Context-Aware Services Provisioning on Top of Active Technologies

IRENE SYGKOUNA, STAVROS VRONTIS, MARIA CHANTZARA, MILTIADIS ANAGNOSTOU, AND EFSTATHIOS SYKAS

National Technical University of Athens, 9, Heroon Polytechneiou Str., 15773, Zografou, Athens, Greece

{isygk, marhantz, stvront, miltos, sykas}@telecom.ntua.gr

Abstract. With respect to the pervasive computing vision, the paper deals with the efficient provisioning of context-aware services making use of Active Networks (ANs) on top of fixed and mobile infrastructure. This represents a starting point for studying on the utilization of Active Technologies in order to provide context-aware services. In these terms, Mobile Agent (MA) Technology comprises a type of Active Technology and the exploitation of its utilization in the field of context-awareness is analyzed. Following, an evaluation comparison of ANs and MAs paradigms is given, identifying the drawbacks and the advantages of each paradigm. Finally, some concluding remarks regarding the combined usage of them are presented.

1 Introduction

Weiser's [1] "ubiquitous computing", which represented the vision of people and environments augmented with computational resources that provide information and services when and where desired, has excited many researchers. Stemming from this notion, context-aware computing demonstrates promise for making our interactions with services more seamless and less distractive from our everyday activities. As stated in [2], a context-aware system can be seen as a human assistant given user's context to be responsible to make decisions in a proactive fashion, anticipating user needs while not disturbing the user, except for an emergency. The key point is how to better construct a context-aware system that could emulate such a human assistant.

On the other hand, the need for development of more complicated services that react with the network infrastructure has guided the researchers to transfer efficiency into the network nodes. Following this trend, the term "Active Technologies" has emerged, including paradigms that support computational models for the utilization of distributed computing resources inside the network. In this perspective, the IST project CONTEXT [3] is elaborating on the efficient provisioning of context-aware services making use of ANs on top of fixed and mobile infrastructure.

CONTEXT represents a starting point for studying on the utilization of other Active Technologies in order to provide context-aware services. In these terms, MAs paradigm comprises a type of Active Technology and the exploitation of its utilization in the field of context-awareness can prove to be beneficial.

This paper aims to examine the usage of Active Technologies for providing context-aware services. The explored technologies are ANs and MAs. The paper is organized as follows: Section 2 tackles the issue of context-aware service provisioning in terms of the classification and modeling of context information (section 2.1) and CONTEXT architecture for service provisioning (section 2.2). In Section 3, after identifying the benefits gained from active technologies, the implementation aspects of ANs (section 3.1) and MAs (section 3.2) are analyzed. Section 4 presents an evaluation of these active technologies. Finally, Section 5 provides some conclusive remarks and future plans regarding the presented research.

2 Context - Aware Services Provisioning

As computer and network become more pervasive, the nature of services should change accordingly. Services must become more flexible in order to respond to highly dynamic computing environments, and become more autonomous to satisfy the growing and changing requirements from users. Therefore, the need for context-aware services becomes imperative. CONTEXT will provide context-aware services based on the following procedure: specifying what context an application needs, gathering the required context and performing the appropriate actions in order to ensure transparent adaptability to the changeable environment. Following, we deal with classification and modeling of context information in order to introduce an architectural view of our context-aware services provisioning system.

2.1 Classification and Modeling of Context Information

Context can be quite rich, consisting of any attributes that would make a service function in a more proactive manner reflecting user’s needs. A classification according to the nature of context information identifies the following context categories: 1) *User Info*, e.g. name, age occupation, accounting info, calendar/agenda info, 2) *Personal Info*, e.g. health, mood, hobbies, interests, 3) *Activity Info*, e.g. “what the user is doing now”, “with whom is he now”, 4) *Social Info*, e.g. friends, family, employer, employee, colleagues, 5) *User Defined Rules*, e.g. not to be disturbed when being at place *x* or with person y, 6) *Environment Info*, e.g. location info, weather, surrounding facilities, 7) *Application Info,* e.g. application related issues, 8)*Terminal Info*, e.g. type of terminal, type of connection, installed toolkits/programs, installed peripherals, 9) *Network Info,* e.g. characteristics of network connections.

