credibility of mobile code systems. In addition, there has been work done to attempt to fully understand this new paradigm at the abstraction level [19], and develop metrics for system evaluation [22].

### 3 Issues Facing Mobility

Despite the increasing popularity of the concept, there are a number of key issues that the research community must face if it is to gain widespread acceptance and deployment. They remain fundamental issues that address questions such as:

- Is it safe? (Security)
- What's in it for me? (Payment)
- What good is it if I'm the only one? (Interoperation)

In the following sections we elaborate on these issues and review existing research that has been done in these areas.

#### 3.1 Security

Security is one of the most emotive issues raised when discussing mobile agent systems. It is often quoted [12] as the major reason mobile agent systems have

not taken off in the mainstream. There is currently a wealth of research being done on this particular subject [31]. The very nature and architecture of these systems enables new and interesting forms of attack on both host environments and agents in the system. Table 2 summarises the many and varied types of attack identified Lange and Oshima [14] and Jansen and Karygiannis [11]. In particular, the latter's report is extensive, and analyses security in mobile agent systems in quite some depth. The reader is referred to it for in depth coverage of the subject.

As with much in the mobile agent arena, progress in the area of security has been slow. If security has been considered in a platform at all, it is usually biased towards host protection, with much less thought given towards the integrity of an executing agent. Certainly, this approach may go some way to alleviating the fears of systems administrators considering hosting mobile agent environments. It does not however, allay the concerns of the user who has entrusted private information to their agent. Admittedly, protecting the mobile agent is a much harder task since agents are invariably interpreted within an execution environment, but there have been advances made in this area too.

Some agent protection can be gained through continual contact with the originating host. *JumpingBeans* [13] uses a client/server architecture where an agent returns to a secure central host before every migration where it can be checked for tampering. This approach is impractical and severely hinders, indeed breaks, the mobile agent abstraction. *Aglets* [2] enforces a trust based policy, whereby hosting environments will not accept or dispatch agents to remote hosts they do not trust. Both these examples are useful in controlled or closed situations, but are not practical when applied to open public networks such as the Internet.

Vigna [30] has proposed protecting agents through Action Tracing. As an agent executes at a host a log is made of any actions it performs there. These traces are non-repudiable, and allow the agent owner to check with a high degree of confidence whether there has been any tampering with the agent's state during its execution. The drawback to this approach is of course, the size of the traces could potentially be huge.

Riordan and Schneier [26] advocate a technique they term Environmental Key Generation, in which an agent is able to decrypt a predefined section of code upon a task specific trigger. The trigger itself causes the generation of the decrypting key, and thus, a malicious host is unable to inspect an agent's internal to ascertain what its response may be to a particular situation. The major drawback to this technique is that even trusted agent hosts are unlikely to allow an agent to execute a dynamically created piece of code.

Young and Yung [32] propose a technique known as Sliding Encryption, that enables mobile agents to perform encryption on any information it may accumulate at a host. This data is encrypted with a public key, so the host may not extract the text without access to the private key. On return to its origin, the agent is able to access the private key, and decrypt all of its stored data.

Finally, Sander and Tschudin [27] have made some progress in debunking the widely held view that because mobile agents are usually interpreted, the

**Table 2.** Common Security Issues identified in Mobile Agent Systems

Type	Example	Description
Agent/Host	Masquerading	Agent poses as another agent to gain access to services, information or resources at a host
	Denial of Service	An incoming agent may try to access and corrupt the host's local files, resources or even try crashing the server in a denial of service attack
	Unauthorised Access	Agent obtains access to sensitive information by violating ill implemented security mechanisms, or cached information
Host/Agent	Masquerading	Host assumes false identity in order to lure agent, so it may gain payment, or eavesdrop on the transactions of other agents
	Denial of Service	Host can completely ignore an agent's requests for resources, flood the agent with messages, etc
	Eavesdropping	Since agents are interpreted, host can inspect internal algorithms, trade secrets, c-card info, etc
	Alteration	Host can change essential internal data, or results from previous hosts, to e.g. Bias results returned to agent's owner
Agent/Agent	Masquerading	Agent assumes identify of another agent to extract sensitive data, e.g. credit card info, or extract a payment
	Denial of Service	Agent may flood another with messages, or distribute false information, or tie up an agents resources in another similar manner
	Repudiation	After agreeing to some contract, agent subsequently denies that any agreement took place
	Unauthorised Access	Agent interferes with another agent to gain access to sensitive internal data, or modify its behaviour
Other	Masquerading	Remote hosts and agents can act in concert to deceive another, to gain access to info, payment, etc
	Network D.O.S.	Agent may attempt to flood the network with messages or copies of itself, etc
	Copy and Replay	Once an agent has migrated, a host, or agent may attempt to replay the "pay agent" message or code repeatedly to steal money

