Mobile Agents for Network Management

Andrzej Bieszczad
(andrzej@sce.carleton.ca)

Bernard Pagurek
(bernie@sce.carleton.ca)

Tony White
(tony@sce.carleton.ca)

Systems and Computer Engineering, Carleton University
1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6

Abstract

In this paper, we discuss the potential uses of mobile agents in Network Management. We define software agents and a navigation model, which determines agent mobility. We list a number of potential advantages and disadvantages of mobile agents. We include a short commentary on the ongoing standardization activity. The core of the paper comprises descriptions of several actual and potential applications of mobile agents in the five OSI functional areas of Network Management. A brief review of other research activity in the area and a brief prospectus for the future conclude the presentation.

Keywords: network management, mobile agents, plug-and-play networks

I Introduction

1 The need for agents in managing networks

There is a trend towards increasingly heterogeneous networks in today's communications environment. Such diversity requires that network operators have greater knowledge and increased training. Managing these diverse networks requires the collection of large quantities of data from the network; data that must then be analyzed before management activity can be initiated. Simultaneously, there is an expectation of increasing reliability and quality of service by users of today's networks. These challenges are the main forces driving research on software agents. Most of the research on the intelligence aspects of software agents comes from *Distributed Artificial Intelligence* (*DAI*) (O'Hare and Jennings, [1]). Distributed AI is an extension of ideas from Artificial Intelligence that applies to *Multi-Agent Systems* (*MAS*) (Lesser, [2]). Instead of one centralized and usually very large application that encodes the complete intelligence of the system, a number of relatively small systems, or agents, are involved in a cooperative effort to resolve a problem. This does not imply that the large system is merely divided into smaller pieces. For example, several centralized applications, each capable of addressing a certain aspect of a problem, can be tied together by a communication system. It would allow for exchange of their viewpoints and coming up with strategies to make progress or to combine the results into a solution. This kind of problem solving is called *Distributed Problem Solving* (*DPS*) (Decker, [3]) and each of the cooperating systems may be considered an agent. In Artificial Intelligence, an agent is viewed very often in terms of its *beliefs*, *desires* and *intentions* (so-called *BDI architecture*) (Georgeff and Lansky, [4]).

Although an agent-based system can be implemented with any client/server technology, it differs from classical client/server systems, because there is no clear distinction between a client and a server. All agents participate in the computation according to the role statically assigned by the designer or dynamically assigned by a human or agent supervisor. In this paper, we argue that agent mobility benefits agent-based network management systems with additional advantages over classical client/server (i.e., manager/agent) implementations, as well as systems built with stationary agents.

2 Organization of the paper

In Section II, we briefly introduce software agents and mobile agents. Furthermore, we present a list of potential advantages of the use of mobile agents. The section concludes with an overview of an emerging standard, the Mobile Agent Framework (MAF) [5]. In Section III, we present a number of applications of mobile agents in the OSI functional areas of Network Management ([6]). These applications are being developed as part of the ongoing research conducted within the Perpetuum Mobile Procura (PMP) project at Carleton University (Bieszczad et al., [7]). Each of them illustrates how mobile agents can help to overcome or ease the problems of management systems based on stationary components. We conclude the paper with a review of related research activity conducted elsewhere and a prospectus for the future.

II Mobile agents

1 Definition of software agent

It is a challenge to provide a definition of an agent that would not be controversial (Nwana, [8]). One possibility is a definition of a software agent as a computational entity, which acts on behalf of others, is autonomous, proactive and reactive, and exhibits capabilities to learn, cooperate and move. This definition has its roots in the concurrent Actor model (Hewitt, [9]). We will call these basic characteristics a *basic agent model*.

A *mobile agent* is a software agent that can move between locations. This definition implies that a mobile agent is also characterized by the basic agent model. In addition to the basic model, any software agent defines a *life-cycle model*, a *computational model*, a *security model* and a *communication model* (Green et al., [10]). A mobile agent is additionally characterized by a *navigation model*.

Mobile agents can be implemented using one of two fundamental technologies: *mobile code* ([11]; Baldi et al., [12]) or *remote objects* (Vinoski, [13]). Examples of the former approach include AgentTCL (Gray [14], [15]) and Telescript (White, [16]), and the latter - Aglets (Lange, [17]).

To make use of mobile agents, a system has to incorporate a mobility framework. The framework has to provide facilities that support all of the agent models, including the navigation model. For the life-cycle model, we need services to create, destroy, start, suspend, stop, etc., agents. The computational model refers to the computational capabilities of an agent, which include data manipulation and thread control primitives. The security model describes the ways in which agents can access network resources, as well as the ways of accessing the internals of the agents from the network. The communication model defines the communication between agents and between an agent and other entities (e.g., the network). All issues referring to transporting an agent (with or without its state) between two computational entities residing in different locations are handled by the navigation model. Obviously, the framework incurs certain costs including increased memory requirements and execution and access delays on every participating device. The underlying technology, however, is evolving rapidly. For example, the footprints of certain Java Virtual Machines (JVM), which are the basis for many mobile agent frameworks, are very small making them suitable for embedded systems ([18]). We believe that the use of Java chips will be important in the future networked devices. In addition, forthcoming new software packages like Jini ([19]) address many of the needs of agent-based systems.

The size of mobile agents depends on what they do. In swarm intelligence (White et al., [20]), the agents are very small. On the other hand, configuration or diagnostic agents might get quite big, because they need to encode complex algorithms or reasoning engines. Note however, that agents can extend their capabilities on-the-fly, on-site by downloading required code off the network. They can carry only the minimum functionality, which can grow depending on the local environment and needs. This capability is facilitated by code mobility.

2 Advantages of mobile agents

The use of mobile agents may have advantages over other implementations of agents. This does not imply that other technologies (like remote objects) cannot be used instead, because virtually any task that can be performed with mobile agents can also be performed with stationary objects. However, the traditional solutions might be less efficient, difficult to deploy, or awkward.

Table 1 (based on Green et al., [10]) contains the areas that *may* benefit from appropriate use of mobile agents instead of, or in addition to, classical client/server models.

3 Mobile Agent Framework (MAF) – An emerging standard

As with any other communications-related activity, the general acceptance of mobile agents for network management activity will depend heavily upon standards. The Open Management Group (OMG) has already begun work in the area of mobile agents, and a draft standard has been tabled for discussion (OMG, [5]; Cheng and Covaci, [21]). The proposed standard attempts to be platform neutral and has each chunk of mobile code identify itself with a language, or execution environment requirement. The proposal identifies the need for mobile code regions, with gateways between them, that provide an agent application virtual layer on top of the actual network. This architecture is shown in Figure 1. An agent region is defined as a set of agent systems that can access each other, possessing similar authority and identifying a default migration pattern. Mobile agent facilities include the storage and retrieval of agents, remote agent creation transfer and agent method invocation. The draft standard also draws heavily on CORBA, with IIOP being used as the transport protocol, and hinting that many pre-defined CORBA services, such as naming, may be used to support mobile agent activity.

