Management Technologies for a Pervasive Computing System

University of Waterloo
ECE750-5: Distributed and Network-Centric Computing

Michael Jarrett (99 318 764)
msjarret@uwaterloo.ca

July 28, 2003

Abstract

Pervasive computing describes the concept of massively distributed computing systems embedded in our environment. These systems would consist of numerous computing devices, many of them designed for interaction with people in such a form that they may not even be recognized as a computer.

One difficulty in building pervasive computing systems is managing their environments. Management in modern computer systems tend to be done by hand by experts. This approach would be expensive for pervasive systems as simple as thousands of homogenous deveices. When one considers the management of a system where computers are embedded in 'smart dust' or even in the paint in your wall, it becomes apparent that new techniques are needed to manage these networks.

Inspiration for solutions to this problem can often be borrowed from the similar fields of autonomic computing and grid computing. Autonomic computing is the concept of creating self-managing systems as a solution to the problem of managing increasingly large computing environments. Grid computing involves creating large distributed systems that act like a single powerful computing resource, and shares many of the management problems of pervasive computing. Ideas out of both fields can be used in the search for a solution to pervasive computing management.

Much of the focus on management of pervasive systems concentrates individual aspects of management. Work has been done to scale up systems for configuring networks of devices so that they work for large numbers of devices in a changing environment. Management of mobile devices adds extra challenges such that these devices can be contacted regardless of the network they associate with. Security needs to be managed, from the perspectives of controlling access to resources, and authenticating and managing users. For these systems to be used, privacy must be managed in a way that the user can trust the system they are using.

Some work takes a broader approach. The notion of defining policy in such a way that it can be easily distributed and implemented by a system is one such approach. Many others prefer to look at these systems as sets of mobile agents and attempt to manage these collectively.

Both of these broad approaches are useful in attempting to reason about a generalized management structure for a pervasive computing environment. Present in many of the discussed management techniques is the idea of policy-based management. Central to this is the ability to define policy, and techniques to implement it. The former has been researched extensively and many successes have resulted. The latter, the implementation of policy, is an area that has received significantly less attention.

It is proposed that research be done into agent-based systems that implement distributed policies in a unifying system for the overall problem of pervasive computing management.

Table of Contents

1 Introduction

1.1 Pervasive Computing

Gordon Moore made on observation in 1965, later to become the famous 'Moore's Law', that the transistor count in silicon devicecs would double approximately every eighteen months. This tends to reflect our general computing capacity, where both our computing storage and computing power both follow the same trend.

While Moore's Law has allowed for devices to get more powerful, it also allows devices with the same capabilities to get smaller; a handheld computer today has the same computing power as desktop computers of ten years ago. With the progression of networking and wireless technologies, these devices have ever increasingly been connected together.

Pervasive computing is viewed by many as the next step in this evolution. The pervasive computing environment would have computers all around us, but in many different forms; some of them mobile, and all of them connected. All of these computers will gracefully integrate with our environment, such that the devices would not even seem like a computer in the traditional sense - "these devices are going to liberate us from that box on our desks." [1]

Extended far enough, many have envisioned a 'sea of devices' on a scale magnitudes larger than today, and others even of 'smart dust', where even the paint on the wall might contain some form of computer.

Many of the key players in the computer industry are interested in pervasive computing.

IBM Research [2] has dedicated significant research efforts towards pervasive computing. IBM's product teams have concentrated on software to integrate existing mobile devices such as smart mobile phones and personal digital assistants. Most of this is encompassed under their 'WebSphere' product platform, offering among other things voice, connection and smart card management services. IBM Research has also run recent conferences on pervasive computing.

Sun Microsystems Labs While not often talking directly about pervasive computing, Sun Microsystems works on a number of projects that relate in some way to it. Project JXTA [3] looks to develop a set of open technologies enabling peer-to-peer communication, and this project often focuses on mobile devices. Recent work on JXTA has focused on creating implementations compact enough to run on devices, to enable a peer-to-peer pervasive 'web of things'. The Jini network technology [4] can be used to associate identities and resources and run distributed systems over networks.