hosting environment must be able to understand an agent it is hosting in its entirety. Whilst this assumption remains true for all plaintext data and programs, they have devised a mathematical technique that allows a mobile agent to safely compute cryptographic primitives in an untrusted computing environment. This process pivots around the ability to execute encrypted polynomial functions, without decrypting them in the first place. This is an extremely promising avenue of research, but to date there are few examples of functions that can be transformed in this manner. The single example described by Sander and Tschudin is of a function that enables digital signing with this technique.

### 3.2 Micropayments

In order for mobile information agent systems to achieve widespread adoption, there must be an incentive for hosting sites to allow these alien software programs onto their machines. This implies that hosts should be compensated for the additional load and security risks associated with supporting a mobile-agent platform. By providing a mechanism for the agents to carry limited amounts of some type of currency, they would be able to purchase computing services from public hosts. To date, most examples of mobile agent applications operate in a closed environment [20], and the variety of actual hosts the mobile agents migrate between are relatively low. Further, these applications usually feature mobile agents that are acting for a single user, or a party of mutually interested participants. Thus it is usually in the interest of the user population if their agents cooperate. In an open system, this will certainly not be the case as agents vie for limited computing resources, or compete to provide services.

The question of providing micropayment mechanisms in the information age is a concern in many fields of research. It is not even particularly new, with examples appearing in the late 1960's [28]. More recently, Huberman's [10] edited volume discusses many possibilities for computational economies. Indeed, the World Wide Web Consortium have a micro payments group who are actively developing protocols [35] and strategies for web based micro payment.

The question of resource allocation in a mobile agent system however, was first tackled in Telescript, where agents were given permits whose strength diminished over time as they travelled through the network. In the Messengers project, Tschudin [29] developed a mechanism for non discriminatory open resource allocation in which agents were able to purchase key computing resources such as CPU, memory and bandwidth. Each agent was treated as an equal and there was no weighting given to the particular task, destination or origin of the agent.

More recently, Bredin et al [5] have proposed a bidding strategy that minimizes an agent's execution time for a given itinerary, while preserving a fixed budget constraint. They have constructed a resource allocation policy where hosts take bids from agents for prioritized access to computational resources (CPU time). The priority of access to a resource an agent receives is proportional to its bid relative to the sum of all current bids at the host. Hosts collect revenues from each agent at a rate equal to the agents' bids.

To date, there are few full-scale implementations of micropayment schemes in mobile agent systems. Bredin's work, which is very promising, has been realised in a simulation but is yet to be transferred to a proper system. Ultimately, we require mechanisms that support resource owners in allowing agents controlled and metered access to resources and services, so that agents can traverse the globe, searching and paying for their data, interacting with other agents, and selling their services to sites or other agents. However, if this vision is to become reality, then we most certainly require infrastructures to support platform interoperability.

### 3.3 Interoperability

So far, most mobile agent platforms and applications have operated in a closed environment. By that we mean that all hosts and agent types in the system are known at the outset. They are all built from the same platform and thus inter-agent communication is solely with other agents from the system. In addition, the range of sites to which an agent travels is usually extremely limited. If we are to achieve a future of pervasive mobile information agents that is non-monopolistic, then we require mechanisms and infrastructures that enable interoperability between different manufacturers' platforms.

In the mobile agent arena, the biggest drive for interoperability has come from a joint submission to the OMG known as the Mobile Agent System Interoperability Facilities (MASIF) [8], which is a joint proposal by Crystaliz, General Magic, GMD Focus, IBM and The Open Group. The aim of the MASIF standard is to standardize such things as agent management, agent transfer, agent and agent system naming conventions, agent system types, and location syntax. The aim is to achieve a certain degree of interoperability between mobile agent platforms without enforcing radical platform modifications. MASIF is not intended as a reference model from which to build a new mobile agent platform. Rather, it provides specifications that can be used as an add-on to existing systems. To date there are a handful of mobile agent platforms that support the MASIF standard, for example Grasshopper [9], MAP [25], SOMA [3] and Aglets [2]. In reality though, the standard has failed to gain widespread adoption amongst platform implementers, due mainly in part to its reliance on many other OMG specifications, for example CORBA and IDL, and its bias towards many of the standard submitters' systems. In truth, after a flurry of activity in 1998, the MASIF standard's impetus has petered out.