Two MAF objects and their interfaces are defined in the specification; the MAFAgentSystemInterface and the MAFFinderInterface. The MAFFinder provides a naming service for agents, one MAFFinder (at most) being provided per region. The MAFFinder is registered as a CORBA object, the intention being that one agent, or MAFClient, may locate, and communicate with, another agent. The MAFAgentSystemInterface provides standard management operations for agents, such as receive, create, suspend and terminate. The specification provides interface definition language (IDL) details on agent naming, authority (or in CORBA terms, principal) and type; together these being used to generate a globally unique agent name.

While the MAF specification is a useful starting point, it is limited in scope. No notification services are defined, security is briefly mentioned and the MAFFinder, in particular, has an impoverished interface. For example, the query, "List all agents of type X" cannot be asked.

III Mobile agents for Network Management

Network management systems have to deal with the very issues that are driving research on software agents: proliferation of data and heterogeneous environments. The problem with information flooding a network in the case of malfunction is particularly severe if we take into account that a solution has to be found quickly. The fault has to be diagnosed quickly and fixed automatically or a human operator needs to be informed and advised as to the proper course of action. In large networks, operators have to interact remotely with many devices from the managing workstation. To accommodate the diversity of network components, management applications incorporate large numbers of interfaces and tools. Network management systems are usually huge monoliths that are difficult to maintain. We will review a number of application areas that illustrate how mobile agents can be used and how they may help.

1 OSI management functional areas

Many tools are required to address all aspects of managing a communication network. The OSI management model recognizes that fact by categorizing these requirements into several functional areas. They are:

- fault management
- accounting management
- configuration management
- performance management and
- security management.

We will analyze several potential applications in each of these areas. We start with network modeling, which we consider necessary for many network management functions.

2 Network modeling

In Network Management, automatic discovery is one of the fundamental functions of the management system. We use an unqualified term on purpose, because discovery might target many goals. In the simplest case, just finding the devices of the network is of interest. An extended version

of discovery is concerned with the construction of more detailed views that may include, for example, services available on a device or devices that satisfy certain constraints. If the constraints are functions of device status, then we approach problem discovery. As the complexity of discovery grows, it is harder to implement using classical client/server approaches.

Mobile code is a convenient vehicle for performing discovery tasks (Schramm et al., [22]). While basic network discovery (node discovery alone) is not a convincing justification for the use of mobile code, its more sophisticated variations certainly do benefit from the new capabilities. Nevertheless, the basic discovery of network devices is an excellent vehicle to illustrate the approach. One of the commonly used discovery techniques is sending *ping* messages to IP addresses in a certain domain. The discovering process builds its view of the network from the received responses. Instead, a mobile agent (a *deglet* – after a *delegation agent* (Bieszczad and Pagurek, [23])) can be created with a sole task of sending the identifier of a visited node to the creator (Figure 2). The deglet is then injected into the network and travels by the means of the implemented migration patterns (Susilo et al., [24]). There are several similar issues to resolve as in the *ping* algorithm (e.g., the scope of the search), but the method is more flexible. For example, it does not require that the interlocutor have knowledge about the network. The termination of the task can be determined heuristically inside the deglet, for example, by counting the hops or an average number of visits of a particular node.

By incorporating constraints in the discovery deglet, we can create partial network models (White et al., [25]). For example, constraints on the type of the devices in the discovery deglet will result in a network model consisting only of devices of a certain type. If the constraint is on the utilization of a node, then we may model over-utilization problems. These network models can be created dynamically. They can be tailored to meet expectations of a requesting application.

If the delegated task has a permanent nature, then we call the agent a *netlet* after a network agent. Netlets are considered a part of the network infrastructure. The discovery process becomes ongoing with the use of discovery netlets. The network model can be maintained dynamically, because the netlet can discover the changes to the network configuration. A number of netlets might be assigned to perform the task. The speed with which changes are detected can be controlled by the density of netlets; the more netlets in the network, the shorter the detection process is. Of course, there are

certain constraints that define an upper bound on the number of netlets; e.g., throughput. Netlets can use the default migration policies implemented by the migration facilities. Alternatively, they may implement their own migration patterns. For example, a netlet may define the scope of the coverage, so it never leaves a particular sub-network.

3 Fault management

3. 1 Network diagnosis

The same principles as we saw in network modeling can be used to diagnose network faults. Detection of faults is a process of building a specialized model of the network. For example, a simple deglet performing selective discovery of nodes with utilization that exceeds a certain threshold builds a model of over-utilized nodes. If the constraints on discovery describe violations of what is considered normal behavior of network elements, then the agents testing the constraints perform a fault detection function. Either deglets or netlets may be used. Deglets can be used in reaction to discovered or suspected problems. They might try to collect additional information, perform enhanced tests or execute a recovery routine.

The constraints do not have to be confined to a single network device. They may encode complex fault detection and correlation algorithms (El-Darieby, [26]). The only limitation is the size of the netlet with a direct impact on the efficiency with which it can migrate and on network throughput. That issue can be minimized by taking advantage of the expressiveness of Java (Gosling et al., [27]), code compression techniques and intelligent methods. The theoretical foundations supporting engineering of mobile agents from the performance perspective are also being researched (Baldi and Picco, [28]).

On the other end of the size spectrum, we may have societies of small, biologically inspired and relatively simple agents that need to cooperate to deliver the intelligence needed for diagnosing network faults (White et al., [29]). A number of types (or species) of such tiny agents are usually injected in the network. Each type can address one aspect of the problem, and that aspect is resolved by reinforcing the given hypothesis by the observations of a large number of the same agents. The

solution to the problem emerges through the integration of the hypotheses of each species (White et al., [20], [30]).

Both deglets and netlets that comply with certain security provisions might be allowed to perform actions on network devices (Figure 2). Such *active* netlets can be used to address problems autonomously leading to an immediate recovery if such an action is possible. The network manager will either be informed about the event or will be alerted if an automatic recovery is not possible or requires human involvement. A number of such specialized network repair agents might provide a high degree of network immunity to a range of problems. In the context of deglets, tasks are delegated as needed by an interface agent interacting with the delegating entity. In contrast, netlets can be assigned their tasks *á priori* by their designers and start automatically as an integral part of the networking infrastructure. The texture and density distribution of societies of netlets can be controlled by certain security mechanisms. For example, the frequency of visits can be measured and used to generate or terminate agents (Bieszczad and Pagurek, [6]).

3. 2 Remote maintenance of heterogeneous elements

Mobile agents need to interact with hosting nodes through an interface that provides secure, indirect access to the host resources and a number of other services. In the Perpetuum project this interface is called Virtual Managed Component (VMC) (Susilo et al., [24]). A VMC is to be designed and implemented by the vendor of the network device. It may reside on the actual device or on a proxy if the device is not capable of accepting mobile agents. One of the facilities that vendors may include in the VMCs of their devices is a special downloadable data presentation applet, which makes managing heterogeneous environments easier. When a device is connected to the network, knowledge about the available facilities is disseminated to interested parties residing in the network. For example, a discovery agent may request a list of supported services. A user browsing a list of devices may need to examine a particular device (Figure 4). If the device is registered as one that provides its own facility for examination of its state, then the applet is retrieved from the selected item and executed locally on the manager's workstation. It may execute inside a Web browser or as a standalone application. The applet provides data presentation and interactive functionality in the way that the

vendor of the device considers as the most suitable for this particular device. For example, it may be a simple textual list or a sophisticated graphical representation of the hardware and/or services.