National Institute of Standards and Technology (NIST) [5] has been running a pervasive computing conference since January 2000. They have also released several documents regarding software tools, and are supporting the development of standards for network and communcation technologies for pervasive computing environments.

IEEE Computer Society releases a quarterly publication, co-published with the IEEE Communication Society, called *IEEE Pervasive Computing* [6], servicing as a hub for peer-reviewed pervasive computing research. Also, *IEEE Distributed Systems Online* [7] provides a free web-based resource which often features articles on pervasive computing.

1.2 Managing Pervasive Computing Environments

Today, most devices and computer infrastructures are managed manually. In a networked environment, IP ranges are assigned for allocation to servers by hand, network topologies and routing are manually tuned, and service locations (for example DNS) are hard-coded. Security-wise, certificates are hard-coded into browsers, and people manually log into a myriad of different services with different account information for each. Software-wise, software patches and updates are normally distributed in an ad-hoc fashion, or more often than not, the user is told to install it themselves.

This approach is already very expensive to manage at the current scale. With pervasive computing, there is a wide variety of challenges that further this process to the point where it is infeasible to manually manage devices.

- The number of devices is orders of magnitude larger.
- The devices (except in tightly controlled situations) are heterogeneous.
- Devices may have wildly varying storage, computational, and bandwidth capacities.
- Users that interact with devices may not be part of the local authentication domain, or may even be malicious. At very least, the users may have goals that do not match that of the owner of the infrastructure they use.
- Devices may be mobile and move across networks or out-of-range of any network.

> Another thorny issue is how to manage so many different computers. It is difficult enough for a company to coordinate a network of desktop computers that stay put. So besides thinking about how data is rendered, the pervasive computing organization is also working on means of managing the groups of users, the software on their devices and the security of those devices. [1]

Methods and technologies are clearly required to allow for the management of devices in an environment that can meet the following requirements.

- Scales up to a large number of devices.
- Gracecfully degrades to run efficiently on compact devices with limited need for configuration or for devices lacking the capacity to deal with processor or bandwidth intensive work.
- Establishes appropriate levels of trusts with known or unknown devices in the environment.
- Able to authenticate users from a variety of domains.
- Can deal with mobile or transient devices.
- Modular and extendable to allow for application-specific configuration attributes.

2 Related Topics

2.1 Autonomic Computing

Autonomic computing started with a document[8] published by IBM in October 2001. The author started by describing a growing problem in information technology of managing complexity, with the conclusion that if something was not done, that systems would not be able to improve due to the sheer cost of managing them.

The proposed solution to this was *autonomic computing*. The analogy to the autonomic nervous system is intentional; like the biological equivalent, an autonomic computer system would manage its day to day operations on its own without requring effort from administrators. These administrators could then concentrate on setting the high-level policies that the system must follow, and rely on the autonomic system to handle the gruntwork.

IBM's proposal listed eight characteristics of an autonomic system, which has since been summarized to four key attributes that an autonomic computing system possesses.

- **Self-configuring:** The system will be able to configure itself appropriately for a given set of requirements, and reconfigure itself in response to a change in conditions.
- **Self-optimizing:** The system pro-actively seeks to optimize its performance and use of resources, and react to changes in demand for particular resoruces.
- **Self-healing:** The system can handle hardware and software failures while continuing to function correctly.
- **Self-protecting:** The system will pro-actively attempt to secure itself from attack, and also detect and respond to intrusions.

This concept instantly caught on with many of the large corporate players in the information technology world, and now IBM, HP and Microsoft, among others, are desperately vying to beat each other to market with new autonomic computing advances for their management platforms. Researchers have tended to focus on individual technologies to provide the tools to build autonomic systems, but little attention has been paid on the overall structure of such a system.

Autonomic computing and the management of a pervasive computing environment share many of the same challenges.

- Handling distributed networks of computing elements.
- Managing heterogeneous devices.
- Allowing for the management of a set of devices that is far too large for the set of administrators to configure manually.

- Responding to dynamically changing environments.

It is conceivable that technologies applied to autonomic computing would be equally useful for pervasive computing environments. Furthermore, a complete autonomic computing infrastructure, suitably tuned for the unique challenges of the pervasive computing environment would certainly solve the problem of managing pervasive computing environments.