More recently the SAFT [4] proposal by Blixt and Öberg encompassed the design of a mobile software agent framework that was based on existing standards as much as possible. The majority of the standards were drawn from the Java(tm) 2 Enterprise Edition and HTTP. Initially, the authors had hoped to provide a reference model specification that any platform implementer could follow in creating their product. Their objective was to allow the creation of many different products that all conformed to the same standard, as is seen in the web-server market. Unfortunately, as is the case with many thought experiments, many of the problems that appear are not immediately visible from

the outset. The authors have since implemented a prototype system and discovered that many of their specifications were unsuitable. However, they intend to update their specification with the experience they have gained from their work.

### 3.4 Commentary

From this brief review, it would seem that the state-of-the-art is relatively poor. In many respects this is true, for this is a fairly new research field that only a few years old. There are still debates as to the best methodology for actually achieving mobility in platforms, although the ubiquitous Java is rapidly becoming the language of choice. The proliferation of new mobile agent packages/platforms (currently listed at 72 by the Mobile Agent List [16]) points towards continued improvements in our understanding and experience from building infrastructures and the amount of reported implementations of real-world inspired systems is also increasingly. Indeed, the position does not fair too badly when considered in context. In the early years of programming, the number of new languages and extensions was initially prolific before common practices and techniques became widely accepted. We are currently moving through that phase in this research field. Things just take time. Remember the twenty years it took for objects to become commonplace?

We have briefly discussed some of the issues that face the mobile agent community if they are to achieve widespread adoption. There are others, for example adding intelligence to mobile agents, which have traditionally been very dumb; providing mechanisms for data description; and enabling true agent communication through the use of language. These are not solely the remit of the mobile agent community and thus, have not been covered in our discussion, but they are equally important.

## 4 Visions

It is our belief that in the future the network will become to be viewed as the computing environment of choice. Local computing will take place on PC's between users and static applications. The network however will be populated by thriving ecologies of agents. Mobile information agents will be able to travel from host to host, alighting where they wish to take advantage of a service or resource, before moving on to complete their goals elsewhere. These hosts are mere islands of resources in the vast expanse of the network. Upon them, languish static agents, rooted to the spot. They will act as interfaces to information sources, such as databases, file systems and web servers, and external devices, such as databases, printers and cameras. In return for providing these services, they will receive payment from the mobile agents, as will the host environments in return for providing essential resources such as CPU, memory and storage.

In the following sections we leave behind our review of the research field. Since this paper accompanies an invited talk, with a provided title, we have taken the opportunity to let our minds wander to the future, to see where mobile information agents can take us.

## 4.1 Mobile Agents in Space

The launch of the Deep Space One (DS1) probe heralded the beginning of a new phase for NASA, which will be embarking on a series of missions to test and deploy low cost, demonstration technologies. In an attempt to cut mission costs from billions to under 100 million, NASA is looking towards autonomous spacecraft and robots to establish a virtual presence in space. The increasing motives for space exploration, and the diversity of the environments and missions being encountered require machines that are both robust mechanically, as well as computationally. The recent paradigm shift from billion dollar missions with massive ground crews, to cheaper, targeted missions with much smaller ground crews has required the creation of spacecraft that are largely controlled by intelligent and self sufficient entities, known as Remote Agents [17].

Current research is investigating how to control the interaction between agents, their internal sub-systems and humans. Adjustable autonomy is of particular interest when dealing with systems that may have to function for several years unattended [7], or may be forced to halt their mission until they can get further instructions from human operators. However, the increasing distances and time delays involved with space exploration make remote operation of robotic probes and vehicles logistically impossible. New methods to support assuming full control of a probe, or subtle alteration of its operating constraints, are required that can overcome amongst other things, the problem of long distance interaction.

Even if the future brings the reality of unlimited bandwidth (and that is unlikely - see Section 2) we may discard it in this example since the network we are considering is extremely vast, i.e. a network of spacecraft distributed across the solar system and beyond. With a network this large, immutable factors such as the speed of light will come into effect, that can not be overcome with any amount of bandwidth. Thus, local agent interaction via migration through the network rather than traditional client/server or asynchronous messaging must have a role to play in the future of our virtual presence in space. By now, it should be clear that mobile agents are inherently suited for long distance interaction.