If a device does not provide a presentation applet, then either the manager has a proper handler for the device or a generic device browser will be used instead. This is a solution widely used in the legacy device agents of today's network management systems. A device not only has to have its representation in the manager implemented, but it also has to implement a communication protocol such as SNMP (Case et al., [31]; Case and Levi, [32]) with the manager, which allows for the transfer of all device parameters. This is what makes today's managing systems so big and inflexible. The solution with downloadable applets does not have such constraints.

An additional attraction to this approach is provided by maintaining a repository of applets on a server, for example, on the vendor's web server. The VMC shipped with a network device would include only a reference pointer to the remote applet, which would be brought from the server when needed. An obvious advantage is that the vendor can maintain its repository up to date and structure it conveniently, for example, with the use of various software components. This approach might be a part of a more generic scheme for plug-and-play components. In fact, structuring the vendor's server as a mobile code environment with a VMC implementing the storage of device behavior seems attractive.

Another example of the use of mobile agents for remote maintenance addresses the problem of taking care of on-site devices that provide certain services at customer premises; for example, in a network for on-demand video delivery (Mennie, [33]). In such future networks, it will be impossible to send a technician to test every faulty device, because there will be hundreds of thousands or more of them installed. A mobile agent will be sent instead. It may perform a suite of tests and attempt to repair the device if possible. Only if that fails will a human operator be involved. The agents can incorporate machine learning techniques, which may improve their future behavior. A library of agents can be established with an automatic selection of agents for specific tasks.

4 Configuration management

4. 1 Service provisioning

Provisioning services in telecommunication networks is a complex process, which usually involves several parties. Mobile agents can help to streamline the process. This fact has been recognized by agent-related activities of organizations working towards service provisioning standards; e.g., the Telecommunication Information Networking Architecture Consortium, TINA-C. In spite of aggressive popularization of ideas (Magedanz et al., [34]), the actual work seems to progress less vigorously. At this moment, very little information on the progress is available to non-members, but the results will be made public in 1999. The research on active nets and switchlets can also be seen as an attempt to provide a comprehensive and flexible service provisioning platform based on code mobility, albeit at a lower level (Tennenhouse et al., [35]).

To illustrate some of the opportunities for using mobile agents, let us analyze an example. Provisioning permanent virtual circuits (PVCs) in an ATM network is an example of service provisioning. It may take a long time to negotiate all aspects of a PVC that needs to be established between two ATM switches, especially if from the two switches are from different vendors. Very often, a PVC includes a path through a network cloud maintained by a network operator. The process would be even longer if another operating company offered similar services but with different prices. In a heterogeneous network, there could be several network managers involved in the process. There is no standard protocol to achieve this task.

A system based on mobile agents (Figure 5) can handle similar tasks in an autonomous way (Pagurek et al., [36]). A request to set up a PVC can be assigned to a deglet, which coordinates the overall process. It uses additional deglets to perform partial tasks. Using these deglets, all of the necessary data is exchanged by the endpoints using the provisions incorporated into their respective Virtual Managed Components (VMCs). The deglets communicate with VMCs using a special ontology that generalizes the knowledge of setting up cross-connects and PVCs by the vendor. Parts of the necessary data may be brought as required from remote locations; for example, vendor's web sites. Then, another deglet negotiates with the VMCs in the operator's clouds. The best deal is selected and the subsequent exchange of necessary information completes the setup. At that point, the requesting party is informed that the task has been accomplished.

4. 2 Component provisioning

Configuring a device requires that a number of attributes in the network and on the device be set as well as certain software components be installed. For example, a printer requires that its drivers be present on workstations that will be using it. Currently, the manager of the network has to perform all required tasks manually. Mobile agents can be used to implement plug-and-play network components.

As an example, consider a task of installing a network printer. This task involves an established enterprise network of a number of workstations with potentially different requirements. For example, the drivers that are required to properly interact with a printer do differ between a Macintosh and Unix computers, or between PCs running Windows NT and OS/2. Let us assume further that the network is connected to the Internet. To make proper use of the printer, the workstations have to be provisioned by installing proper drivers. If the printer came with all required drivers or if the operating system included all of the drivers, then this part of the task might be tedious, but manageable. On the other hand, if there are no proper drivers, then the whole process becomes very difficult. The dynamics of the network topology have to be handled as well since new printers and workstations are added, printers are taken out of service, operating systems are updated, etc.

The scheme illustrated in Figure 6 is based on the use of several mobile and stationary agents. It can be used to provide plug-and-play capabilities for network components (Raza, [37]). The VMC, which comes with the new device, a printer in our example, includes a bootstrapping provisioning agent. Upon connection to the network, a number of deglets or netlets are sent to discover the network devices that will need printer drivers. Then, the web page of the printer manufacturer is contacted, so the latest versions of the required drivers can be downloaded. A printer's VMC may maintain a list of the devices that use the printer, so it can coordinate installations of new drivers as needed. The printer registers with the vendor, so if a new version of a driver is available from the vendor, then it is automatically sent to the printer's provisioning agent.

As in the case of agents for fault diagnosis, configuration can be a result of cooperative behavior of large numbers of simple agents (White et al., [29]; Dorigo, [38]).

5 Performance management

Certain aspects of measuring the performance of networks are difficult if a centralized server is used. Network delays make measurement precision questionable. Instead of remotely polling network elements, a mobile agent can be dispatched to perform an analysis of the component locally. The information collected in this way is more accurate, because there are no delays involved whatsoever. A similar improvement can be achieved with a monitoring process being a part of the tested element, but that requires that the process be a static part of the local system. The solution with a mobile agent is superior, because it does not require permanent consumption of local resources. It is far easier to manage from the maintenance perspective, since we can always use the latest version of a mobile deglet. In contrast, updating a static local monitoring agent is a longer process.

Hot-swapping technologies can also be based on mobile code. Hot-swapping schemes can be applied to keep stationary network monitoring agents up to date. If the local monitoring agent is a dynamically extendable application (for example, a Java application), then modules comprised of mobile code (extlets) can be used to upgrade the application whenever necessary without distracting the services that the application provides. The mobile module does not have to reside on the local element all the time, because it could be downloaded whenever it is needed, not unlike the way applets are downloaded in a Web browser. The schemes with actual mobile agents are still better, because they require fewer resources, and additionally provide distributed processing and intelligent correlation of facts from different sources in the measurement process. For example, a managing application may use a deglet to locate an under-utilized node, which could take over certain services.

Server migration, where a server is moved or cloned to a better execution environment, can be considered an example of a hot-swapping scheme. The decision to move or clone requires intelligent analysis of a number of factors such as service demand, network load, failure rate, etc. For example, a mid-level manager may be migrated to another location, if the failure rate in accessing its services or network latency is unacceptable. In another case, a server providing communication services in a distributed system (e.g., a blackboard coordinator) might be moved or cloned if the requirements for its services is on the rise in certain areas. In all these cases, the server is, in effect, a mobile agent.

There is no constraint on the size of an agent in the definition, although usually we think about a mobile agent as something small.