2.2 Grid Computing

Frequently associated with either autonomic computing or pervasive computing is the idea of grid computing. Grid computing refers to techologies enabling the creation of large distributed systems of (normally geographically-distributed) computers to create the apprearance of a single large computing resource.

The name "grid computing" refers to a vision of the future where a giant distributed computing system would be available, where one could 'plug in' to the grid computing network and have access to a large supply of computing resources, much like how one draws power from the power grid.

Much of the efforts on grid computing have been towards creating large distributed research networks on which to do scientific computation. Also a part of this is the ability to share large sets of results across groups of peers around the world.

Grid computing has the advantage in that it has been well researched and unlike autonomic computing, is currently experiencing a much stronger push towards open standards. The Globus Project is working on the Open Grid Services Architecture (OGSA), which has been rallied behind by several groups in the hopes of defining [9] standards by which grids can operate. Both IBM and HP have also rallied behind the Global Grid Forum, which is another open group dedicated to furthering grid technology research. Even autonomic computing projects often quote these groups as their hopes for standards, due to the similar challenges and lack of standardization efforts in their own field.

Several companies have already released their own grid computing architectures for their platforms. Sun has released the Sun Grid Engine [10] set of products, and Sun Microsystems Labs has several research efforts into distributed systems technologies that would apply to grid computing. The IBM OptimalGrid[11] system acts as a middleware layer between a computation and a grid, where the grid technology itself is the one defined by the OGSA. It applies autonomic computing technologies to distribute and manage jobs in a grid computing environment.

3 Literature Survey

Pervasive computing management as a general topic gets paid very little attention as opposed to some of its more interesting subtopics, like trust or mobility management. Most research has focused on a narrow subsection of management, choosing to discuss data management or management of network topologies, and only rarely even a cursory investigation of the similarities between different management forms.

Some of the general approaches to management have studied agent based approaches or policy-based approaches. Some of the areas that tend to be focused on for individual attention include network management and security management (especially trust management).

3.1 Agents

A term from the field of artificial intelligence (AI), and often borrowed by the distributed computing field is that of agents. An agent in AI terminology 'thinks' and 'acts', and much of the field has focused on inventing novel methods to do this.

Several groups have focused on using the concept of agents to manage systems. These agents are normally 'software agents' that run at a central location and use a common interface to interact with devices. By far the most common is the Simple Network Management Protocol (SNMP), which has become a common management interface for management of hardware devices like routers. Recently, work has been done on other, similar technologies, such as the Common Information Model (CIM) and Java Management Extensions (JMX), each with specific advantages over the other for certain situations.

A paper[12] published on the development of the MoDPAI describes an attempt to create software agents to manage a network of devices, supposedly for a pervasive environment. The agent operated based on a knowledge base of behaviour to define its functionality. Using SMNP to monitor the devices, it would analyse the data, and take action as required to keep the network running. Furthermore, it communicates with a human administrator when it makes changes, essentially acting as an intermediary between it and the system.

Unfortunately, MoDPAI, while useful in management of traditional networks, would have trouble with truly pervasive networks, since devices need to be entered into the system manually. Also the rules are defined beforehand in a knowledge base at a single central point of control, with no means to distribute the management responsibility.

Others have approached the problem from the opposite direction, instead terming the devices (or at least some entity representing one or more devices) in a pervasive system as agents, and then applying techniques that are used to develop multi-agent systems to coordinate these efforts.

Turner and Jennings' paper[13] on the scalability of multi-agent systems fits perfectly with the idea of pervasive computing. Their model, based on a scenario of customer agents trading with supplier agents, assumes that each agent is working to meet its own goals, and measures the effects of various topologies on allowing them to meet these goals. Finally, they devise a method by which a subset of agents can decide to change their topology when each device determines their utility to do so is sufficient. This works particularly well with numerous small devices with various computational and communcation capacities, since each device can determine the utility in reconfiguring to use more or less of these resources.