To examine the use of mobile agents in this scenario we will place them in a hypothetical context. Their particular strengths of being able to react to situations that may have been unforeseen at the time of their dispatch makes them a viable option for interacting with deep space probes, missions into the farther reaches of the solar system, and planetary ground teams (be they human or robotic). Let us imagine that we have moved into the near future, and the solar system has been populated with a network of probes, spacecraft and robots. Circling the moon Europa is a small transport ship from which a group of hydrobots are sequentially dispatched to search for life under the ice rafts. At some point, one of the hydrobots returns to the surface with what looks like some very exciting results. On arrival at Earth, the results cause several scientists to get extremely excited and they pull an all-nighter devising a mission to further examine these results. The command sequence is dispatched to the transport

ship, which receives and programs one of the remaining hydrobots with the new mission. The hydrobot is launched and carries out the new mission.

This scenario seems plausible enough, but what is the effect of new results emerging whilst the mission is in transit? Perhaps the initial hydrobot happens to run a routine diagnostic and discovers that one of its sensors was damaged and reports from it have been skewed. In the mobile code instance (the command sequence), the second hydrobot would probably still be launched and might in fact be wasting valuable resources investigating a worthless area. With a mobile agent, the agent would update its knowledge base on arrival with any reports it had missed en route and deduce that the mission it was tasked with was not required. The decision could be made to terminate itself, or perhaps continue with an alternative mission plan. In either case, a potential waste of a valuable hydrobot or other irreplaceable resources would be avoided. It can be argued that the sensor check should have been done before the report transmission, or that conditional command sequences might alleviate some of the problems with long distance interaction, but Murphy's Law may rear its head at any time. Unforeseen circumstances are a reality in space exploration. Mobile agents offer a solution to long distance interaction by providing a flexible, reactive approach to re-tasking probes and robots in-situ rather than a less flexible approach.

Ultimately, we envisage a vision of space inhabited by an ecology of robots, satellites, spacecraft, rovers, planetary bases and the like. In contrast to most visions, these entities act not only as fundamental actors in the vision, but as a network of resources. In it, each entity has some level of autonomy, which may range from very simple task specific instructions, to more complex autonomous agent architectures. Mobile agents live in the network, able to migrate, clone, sleep, wake, but in reality insert a higher layer of control and abstraction over the underlying hardware and software.

In truth, it will be some time before sufficient numbers of these entities inhabit space to make the vision reality. However, the advantages of local interaction, adjustable autonomy and interaction at a distance make mobile agents a particularly useful technology even with a single spacecraft. These ideas have been aired before [18].

## 4.2 Self Tuning Networks

As we have seen, the nature of the network is changing and we require new software architectures for exploiting these new frontiers. Nearly all contemporary computer networks consist of a mixture of disparate hardware, operating systems and other computing devices, the topology of which is continually changing. Current methods for resource location revolve around a centralised registry that participants in the network may query to locate required services. This type of architecture quite clearly can not work in the networks of tomorrow. In this section, we present a mobile agent based architecture for resource discovery that is completely decentralised, and further, is able to tune itself in response to changing demands placed upon it.

At each node in a network sits a static DirectoryAgent. The role of this agent is to maintain a local list of other known hosts, along with a record of the services and resources available at that host. DirectoryAgents communicate exclusively with local agents, they do not communicate across the network. In addition, the system contains Roamers, agents that travel the network, alighting at nodes. When they come to rest at a node, they briefly interact with the local DirectoryAgent before moving on to another node. Again, Roamers do not communicate across the network.

The interaction between the DirectoryAgents and the Roamers is the essential aspect of this system. At each node, the Roamers update DirectoryAgent's registry of known hosts, resources and services. In addition, it makes a note of the resources and services available at the current node, before moving on to another host. Which host they choose to go to next, is chosen at random. To avoid Roamers becoming too bloated, there is mortality in the system. Roamers die before becoming too old, or after making a certain amount of hops.

To avoid the information stored in a registry becoming out of date, the records that a Roamer carries are inspired by ant pheromones, in that they become weaker over time. As soon as the Roamer departs the current node, the newly acquired record begins to decay. If a Roamer alights at a node whose DirectoryAgent's registry contains a more recently record, it is able to update the one it carries with the new information.