In the context of network performance, we need to raise the issue of potential performance problems from the use of mobile agents. Taken to the extreme, uncontrolled mobile agents could flood the network taking over a large proportion of its resources. The infrastructure for mobile agents enforces certain rules on the density of agents. For example, not everybody is allowed to inject mobile agents into the network. Those who are authorized to do so (human operators or system applications) have to provide information on what constitutes a normal pattern of behavior for their agents. If those rules are violated, then the infrastructure provides counter-attack mechanisms or, if the problem is persistent, the agents are refused services. The former may involve density thresholds and a license to generate and kill agents; the latter translates to extinction of agents through refusal to migrate them.

6 Plug-and-play networks

Mobile agents bring us closer to the ultimate network: a plug-and-play network (Bieszczad and Pagurek, [6]). A plug-and-play network can automatically self-configure to accommodate component and user requirements. We have seen examples of mobile agents performing such functions as configuring network components and providing services. In another example, an applet was used to adapt the presentation of network data to the profile of the device and the profile of the user. If the components or the needs of a plug-and-play network evolve, then the network can detect the change and modify the configuration data. Discovery and provisioning agents in our examples provided such capabilities. A plug-and-play network can detect problems that may potentially affect its integrity resulting in deteriorating quality of service or compromised security. If a detected problem can be addressed automatically, then a plug-and-play network can repair itself autonomously. Again, we saw examples of netlets that can incorporate intelligence necessary to discover, correlate and address emerging network problems. A plug-and-play network incorporates a middleware layer that provides a range of services to the overlying network applications. In our examples, we described netlets providing network modeling and performance measurement services.

IV Research activity

There are a growing number of centers conducting research on mobile agents. Such pioneers in providing agent frameworks as GeneralMagic (Telescript and newer Odyssey; White, [15]) and Dartmouth College (D'Agents a.k.a. AgentTCL; Gray, [14]) have been joined, and sometimes surpassed, by IBM (Aglets; Lange et al., [17]), Mitsubishi (Concordia), ObjectSpace (Voyager) and others. Research flourishes in many application areas. In spite of being identified as one of the primary application areas of agents by the Foundation for Intelligent Physical Agents (FIPA, [39]) and the Agent Society ([40]), Network Management is still relatively *terra incognita* as far as the use of mobile agents in concerned. Nevertheless, a number of centers are actively pursuing research on various aspects of mobile agents and mobile code for managing telecommunication networks. The Perpetuum Mobile Procura group at Carleton University ([41]) applies several types of mobile code in a pursuit of its ultimate goal, a plug-and-play network. Interesting research on applications of and the tradeoffs in using mobile code for Network Management has been conducted by the Computer Networks Group at Politecnico di Torino (Baldi et al., [11], [27] and [42]). The Astrolog group ([43]) at the Institut de Recherche en Informatique et Systemes Aleatoires (IRISA) uses mobile agents for the Mobile Network Manager (MNM) (Sahai, [44]). This manager can be used from any location to manage a network remotely, for example, from a laptop connected through a modem. The research at the Intelligent Mobile Agent Center of Competence (IMA-CC) of GMD Focus ([45]) concentrates on specification and development of agent platforms and applications based on state-of-the-art Intelligent Mobile Agent technology including applications in Network Management. Another group, Intelligent Communication Environments (ICE, [46]) attempts to augment the standard open networking architectures, like Telecommunications Information Networking Architecture (TINA), by incorporating technologies based on mobile code (Krause and Megadantz, [47], [48]). Active Networks (Tennenhause et al., 35]), which originated at MIT's Software Devices and Systems Group (SDS, [49]), but now spread to a number of research centers, is another interesting research activity advocating the use of programmable packets ([50]). Such packets constitute an additional soft layer on top of the hardware and fixed communication protocols, which can provide dynamic configuration,

improved security, flexible interoperability, extendable protocols, etc. This research is relevant in this context, because programs carried by communication packets can be considered mobile agents.

V Conclusions

In this paper, we discussed the use of mobile agents for managing networks. Much of the research in this area is in its infancy, so the available resources are scarce. We used our own experience to illustrate application ideas in several areas of network management. Throughout the paper, we provided a number of references to related activities at other research centers. In the preceding section, we included an overview of the research conducted at several academic and industrial institutions, which are involved in the research on network and system management with mobile agents.

Many questions still need to be answered before any definitive remarks can be made. Nevertheless, our own experience and results of the research conducted by others lend support to the claim that mobile agents present a technology that will become very useful, and perhaps even critical, in many areas of system and network management. Recent developments in the use of the Internet indicate that underline the importance of the agents in information intensive applications. Agent mobility is being increasingly used to perform various tasks that would otherwise require extensive attention spans by the users of the services provided on the Internet. Network management researchers should watch very closely the advancements in the Internet-based network computing technology, because the proliferation of this technology will be a driving force for designing better management systems. One trend, which is increasingly visible as exemplified by the technology based on Jini, is an intersection of technologies based on remote objects (like CORBA), mobile code and agents.

VI Bibliography

1. O'Hare, G. and Jennings, N., (Eds.), *Foundations of Distributed Artificial Intelligence*, John Wiley and Sons, 1996.
2. Lesser, V. R., Multiagent Systems: An Emerging Subdiscipline of AI, ACM Computing Surveys, 27(3), pp. 340-342, ACM Press, September 1995.

3. Decker, K. S., *Distributed Problem Solving: A Survey*, IEEE Transactions on Systems, Man, and Cybernetics, 17(5), pp. 729-740, September 1987.
4. Georgeff, M. and Lansky, A., *Reactive reasoning and planning*. In Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87), pp. 677-682, Seattle, WA, 1987.
5. *Mobile Agent Facility Specification*, OMG TC Document cf/xx-x-xx, June 2nd, 1997
6. Yemini, Y., *The OSI Network Management Model*, IEEE Communication Magazine, pages 20-29, May 1993.
7. Bieszczad, A. and Pagurek, B., *Towards plug-and-play networks with mobile code*, Proceedings of the International Conference for Computer Communications ICCC'97, Cannes, France, November 19-21, 1997.
8. Nwana, H. S., *Software Agents: An Overview*, Knowledge Engineering Review, Vol. 11, No 3, pp.1-40, Sept 1996.
9. Hewitt, C., *Viewing Control Structures as Patterns of Passing Messages*, Artificial Intelligence 8(3), pp. 323-364, 1977.
10. Green, S., Hurst, L., Nangle, B., Cunningham, P., Sommers, F. and Evans, R., *Software Agents: A review*, Technical Report, Department of Computer Science, Trinity College, Deblin, Ireland.
11. *Mobile Code Bibliography*, URL : http://www.cnri.reston.va.us/home/koe/bib/mobile-abs.bib.html
12. Baldi, M., Gai, S. and Picco, G. P., *Exploiting Code Mobility in Decentralized and Flexible Network Management*, First International Workshop on Mobile Agents Mobile Agents'97, Berlin, Germany, April 7-8, 1997.
13. Vinoski, S., *CORBA overview: CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments*, IEEE *Communications Magazine*, Vol. 14, No. 2, Feburary, 1997.
14. Gray., R. S., *Agent Tcl: A transportable agent system*. In Proceedings of the CIKM Workshop on Intelligent Information Agents, Baltimore, Md., December 1995.
15. Kotay, K. and Kotz, D., *Transportable Agents*. In Yannis Labrou and Tim Finin, editors, Proceedings of the CIKM Workshop on Intelligent Information Agents, Third International Conference on Information and Knowledge Management (CIKM 94), Gaithersburg, Maryland, December 1994.