While not discussed in detail, the paper also presents the ideas of the agent topologies being self-building, and the idea of agents creating intermediaries as required to perform tasks as potential techniques to create scalable multi-agent systems.

3.2 Policy-Based Management

A common approach to management is one borrowed from software development techniques - namely separation of policy from mechanism. With a sufficiently expressive policy language, one could define the specifications of the system, and then rely on a generic mechanism to operationalize these policies as appropriate.

This leads to the field of policy-based management. Two key developments that need to be made in this field are the creation of languages and tools for the definition of these policies, and system management tools to implement them.

Discussed in a paper[14] by Sloman of Imperial College are the details of work to implement management in terms of 'obligation policies' and 'authorization policies'. Management is divided up into management domains controlled by management objects, which are themselves a domain and a target of management (a 'meta-management' of the policies regarding management policies). These may, in turn, form subdomains, which may inherit parts of the parent domain's policy. Policies can also be defined to allow for the delegate management responsibilies to other objects, and the ability and responsibility to delegate these also controlled. Transforms must also exist to transform these policies into forms suitable for interpretation and enforcement. Also defined is the concept of an adaptor object, which would serve as the intermediary between a human operator and the system. Implementations by the *SysMan* and *IDSM* projects have taken different approaches to the methods used to implement these policies.

The Department of Computing at Imeperial College has followed up with number of papers regarding their creation of the *Ponder*[15] policy specification language. While mostly used in definition of security policies (which is certainly important in pervasive computing, though certainly not the only issue), it has inherited the ideas of specifying management functions in terms of obligations and authorizations, adding various subtypes of these policy types. The language has been published, allowing the possibility of standardization and study.

3.3 Network Management

In a traditional network infrastructures, management is done by hand down to a level above individual devices, and in some cases even the devices themselves are manually configured. Even in traditional computing environments, this quickly becomes an expensive undertaking. In a pervasive computing environment, the number of devices are far too numerous to configure by hand, and furthermore, no one authority may have sufficient control to perform these configurations. They must also be done on a timescale applicable to a transient device only existing in one environment for a short period of time.

One of the most popular techniques today for configuring network devices, the Dynamic Host Configuration Protocol (DHCP) is used for many forms of broadcast network to allow clients to automatically obtain an IP address, as well as allowing certain fixed information like DNS naming servers to be autoconfigured. However, a DHCP server itself has to have its information manually configured by an administrator, who must manually assign ranges to each DHCP server and ensure these addresses route correctly through the network to these particular devices.

One paper[16] discusses the need to quickly and efficiently configure networks. "Even if the nodes were static, manually configuring botentially billions of devices would be too time-consuming and error-prone."[16]. The authors have designed a protocol called the *Dynamic Configuration Distribution Protocol* (DCDP) which can effectively distribute a range of addresses over a set of devices, and allow these devices to acquire subranges to allocate to their own sets of devices. Their experiments demonstrate the successful configuration of several laptop computers in a wireless environment with no other information than a single range of addresses offered to one computer. Their calculations predict the system has the ability to configure 7280 device from scratch in approximately six seconds!

These sorts of approaches, while simplistic and often appropriate only to network management, show an important trend - that for network managemnet to be effective, it must be almost entirely automatic. The MoDPAI system described in section 3.1 showed how an administrator could be included for communication of events that were determined to be important enough to merit interrupting the administrator.

3.4 Trust Management

Security is a strong focus of pervasive computing. Almost completely absent is the concept of traditional intrusion detection and managing software security vulnerabilities. Rather, the focus is on trust: namely whether an environment can trust a device, and also the important issue of whether a device can trust the environment. This often involves the problem of authorization in a distributed environment - how does one communicate (and prove) one's identity in a pervasive computing environment, and how does a local environment interpret an identity from a remote domain?

Often opposing efforts to improve trust is concerns of privacy. The dilemma is that by proving one's identity, one has assumed that an individual wishes to be identified. A pervasive computing environment gives those with the knowledge to manipulate it an unparalled ability to track and monitor the many devices it contains and the users that make use of it.

One of the most popular techniques for trust management is policy specification. Much of the work described in section 3.2 was derived from work on access control policies, and still concentrates on authentication and permission.