The novel part of this system is that Roamers are created by the DirectoryAgents themselves. If a DirectoryAgent hasn't received a visit from a Roamer for a certain amount of time, it creates a new Roamer and dispatches it out into the network. In addition, to avoid clogging the network, DirectoryAgent's are intelligent enough to be able to respond to the time since a last visit. Eventually the network of nodes could tune itself with respect to how many Roamers exist within the system, and how frequently they must be generated. Further, each DirectoryAgent can tune itself in direct response to its situation, and is unlikely to create Roamers at exactly the same time as a near neighbour. For example, a DirectoryAgent located on the core network may never need to create any Roamers! In effect, the network becomes an emergent ecology that tunes itself self referentially.

We can further expand this vision to consider how large-scale networks and subnets can interact. If a Roamer was able to do a lookup on the node to which it was next heading, it would be able to discover if that node was on the same subnet, or LAN. Depending on what type of Roamer it was, it could determine whether to venture out of the local network, or choose another node to visit. At each join of a network, a special type of DirectoryAgent, a GateKeeper, sits whose role it is to be a local resource for knowledge of location of services outside the current catchment area (ie. Subnet or LAN). The GateKeeper could even be mobile itself, in case some of the nodes needed to shutdown. It would not matter, as the Roamers will find it in the end, and its new location will trickle through the network eventually. The last additions to the family of agents are

Trackers, whose role it is to specifically locate a particular resource, wherever it is, and return with the information.

While this vision seems far fetched, it illustrates some fundamental principles that we must adopt if we are to build software architectures of the future. Decentralisation ensures that our networks can reliably scale, without major bottlenecks occurring certain nodes. In addition, redundancy in the system ensures that failure of any single node does not compromise the integrity of the entire system.

## 5 Conclusions

In this new era of pervasive computing we require new paradigms for building software systems that can support users in the Information age. Users require tools that can locate and filter information for them, presenting it at their convenience, wherever they connect to the network. This paper has reviewed the state-of-the-art of Mobile Information Agents, a technology that promises to fulfil many of these requirements. We have found that the technology remains youthful and that many of the essential issues for enabling this future have yet to be truly realised, although there is much promising progress in understanding them.

In the latter sections we have presented visions of where this technology may take us. Ultimately, this paper presents a vision of the network inhabited by an ecology of agents, representing a myriad of users and devices. The ecology is vibrant and dynamic, with its topology changing frequently. In contrast to most visions, all the agents are not static, but consist of a hybrid mixture of mobile and static agents. Static agents act as brokers for resources and services, in addition to performing more traditional roles as proxies for their owners. Mobile agents live in the network, able to travel, clone, sleep, wake, and autonomously go about fulfilling their owner's requirements.

## References

1. Adobe Systems Inc.: The Postscript Language Reference Manual. Addison-Wesley, 1985.
2. Aglet Software Development Kit: IBM, <http://www.trl.ibm.co.jo/aglets>
3. Bellavista, P., Cavallari, C., Corradi, A., Stefanelli, C.: Mobile Agents for Internet Services: Directions of Standardization and their Implementation in SOMA, Proceedings of the 37th Conference of the Associazione Italiana per l'Informatica ed il Calcolo Automatico (AICA'99), Abano Terme, Italy, pp. 19-31, September 27-29, 1999.
4. Blixt, K-F., Oberg, R.: Software Agent Framework Technology. Master's Thesis Linköping University, 2000. Available at <http://www-und.ida.liu.se/~karbl058/saft/>
5. Bredin, J., Maheswaran, R.T., Imer, T.B., Kotz, D., Rus, D.: A Game-Theoretic Formulation of Multi-Agent Resource Allocation. Proceedings of Autonomous Agents 2000, Barcelona, 2000.