16. White, J. E., *Telescript Technology, The Foundation of the Electronic Marketplace*, General Magic White Paper, 1994.
17. Lange, D. B.,Oshima, M., Karjoth, G. and Kosaka, K. *Aglets: Programming Mobile Agents in Java* In Proceedings of Worldwide Computing and Its Applications (WWCA'97), Lecture Notes in Computer Science, Vol. 1274, Springer Verlag 1997.
18. URL : http://java.sun.com/products/embeddedjava/
19. URL : http://java.sun.com/products/jini/
20. White T., Pagurek B. (1998), Towards Multi-Swarm Problem Solving in Networks. In Proceedings of the 3rd International Conference on Multi-Agent Systems (ICMAS '98), July, 1998.
21. Cheng, D. T. and Covaci, S., *The OMG Mobile Agent Facility : A Submission*, in Rothermel, K. and Popescu-Zeletin, R., Eds., *Mobile Agents*, Springer-Verlag, 1997.
22. Schramm, C., Bieszczad, A. and Pagurek, B. (1998), *Application-Oriented Network Modeling with Mobile Agents.* Proceedings of the IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana, February 1998.
23. Bieszczad, A. and Pagurek, B., (1998), *Network Management Application-Oriented Taxonomy of Mobile Code*, Proceedings of the EEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana, February 15-20, 1998.
24. Susilo, G., Bieszczad, A. and Pagurek, B. (1998), *Infrastructure for Advanced Network Management based on Mobile Code.* Proceedings of the IEEE/IFIP Network Operations and Management Symposium NOMS'98, New Orleans, Louisiana, February 15-20, 1998.
25. White T., Bieszczad A., Pagurek B., Sugar G. and Tran X., *Intelligent Network Management using Mobile Agents*, Proceedings of the 2nd Canadian Conference on Broadband Research (CCBR '98), June 21-24, 1998, Ottawa, Canada.
26. El-Darieby, M., *Intelligent Mobile Agents for Network Fault Management*, Technical Report SCE-98-13, System and Computer Engineering, Carleton University, 1998.
27. Gosling, J. and Arnold, K., Joy, B., Steele, G., Lindholm, T., Walrath, K., Campione, M., Yellin, F. et al, *The Java Series*, Addison-Wesley, 1996.

2.28. Baldi, M. and Picco, G. P., *Evaluating the Tradeoffs of Mobile Code Paradigms in Network Management Applications*, 20th International Conference on Software Engineering (ICSE '98), Kyoto, Japan, Apr. 1998.

29. White T., Pagurek B., and Oppacher, F., *Connection Management using Adaptive Agents*, Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), Las Vegas, July 13-16, 1998.
30. White T., Bieszczad A., Pagurek B., *Distributed Fault Location in Networks Using Mobile Agents*, Proceedings of the International Workshop on Agents in Telecommunications Applications IATA'98, AgentWorld'98, Paris, France, July 4th-7th, 1998.
31. Case, J. D., Fedor, M., Schoffstall, M. L. and Davin, C., *Simple Network Management Protocol*, RFC 1157, May 1990.
32. Case, J. D. and Levi, D. B., *SNMP Mid-Level-Manager MIB*, Draft, IETF, 1993.
33. Mennie, D., *Agent For Remote Maintenance System*, 4th-year Project Report, System and Computer Engineering, Carleton University, 1998.
34. Magedanz, T., Rothermel, K. and Krause, S., *Intelligent Agents: An Emerging Technology for Next Generation Telecommunications?*, IEEE INFOCOM 1996, San Francisco, USA, March 24-28, 1996
35. Tennenhouse, D. L., Smith, J. M., Sincoskie, W. D., Wetherall, D. J. and Minden , G. J., *A Survey of Active Network Research*, IEEE Communications Magazine, pages 80-86. January 1997.
36. Pagurek B., Li Y., Bieszczad A., Susilo G., *Network Configuration Management In Heterogeneous ATM Environments*, Proceedings of the International Workshop on Agents in Telecommunications Applications IATA'98, AgentWorld'98, Paris, France, July 4th-7th, 1998.
37. Raza, K. S., *Implementation of Plug-and-Play Printer with Mobile Agents*, Technical Report SCE-98-04, System and Computer Engineering, Carleton University, 1998.
38. Dorigo, M., Maniezzo, V. and Colorni, A., *The Ant System: Optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics-Part B, vol.26, No1, pp 1-13 (1996)
39. URL : http://drogo.cselt.stet.it/fipa/
40. URL : http://www.agent.org/
41. URL : http://www.sce.carleton.ca/netmanage/

42. URL : http://spabalda.polito.it/netgroup/

43. URL : http://www.irisa.fr/solidor/work/astrolog.html

44. Sahai, A., Morin, C. and Billiart, S., *Intelligent agents for a Mobile Network Manager (MNM).* In Proceedings of the IFIP/IEEE International Conference on Intelligent Networks and Intelligence in Networks (2IN'97) , Paris, France, September 1997.

45. URL : http://www.fokus.gmd.de/ima/

46. URL : http://www.fokus.gmd.de/ice/

47. Krause, S. and Megadantz, T., *Mobile Service Agents Enabling "Intelligence on Demand" in Telecommunications*, Proceddings of IEEE GLOBECOM'96, London, U.K., 18-22 Nov. 1996.

48. URL : http://www.eurescom.de/public/Projects/P700-series/P712.htm

49. URL : http://www.sds.lcs.mit.edu/

50. URL : http://playground.sun.com/pub/ipng/html/ipng-main.html

VII Author biographies:

Andrzej Bieszczad received his Master of Informatics degree from the Jagiellonian University in Cracow, Poland in 1981 and Master of Computer Science and Ph.D. in Systems and Computer Engineering from the Carleton University in Ottawa in 1993 and 1996 respectively. Dr. Bieszczad spent many years in the computer and telecommunications industry including ITT/Alcatel and Nortel, where he was involved in research and development activities. Since 1988, his work focused on exploring intelligent methods for Network Management. Between 1993 and 1998, Dr. Bieszczad was associated with the Network Management and Artificial Intelligence Lab of Carleton University where he studied and worked as a researcher, and then as an Associate Professor in the Systems and Computer Engineering department. Currently he is with the Network Management group in the Advanced Technologies Division of Bell Laboratories. His current research concentrates on applications of agents, mobile code and artificial intelligence in Network Management.

Tony White received BA and MA degrees in Theoretical Physics from Cambridge University in 1978 and 1982 respectively. He has an MCS in Computer Science from Carleton University, awarded in 1993, and is currently studying for a Ph.D. in Electrical Engineering there. Tony has published papers in the areas

of Genetic Algorithms, Object Oriented Design, Mobile Agents, Biologically-inspired agents, Expert Systems and Fault Diagnosis and has co-authored six patents in the Telecommunications domain. His research interests center on the exploitation of biological metaphors for problem solving in Telecommunications, this being the subject of his Ph.D. research.