The *PolicyMaker* system, described in a paper[17] on distributed trust management, allows the specification of trust policies that are associated only with keys, instead of with identities. The PolicyMaker system acts as a separate module on the local system, which can be queried for authorization decisions for certain actions. These actions are application-specific, and not interpreted in any way by PolicyMaker itself.

An idea used by PolicyMaker, and one that incidentally is critical to a pervasive environment is the idea that each local PolicyMaker system trusts only its own local policies. These local policies in turn specify how much remote systems can be trusted and in turn authorize trust of others. This allows for devices to have as much control as the environment in trust relationships - each device can have its own policy on how much it trusts the environment to perform actions and assign trust to other actions. Conversely, an environment can distribute the right to assign trust throughout the environment and specify who is trusted for what actions. This could be extended for environments to trust remote systems for authentication. Also, due to the decoupling of keys from user identifiers, authorization can be expressed in a way that does not require any violation of privacy.

The authors of the *Vigil*[18] framework criticize PolicyMaker only in the difficulty of defining security policies for the system and rudimentary nature of its delegation control. The Vigil system, by comparison, uses role-based authentication in a distributed *SmartSpaces* environment. Here, security decisions are transferred from a device (after this device sends a certificate identifying itself) to a 'Smart Manager' which handles communication with various security components on behalf of the device. A 'Role Manager' determines what roles should be associated with what credentials, and a 'Certificate Manager' manages and authorizes these credentials themselves. The certificate manager will often generate a certificate for a device where it is needed. The 'Security Manager' makes access decisions using information from the certificate manager, role manager, smart managers, and internal information. The security manager maintains detailed delegation information so that rights can be temporarily transferred.

Despite the authors' criticisms of policy-based systems, the security manager ends up relying on policy stored in a large Prolog-based knowledge base. This knowledge base serves to define the rules by which security decisions are made, and is scanned for updates frequently

The system defines a more rigid and complex hierarchy than PolicyMaker, and tends to gloss over ideas of distributed trust management. However, the simpler role-based administration may be

more effective, and a slightly more rigid hierarchy necessary to meet a complex set of security needs.

3.5 Data Management

Traditionally, systems have been built with the goals of centralizing data as much as possible. A whole industry has formed around centralized data storage and manipulation on the scale of terabytes or larger. This approach is insufficient for pervasive networks. In a pervasive environment, data may come from a central database, but be transmitted to thousands of devices over a large area. Conversely, a central point may wish to perform analysis on data that is being collected by a smart dust cloud of 10,000 sensors.

With the potential for such diverse structures of data sources and data sinks, methods are needed to manage this data, determining where it should be stored, how it should be cached and spread, and controlling who can access to it despite it being spread and cached across the network. Furthermore, in environments where devices may be mobile, data may only be applicable in a certain context, which needs to be represented and taken into account by management techniques.

Studies with the *Rome*[19] system from Stanford Univeristy suggests that data in pervasive systems is almost never needed at the same place it originates. Rome deals with this by defining 'triggers', representing the spatial and temporal locations in which data is relevant, and then uses this data to distribute subsets of these triggers to devices that could act upon these.

This system is designed primarily around a home environment, with the assumption of a strong Internet-based communications structure backing it. While centralized, some distribution of data is done, which is useful for coordination. However, this cerainly could not be used as a solution to data management in all cases.

SCaLaDE[20] is a middleware designed to handle data management for mobile situations, but could easily be applied to the general case of data management. Central to the design is the concept that application developers should not need to be aware of how mobile devices gain access to data, and as such, this is all handled by a middleware layer. A 'shadow proxy' is created for each device requiring information. This shadow proxy is implemented as a mobile agent using existing agent technologies. This agent 'follows' a mobile device or user in the infrastructure, and reacts to changes that may suggest a better way to access the data used by the device. Under these agents, the 'binder' serves to manipulate and change data bindings and locations as appropriate.

To separate the concerns of data management from the application, policies are defined to control data management for specific agents and items. This is implemented with the Ponder policy specification language, as described in section 3.2.