6. Chess, D., Harrison, C., Kershenbaum, A.: Mobile Agents: Are They a Good Idea?. In: Vitek, J., Tschudin, C.: Mobile Object Systems, Towards a Programmable Internet. LNCS Vol 1222, Springer-Verlag, 1997.
7. Dorais et al: Adjustable Autonomy for Human Centered Autonomous Systems on Mars. Proc. of the First International Conference on Mars Society, August, 1998.
8. GMD FOKUS, IBM Corp: Mobile Agent Systems Interoperability Facilities Specification. OMG TC Document, available at <ftp://ftp.omg.org/pub/docs/orbos/1997/97-10-05.pdf>.
9. GMD FOKUS: Grasshopper Platform, <http://www.ikv.de/products/grasshopper/overview.html>
10. Huberman, B.A., (ed): The Ecology of Computation. Elsevier, 1998.
11. Jansen, W., Karygiannis, T.: NIST Special Publication 800-19 - Mobile Agent Security. National Institute of Standards and Technology, 2000.
12. Johansen, D.: Interview in: Milojevic, D.: Trend Wars: Mobile Agent Applications. IEEE Concurrency, pp 80-90, July-September, 1999.
13. JumpingBeans, Ad Astra Engineering Inc, <http://www.jumpingbeans.com>
14. Lange, D, Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, 1998.
15. The Mobility Mailing List. De facto mailing for discussion of mobility. Home page at <http://mobility.lboro.ac.uk>
16. The Mobile Agents List, a repository of mobile agent systems, available at <http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole/mal/>
17. Muscettola, N., Nayak, P.P., Pell, B., Williams, B.C.: Remote Agent: To Boldly Go Where No AI System Has Gone Before. Artificial Intelligence 103(1/1), August 1998.
18. Papaioannou, T.: Mobile Agents: Are They Useful for Establishing a Virtual Presence in Space?. In: Agents with Adjustable Autonomy Symposium, part of the AAAI 1999 Spring Symposium Series.
19. Papaioannou, T.: On the Structuring of Distributed Systems: The Argument for Mobility. PhD thesis, Loughborough University, 2000.
20. Papaioannou, T., Edwards, J.M.: Using Mobile Agents To Improve the Alignment Between Manufacturing and its IT Support Systems. Journal of Robotics and Autonomous Systems, Vol 27, pp 45-57, 1999.
21. Papaioannou, T., Edwards, J.M.: Manufacturing System Integration and Agility: Can Mobile Agents Help?. To appear in Jan 2001 Special Issue of Integrated Computer-Aided Engineering, IOPress.
22. Papaioannou, T., Edwards, J.M.: Towards Understanding and Evaluating Mobile Code Systems. To appear in forthcoming special issue of Journal of Autonomous Agents and Multi-Agent Systems.
23. Papastavrou, S., Samaras, G., Pitoura, E.: Mobile Agents for WWW Distributed Database Access. Proceedings of IEEE International Conference on Data Engineering (ICDE99), 1999.
24. Picco, G.P., Baldi, M.: Evaluating Tradeoffs of Mobile Code Design Paradigms in Network Management Applications. In: Kemmerer, R., Futatsugi, K. (eds): Proceedings of 20th International Conference on Software Engineering (ICSE'98), Kyoto (Japan), IEEE CS Press, 1998.
25. Puliafito, A., Tomarchio, O., Vita, L.: MAP: Design and Implementation of a Mobile Agents Platform. Journal of System Architecture. to be published.
26. Riordan, J., Schneier, B.: Environmental Key Generation Towards Clueless Agents. In [vigna98], 1998.

27. Sander, T., Tschudin, C.F.: Protecting Mobile Agents Against Malicious Hosts. Appears in [vigna98], 1998.
28. Sutherland, L.E.: A futures market in computer time. CACM, Vol. 11 (6), 1968.
29. Tschudin, C.F.: Open Resource Allocation for Mobile Code. In Proceedings of The First Workshop on Mobile Agents, Berlin, 1997.
30. Vigna, G.: Protecting Mobile Agents through Tracing. Proceedings of Third ECOOP Workshop on Mobile Object Systems, Jyväskylä, 1997.
31. Vigna, G., (ed): Mobile Agents and Security. LNCS 1419, Springer-Verlag, 1998.
32. Young, A., Yung, M.: Sliding Encryption: A Cryptographic Tool for Mobile Agents. (ed) Eli Biham, Proceedings of the 4th International Workshop on Fast Software Encryption, FSE'97, January 1997, LNCS 1267, Springer-Verlag, 1997.
33. White, J.E.: Telescript technology: the foundation for the electronic marketplace. White Paper, General Magic Inc., Mountainview, Sunnyvale CA, USA, 1994.
34. White, J.E.: Telescript TEchnology: Mobile Agents. In: Bradshaw, J. (ed): Software Agents. AAAI Press/MIT Press, 1996
35. World Wide Web Consortium, Micro Payment Transfer Protocol (MPTP) Version 0.1, Nov 1995