Bernard Pagurek received his M.A.Sc. and Ph.D. degrees in Electrical Engineering from the University of Toronto in 1962 and 1965 respectively. After a sojourn as a postdoctoral fellow at Imperial College, London, he came to Carleton University, Ottawa, where he taught in the Electrical Engineering program and conducted research in automatic control theory, optimization, and queueing theory. His current research interests include artificial intelligence and its application to diagnosis and network fault management. For network management, he is involved in the development of a cooperating mobile agent architecture. He is presently a professor at Carleton University where his research is funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada and the Communication and Information Technology of Ontario (CITO).

Table 1

Possible Benefit	Justification
Efficiency savings	CPU consumption is limited, because a mobile agent executes only on one node at a time. Other nodes do not run an agent until needed.
Space savings	Resource consumption is limited, because a mobile agent resides only on one node at a time. In contrast, static multiple servers require duplication of functionality at every location. Mobile agents carry the functionality with them, so it does not have to be duplicated. Remote objects provide similar benefits, but the costs of the middleware might be high.
Reduction in network traffic	Code is very often smaller than data that it processes, so the transfer of mobile agents to the sources of data creates less traffic than transferring the data. Remote objects can help in some cases, but they also involve marshalling of parameters, which may be large.
Asynchronous autonomous interaction	Mobile agents can be delegated to perform certain tasks even if the delegating entity does not remain active.
Interaction with real-time systems	Installing a mobile agent close to a real-time system may prevent delays caused by network congestion. In Network Management systems NM agents usually reside close to the hardware, so this advantage might not be as clear as others.
Robustness and fault tolerance	If a distributed system starts to malfunction, then mobile agents can be used to increase availability of certain services in the concerned areas. For example, the density of fault detecting or repairing agents can be increased. Some kind of meta-level management of agents is required to ensure that the agent-based system fulfills its purpose.
Support for heterogeneous environments	Mobile agents are separated from the hosts by the mobility framework. If the framework is in place, agents can target any system. The costs of running a Java Virtual Machine (JVM) on a device are decreasing. Java chips will probably dominate in the future, but the underlying technology is also evolving in the direction of ever-smaller footprints (e.g. Jini [19]).

On-line extensibility of services	Mobile agents can be used to extend capabilities of applications, for example, providing services. This allows for building systems that are extremely flexible.
Convenient development paradigm	Creating distributed systems based on mobile agents is relatively easy. The difficult part is the mobility framework, but when it is in place, then creating applications is facilitated. High-level, rapid application development (RAD) environments for agents will be needed when the field matures. It is quite probable that the flourishing tools for object-oriented programming will evolve into agent-oriented development environments, which will include some functionality to facilitate agent mobility.
Easy software upgrades	A mobile agent can be exchanged virtually at will. In contrast, swapping functionality of servers is complicated; especially, if we want to maintain the appropriate level of quality of service (QoS).