In order to cope with context information handling, a special structure is used, named Context Entity (CE) that consists of *information attributes* featuring a specific type of context information and *interfaces* indicating the means to retrieve this information. Each such interface corresponds to a capability provided by the underlying network and in this case we refer to capabilities that return acquired information. The actions that will be performed are modeled through another structure, named Action Entity (AE) that consists of *interfaces* representing the means to invoke these actions and corresponding to capabilities that apply certain configuration actions on the network infrastructure, invoke other services, or notify users.

2.2 Architecture Design of CONTEXT

CONTEXT project is currently working on the definition of a middleware that aims to offer an automatic creation, delivery and management of context-aware services. The model of context-aware services is based on policies and the Policy Core Information Model (PCIM) [4] is used. Based on the pattern that CONTEXT proposes, a service developer is enabled to construct a context-aware service using as building blocks a set of predefined CEs and AEs. The developer, through a graphical environment, just picks the available entities in order to produce a complete description of the service and based on this description, the necessary mechanisms for deploying the service are created. Following, through the subscription phase, the customer enters the attributes that customize the service and finally, through a set of responsible components, the service logic corresponding to each customer is produced (Service Logic Object - SLO) and deployed on the execution environment. The execution environment of the produced context-aware services resides within the AN platform that CONTEXT is going to utilize.

During the operation phase of a context-aware service, context information is acquired, evaluation of policies is performed and the appropriate actions to be enforced are triggered, according to the logic of the service. However, the richness of context information and the fact that context may be acquired from multiple context sources, represent the key motivation behind the introduction of a module named Context Information Access Handler (CIAH). CIAH is going to handle the gathering and dissemination of context information as well as the triggering of actions imposed by the service logic. The introduction of CIAH provides abstraction of context acquisition from applications and makes easier the development of applications. Fig.1 depicts the interactions of CIAH with the SLOs and the underlying network. As it is shown in the figure, CIAH is also deployed on the AN layer.

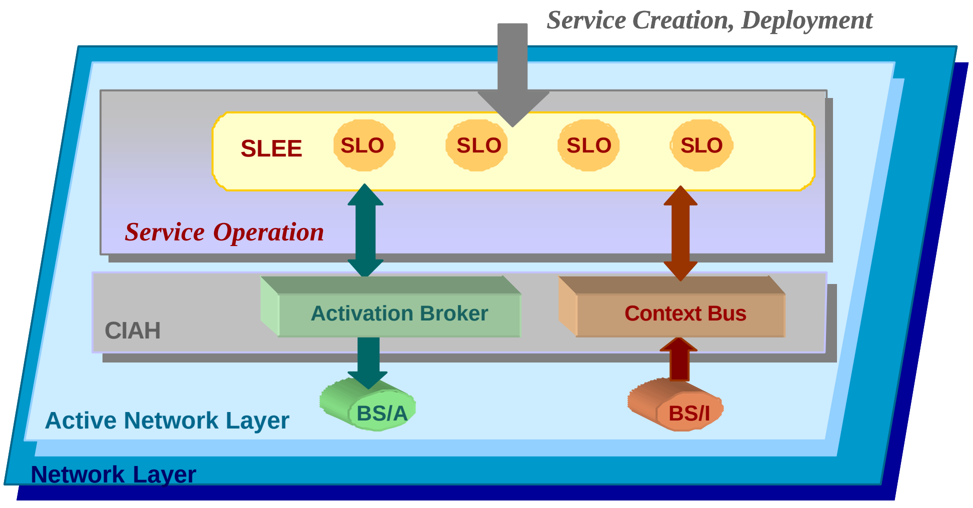

Fig. 1. CIAH Interactions

CIAH receives requests from the SLOs asking for a specific type of context information. Context Bus sub-module contacts the appropriate context information sources and disseminates the context value to the requestor SLO. Context Bus also handles SLO requests asking to be notified when a specific context value occurs. The aforementioned context retrieval mechanisms correspond to the “pull” and “push” mechanisms of receiving information. Activation Broker sub-module is responsible to trigger the actions imposed by the service logic.

In order for the CIAH to accomplish the aforementioned tasks, it interacts with the Basic Services (BSs) that are implemented on top of the AN platform and reflect the capabilities provided by the AN. BSs are offered by the active application layer and can be discriminated in those that obtain context information and are named Info BSs (BS/I), and those that are responsible to trigger actions and are named Action BSs (BS/A). BSs are distinguished from the offered context-aware services that the CONTEXT service platform is going to produce, in the sense that they are considered as auxiliary services that enable context-aware services to take advantage of the capability features offered by the AN infrastructure.