4 Propsal

Central to many of the ideas proposed for solutions to specific areas of pervasive computing management is the idea of separating policy from mechanism. Many projects in the area of data and trust management rely on policy definition languages to have management tools automatically make decisions where required to configure and manage pervasive systems.

The appearance of policy in so many areas of pervasive computing management implies that a policy-based approach may be effective in the general case for management of the system as a whole. By applying a uniform policy language and a single implementation system, entire systems could be managed by specifying an all-encompassing policy. These systems could still communicate with administrators (including device owners), by defining these requirements in the policy, and the policy can itself be used to filter what reaches the administrator, and to automatically handle most common occurances in the system.

Two key areas require research to accomplish this. One area is the ability to specify policy in a way that is parseable by computers, and to develop tools capable of defining these policies in a way that can be understood by an administrator. Techniques for dividing up these policies are needed such that devices can be given only relevent pieces required for that devices' functions. Conflict resolution techniques need to be developed when policies from multiple sources exist.

This area has already received a large amount of research. Of particular interest is the work on the Ponder policy specification language. At first glance, this language seems geared towards security policies, however its appearance in the SCaLaDE mobile data management system demonstrates that this is in fact not the case.

The more pressing issue is often that of policy implementation. Mechanisms are needed to enforce policy in a general, extensible manner across a pervasive system. These mechanisms have to deal with the many challenges of pervasive systems, such as being able to be used on constrained devices, working over large sets of such devices.

4.1 Possible Approach

Work with agent-based systems seems particularly appealing, especially where it is assumed that these agents cooperate only where it aids them in satisfying their own goals. This works particularly well with the idea of mobile devices where the owner of the device may be different from the owner of the immediate infrastructure. In this case, agents representing both owners can determine what interactions are acceptable to them, and determine whether co-operating serves their combined purposes. The techniques used by PolicyMaker for security policy is quite interesting in that each device defines its own trust policies based on the requirements of its owner. These sorts of systems need to be extended past the realm of security, and be designed to handle other issues, such as data management policy, and network configuration policy.

Therefore, it is proposed that research be performed towards the study of implementing agents that can implement a generalized pervasive network policy. It is assumed that the policy language can be derived easily from the combination of existing policy languages and the requirements discovered during the development of these agents. The most effort would be directed towards creating the architecture in which these agents would operate, and defining their behaviour and means of communication.

This could be broken down into several specific areas that would need to be addressed.

- The method by which agents communicate to each other their policies, and determine how to best act on a communicated policy from a remote agent.
- Resources available to enforce these policies. For a mobile device, this may be as simple as adjusting the internal configuration of the device's software, or responding to requests from the environment. For agents in the environment, action may need to be taken to enforce policy, such as the copying of data, or adjusting routing tables to allow devices to communicate.
- The means by which agents cooperate and form structures, especially ones that manage the infrastructure of a pervasive environment. Often these structures will be so large that they will necessarily be self-building. They will also need to optimize themselves to meet the goals of the structures' participants.
- Creating delegate agents for specific tasks, or assigning an agent to act on behalf of devices too limited or specialized to run agent software themselves.
- Extensibility must be included, to allow for specifying policies for aspects of system management that may not be apparent when the agents were originally designed. Defining methods to extend both policy definition and the means by which the agents may act to implement them. Conversely, how one handles agents which are representing devices that are less capable of implementing certain policies and methods to gracefully degrade in these cases. Note, we are making the basic assumption that agents always have the ability to understand requests, even if they do not act upon them.

While it may be tempting to focus on just one of these areas as a research topic, more important in the creation of these systems is looking at the problem as a whole; a system needs to be designed to encompass all of these attributes. Each element, taken in isolation, while useful, cannot hope to solve the problem of pervasive computing management.

5 Conclusion

Pervasive computing allows for a new era where computers are everywhere in the environment. These computers may be interacted with in such a way that people may not even perceive them as computers, or even realize they are interacting with them. Computing devices become numerous and varied on a scale unimaginable by today's computing ideals. However, this initiative cannot possibly succeed without the ability to manage these pervasive computing environments. This applies both to the owners of the devices in the environment, and the owners of the infrastructure in which they reside.