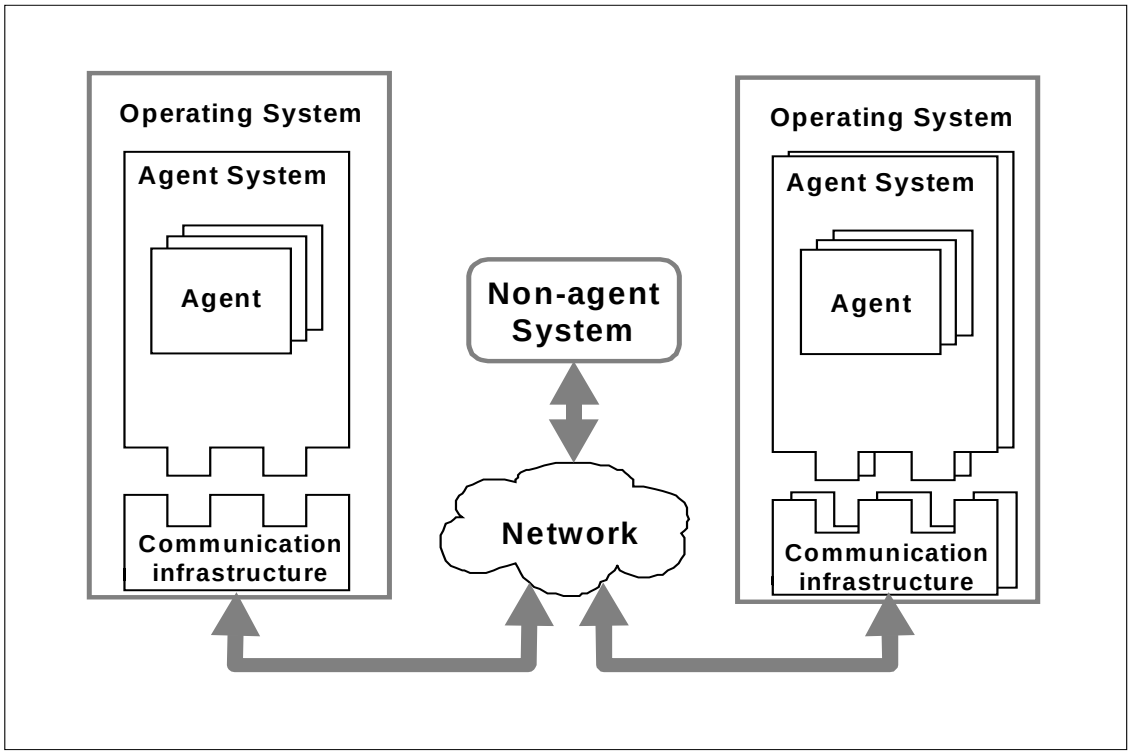

Figure 1 Mobile Agent Facility Architecture

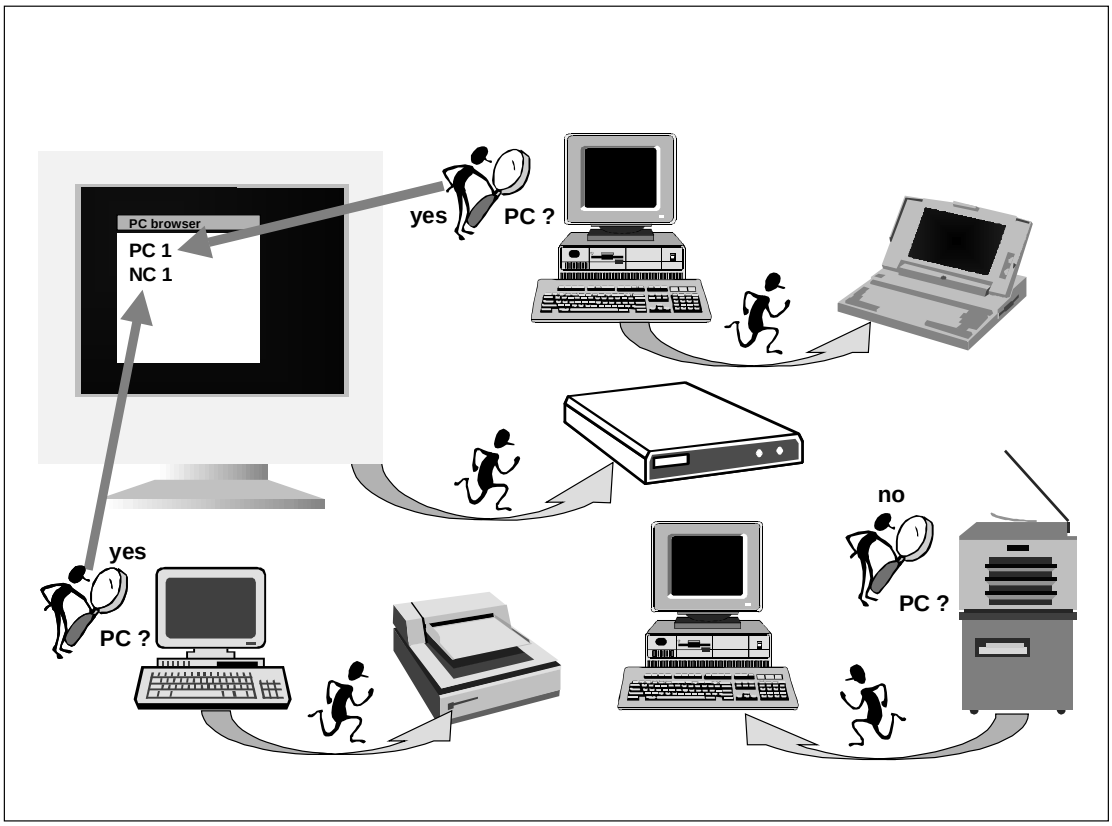

Figure 2 Discovery agents

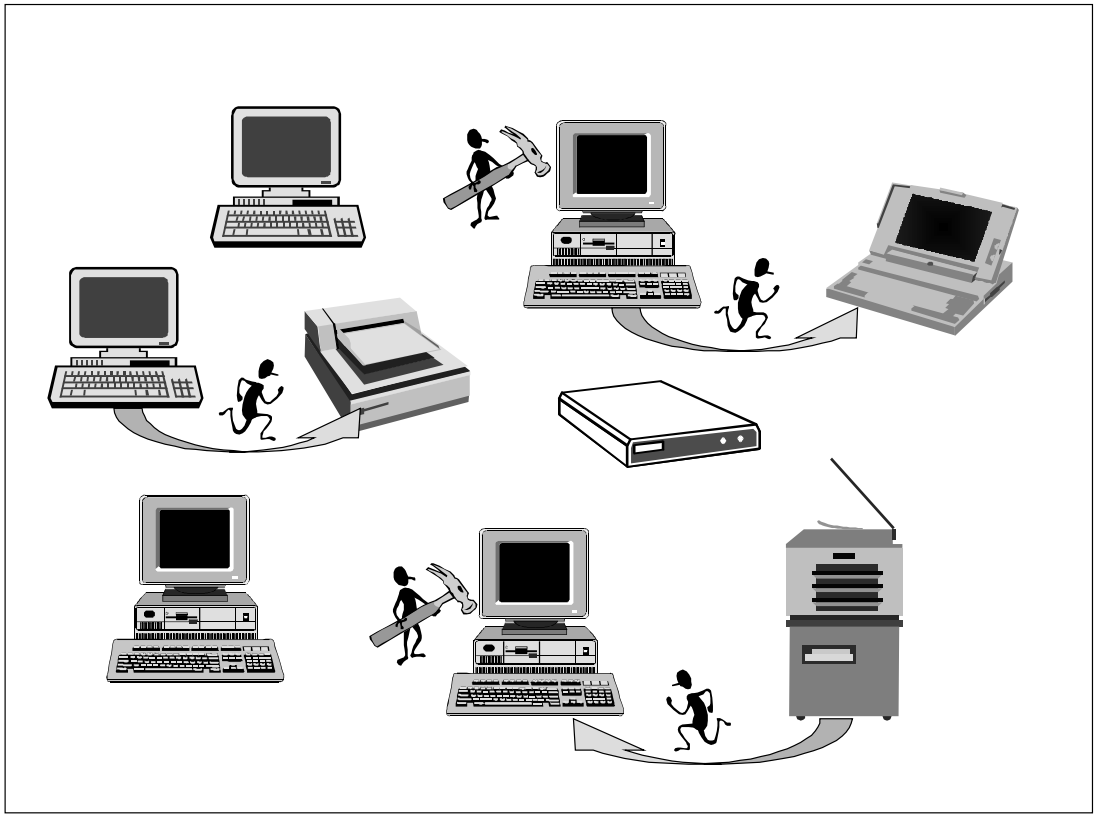

Figure 3. Maintenance agents