CONTEXT architecture exploits the benefits of implementing context-aware services with the use of AN technology. This architecture represents the motivation in order to study on the utilization of other Active Technologies for the provisioning of context-aware services. Following, we identify the benefits gained from Active Technologies and exploit apart from the ANs paradigm, the integration of MAs Technology in the field of context-awareness.

3 Utilization of Active Technologies

The development of context-aware services has been hampered by the need to develop a custom context infrastructure for each application. CONTEXT removes this obstacle by placing the context functionality in the network infrastructure in a form of active components. The development of middleware solutions for an efficient provisioning of context-aware services calls for distributed control and programmability inside the network. This will allow the dynamic adaptability of current and future context aware services for the benefit of the global consumer.

The need for distributed control derives from the fact that the various context sources are distributed among the network nodes. A centralized scheme for management control operations would hardly be scalable and would not allow efficient operation. On the other hand, the realization of the concept of management by delegation through the distribution of code to core network nodes will reduce the number of necessary network transactions by local processing.

Apart from distributed operation and control, programmability inside the network is another prerequisite. Building nodes that can conduct appropriate computations and thus gaining the ability to dynamically change network functionality would enable us to implement new network capabilities in a flexible way. These capabilities would be software–based allowing dynamic customization of network resources and thus, the services provided would adapt to context changes without any noticeable changes to the upper levels [5, 6].

Our approach to develop an adaptive service and context execution environment for next-generation networks, which is inherently distributed and programmable, is based on using Active Technologies on top of fixed and mobile networks. We use the term Active Technologies to include the recently developed paradigms of Active Networks (ANs), Programmable Networks (PNs), Mobile Agents (MAs) and Semantic Networks (SNs). Although these concepts were introduced by different research communities to address different problems, they have started overlapping in focus and applicability, as they are being developed further [7]. CONTEXT will make use of the Active Technology as the basic deployment, delivery and assurance mechanism that at the same time supports context information and high bandwidth requirements for future applications. The challenge such an infrastructure poses is to find the right balance among flexibility, performance, robustness and usability [6]. Following this direction, we study two different Active Technology paradigms in order to provide context-aware services: ANs and MAs.

3.1 Active Networks

ANs is a framework where network elements, primarily routers and switches, are programmable. Programs that are injected into the network are executed by the network elements to achieve higher flexibility and to present new capabilities. ANs can work in two different ways: users can preprogram selected network nodes, such as routers, with customized code before running an application, or they can program data packets, which then transport code to nodes along the way to their destination [5].

CONTEXT utilizes Active Bell-Labs Engine (ABLE) [8] system developed in Lucent Technologies to exploit ANs. ABLE is a modular and scalable software architecture that enables the deployment, control and management of active services, usually called active sessions, over network entities such as routers, WLAN access points, media gateways and servers supporting such services in IP-based networks.

The active node is composed of a Forwarding Element (FE), namely router, WLAN access point, media gateway, etc. with an inherent diverter that detects and diverts active packets to the main separate component, the Active Engine (AE), and the AE, which executes the code of the active packet (see Fig.2). The FE and the AE can be either physically separated or located at the same machine. The AE consists of the following modules: the Session Broker, which receives and parses active packets, handles and manages existing services, and distributes active packets according to requests of the services, the Info Broker which provides an efficient monitoring channel by exporting local state to active sessions and mediates all queries for local state and the Control Broker, which provides a secure channel for control and configuration operations on the active node. During the progress of CONTEXT, ABLE is expected to become extended, accommodating new modules in order to meet the forthcoming project needs. In this sense, the need for one additional module has currently been identified. This is called Context Broker and provides capabilities related to other than network-minded context, such as users, environment, application or terminal related context. The modules communicate through UDP transactions and TCP connections.