Techniques have been developed for managing important aspects of pervasive computing. Network management deals with the problem of configuring and reconfiguring a network to deal with more devices than any administrator could manage by hand, but at the same time allowing an administrator to know the state of the network and the resources in use. Security management, in particular, trust management, deals with the need to approve actions only for those authorized to perform them. Data management allows us to address the difficulties that arise in distributing data among so many devices.

Many of the descirbed techniques rely on policy implementation. This suggests that policy-based systems may in fact be useful in the general case of managing pervasive systems. While many efforts exist to define general management policy languages, not as much effort has been spent on defining general ways to implement these policies.

Some of the general techniques for pervasive system management, such as policy-based approaches and agent based approaches seem to be viable solutions to managing pervasive networks in the general case. An idea occasionally explored is the merger of policy and agent-based systems, and the generalization of such a system could be effective in managing pervasive networks.

References

[1] B. Schechter, "Seeing the light: IBM's vision of life beyond the PC," *IBM Think Research*, `http://domino.research.ibm.com/comm/wwwr_thinkresearch.nsf/pages/pervasive199.html` (current July 2003).

[2] "Pervasive Computing," *IBM Think Research*, `http://www.research.ibm.com/thinkresearch/pervasive.shtml` (current July 2003).

[3] L. Gong, "Project JXTA: A Technology Overview," Sun Microsystems, Santa Clara, CA, 2002.

[4] *Jini^{TM} Architectural Overview,* tech. white paper, Sun Microsystems, Palo Alto, CA, 1999.

[5] "Pervasive Computing Program,", *National Institute of Standards and Technology*, `http://www.itl.nist.gov/pervasivecomputing.html` (current July 2003).

[6] "IEEE Pervasive Computing," *IEEE Computer Society*, `http://www.computer.org/pervasive/` (current July 2003).

[7] "IEEE Distributed Systems Online," `http://dsonline.computer.org/` (current July 2003).

[8] *Autonomic Computing: IBM's Perspective on the State of Information Technology,* IBM Research, Westchester County, N.Y., 2001.

[9] I. Foster et al., *The Physiology of the Grid,* The Globus Project, June, 2002.

[10] "Sun ONE Grid Engine Software", *Sun Microsystems*, `http://wwws.sun.com/software/gridware/` (current July 2003).

[11] J. Kaufman et. al., "OptimalGrid – autonomic computing on the Grid", *IBM developerWorks* article, San Jose, CA, 2003.

[12] C. Bianchini et. al, "Devices Monitoring Tool using Pervasive Computing and Software Agents," *International Conference on Security and Management (SAM'2002),* Las Vegas, NV, 2002.

[13] P. Turner, and N. Jennings, "Improving the Scalability of Multi-agent Systems," *Proc. First Intl. Workshop on Infrastructure for Scalable Multi-Agent Systems,* Barcelona, Spain, 2000.

[14] M. Sloman, "Policy Driven Management for Distributed Systems," *Journal of Network and Systems Management*, vol. 2, no. 4, Plenum Press, 1994.

[15] N. Damianou et al., "The Ponder Policy Specification Language," *Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks,* Bristol, UK, 2001.

[16] A. Misra et al., "Autoconfiguration, Registration and Mobility Management for Pervasive Computing," IEEE Personal Communications Systems Magazine (Special Issue of Pervasive Computing), vol.8, August 2001 pp. 24-31, 2001.

[17] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," *Proc. IEEE Conference on Security and Privacy*, Oakland, CA, 1996.

[18] L. Kagal et al., "A Security Architecture Based on Trust Management for Pervasive Computing Systems," *Grace Hopper Celebration of Women in Computing 2002,* Vancouver, B.C., 2002.

[19] A. Huang et al., "Pervasive Computing: What Is It Good For?," *Workshop on Mobile Data Management (MobiDE)*, Seattle, WA, 1999.

[20] P. Bellavista et al., "Policy-Driven Binding to Information Resources in Mobility-Enabled Scenarios," *4^{th} Intl. Conf. on Mobile Data Management (MDM03)*, Melbourne, Australia, 2003.