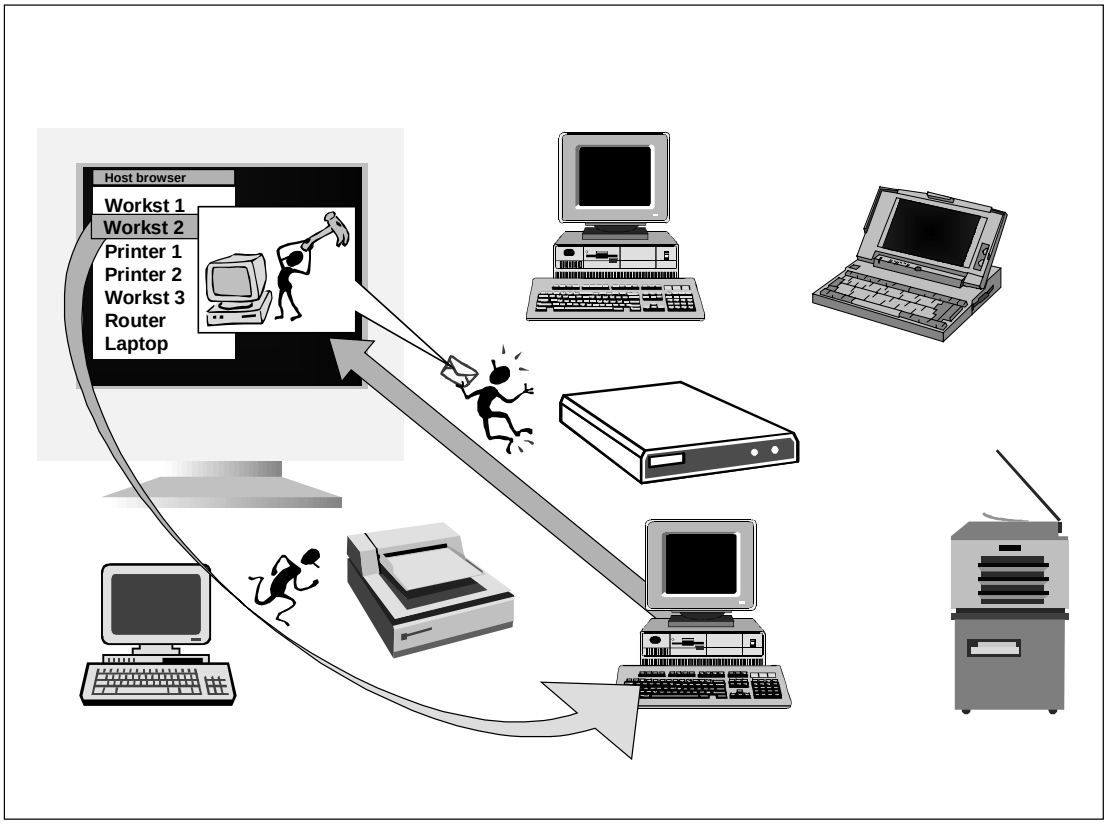

Figure 4. Browsing applets

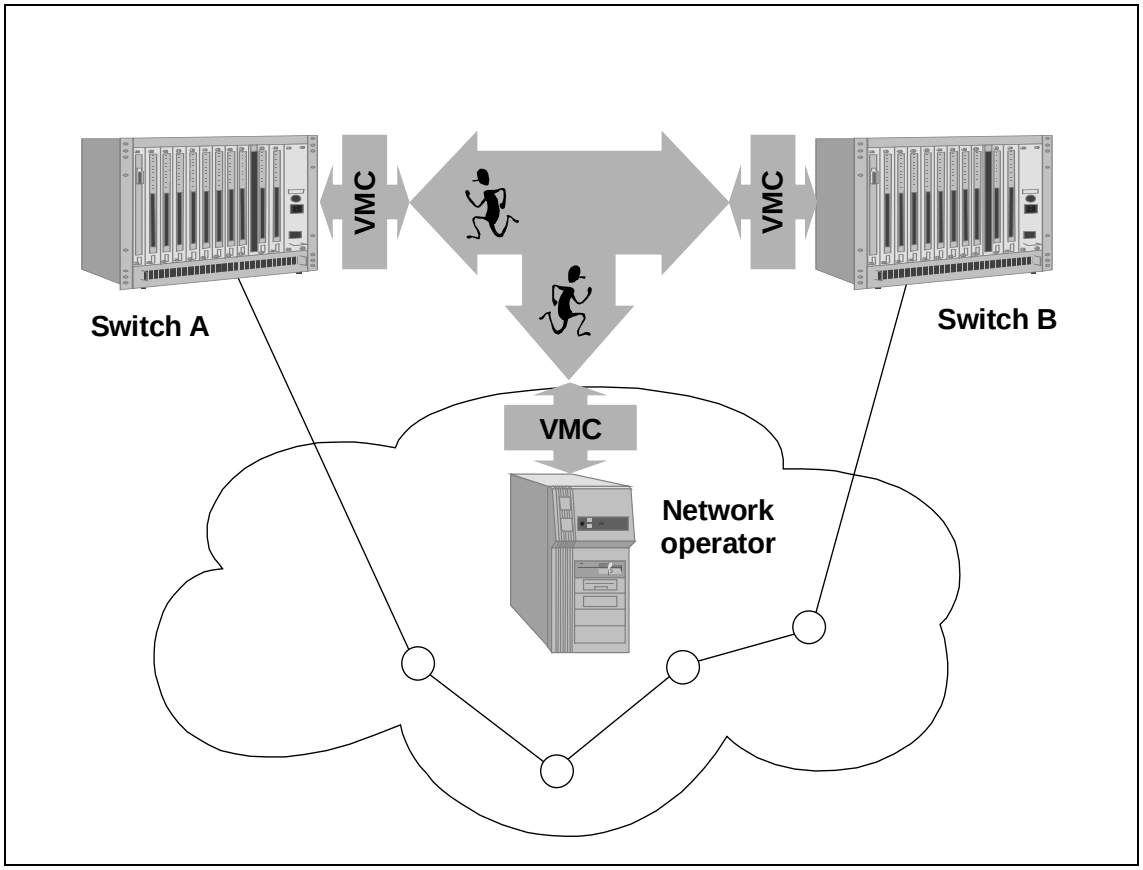

Figure 5. Service provisioning agents

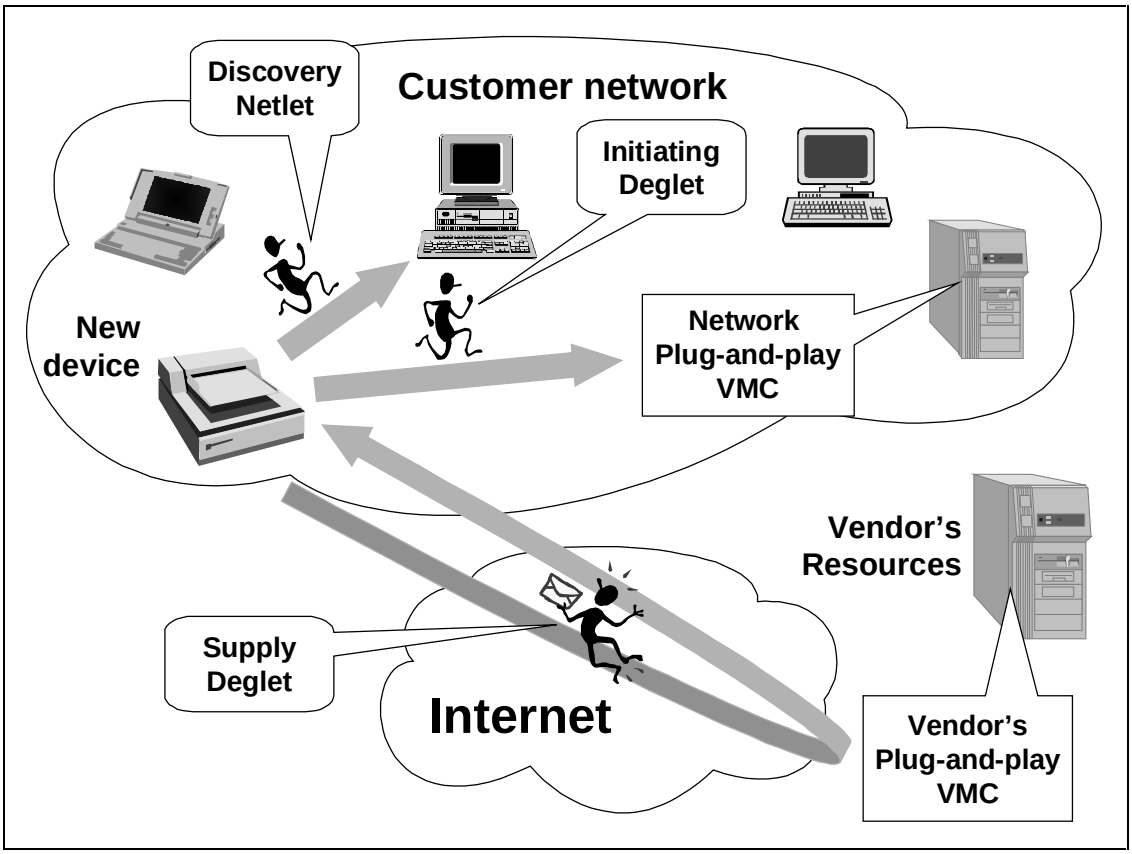

Figure 6. Plug and play agents