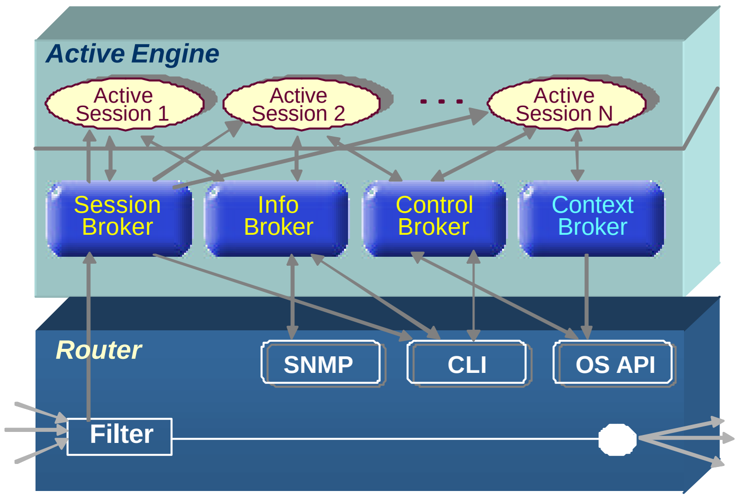

Fig. 2. ABLE Active Node Architecture

Each node can communicate with other active nodes in the following ways: the *topology-blind* addressing mode, which enables a node to send a packet to the nearest active node in a certain direction, removing the need for having available full topology information for any specific node, and the *explicit* addressing mode, which enables a node to send a packet to a specific active node [9, 10].

The testbed network infrastructure that will be used for the trial of the Context-Aware Services is illustrated in Fig.3. Specifically, it consists of two Linux-based active routers (PCs running Linux with the AE residing in the same machine with the router), two Cisco routers with adjunct active engines, and a non-active Cisco router. Apart from the IP network capabilities, Wireless LAN along with GPRS capabilities will be controlled by active nodes through the corresponding wrappers that ABLE is going to accommodate. The end-users will be able to access context-aware services using personal computers, laptops, PDAs and mobile phones.

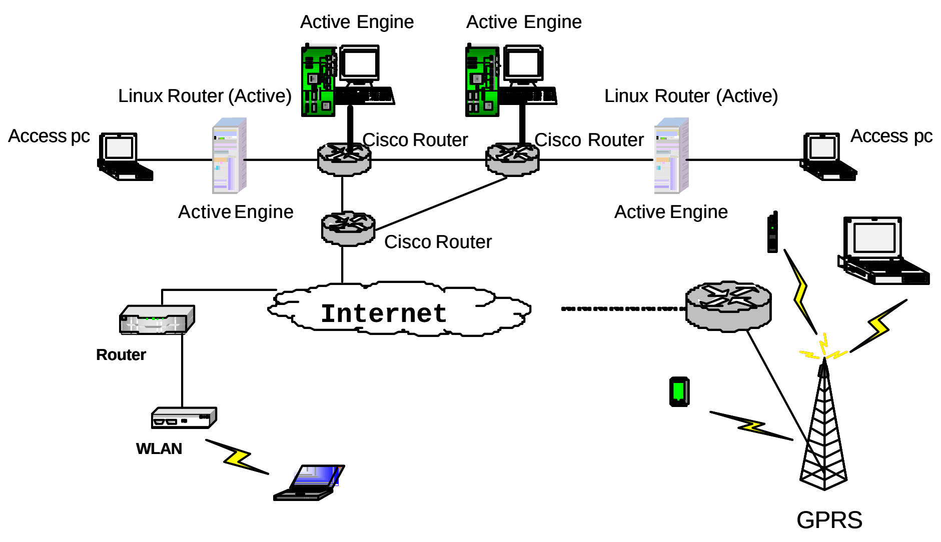

Fig. 3. Testbed Network Infrastructure

3.2 Mobile Agents

The other approach that will be exploited in the framework of the Active Technologies applicability is the MAs paradigm. The MA concept is a computational paradigm characterized by computer programs, called agents, migrating inside the network and executing on network nodes. In this sense, a MA that moves from one node to another can be interpreted as an active packet containing data and a program, which is sent from one node to another in an AN [7].

Mobile agents can move around and execute tasks autonomously and in line with their goals which can dynamically change according to the changeable nature of context information. They can also be intelligent, and this makes them even more interesting for Active Technologies. An intelligent piece of code that moves around can be considered the most advanced form of active code. All other forms derive from this one by combining some but not all characteristics mentioned within the intelligent and mobile agent research domain [11].

The architecture of the active node that will be able to host and execute the active code is depicted in Fig.4. On the bottom, there is the hardware part of the node. Then, the operating system, Linux, will export an open interface that represents the abstraction of hardware resources. The Execution Environment is provided by the MA platform, Grasshopper [12]. This is the agency as described in Mobile Agent System Interoperability Facility (MASIF) [13] standard. Each agency corresponds to a different active session and consists of places. A place is a context within an agency in which an agent is executed. The place can provide various functions such as access to local resources.

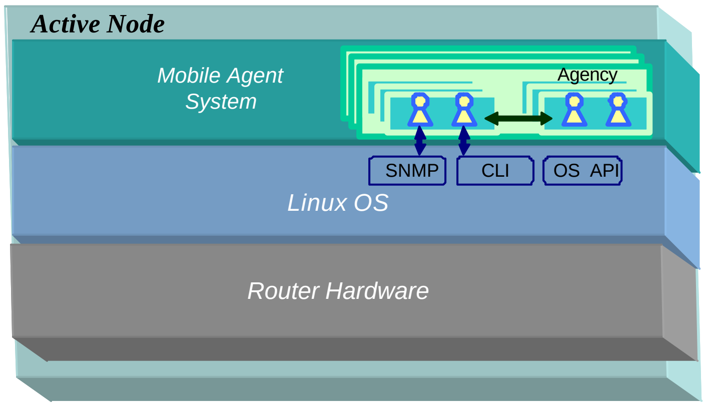

Fig. 4. Active Node for MAs

Mobile and stationary agents cooperate in an agency in order to provide programmability to the node. The *Info Stationary Agent* (SA/I) access router information through Simple Network Management Protocol (SNMP) interface [14] while the *Action Stationary Agent* (SA/A) apply configuration actions on the nodes through the Command Line Interface (CLI) or other Operating System interfaces (OS API). Consequently, SA/I along with SA/A agents provide the interfaces required for a mobile agent to access node's hardware and internal software functions. Each mobile agent implements a different BS, as has already been named a certain capability of the Active Application Layer. The mobile agents can be sent dynamically "on-demand" to routers where they are currently required. After their arrival, they interact with the appropriate stationary agents in order to fetch the required context information or apply the necessary configuration actions. For enabling mobile agents to communicate with different controlled elements, such as Linux routers, Cisco routers, WLAN access points or GPRS gateways, the stationary agents provide the corresponding wrapper functionalities.

4 Evaluation of Active Technologies

Although the motivations for proposing the MAs and ANs paradigms were quite different, coming from different research communities, these two technologies nowadays tend to share common ideas. They both overcome the traditional client/server network management model achieving better scalability through distribution and more efficient usage of resources.

However, these paradigms have some crucial differences. First of all, they are implemented at different layers of the OSI model. The MAs paradigm runs at the application layer while the ANs one at the network layer. Consequently, in the case of ANs no special treatment has been taken into account for the migration mechanism of the active code since the network elements contain sophisticated software to perform packet forwarding, and thus, the main concern is the active processing of the packets in the network infrastructure. Working at the network layer reveals the real potentiality of ANs to exploit network services more efficiently and perform fast deployment of distributed applications. On the other hand, the lowest layer of the OSI model that MAs consider programming for is the transport layer, simply making use of the communication capabilities of the environment (e.g., Java RMI). In this sense, working at the application layer, makes the MA paradigm outperform when application tasks are performed. This can be particularly interesting in the case of context-aware services since the retrieval of context information along with the actions applied do not merely imply interaction with network infrastructure but also with users, external systems or applications. This means that not only management of routers is required but also management of network servers that host applications providing context information, and in the latter case the MA paradigm is supposed to perform more efficiently [15, 16].

Secondly, as also mentioned above, the ANs paradigm supports two communication modes: the *topology-blind* addressing mode, in which an active packet is executed on every active node in a certain direction, and the *explicit* one, in which a packet is executed only on the target active node, while it is just forwarded by the intermediate ones. On the contrary, an agent can only be sent explicitly to a specific node in order to be executed. The topology-blind addressing mode of ANs offers dynamic and thorough customization of network resources even if the application does not have knowledge of the current underlying network infrastructure. However, there are applications that require the actions to be taken on specific nodes that are known a priori. In such cases, the MAs paradigm outperforms since it avoids executing the active code on every active node and eliminates the additional delay for diverting an active packet from the router to the AE, which appears in ANs.

Thirdly, the operation of ANs requires advanced router configuration in terms of special policies that enable the router to identify the active packets and treat them accordingly, namely forward them to the appropriate AE in order to be executed. In this sense, upgrading a node to an active one is more difficult in case of ANs than in MAs.

Last but not least, it is worth mentioning that while the payload of an active packet in case of ANs consists only of the program code and the data, a mobile agent transfers additionally its state while it traverses the network. On the grounds that an agent can make decisions based on this state, it migrates in the network under its own control, namely it can stop its execution in one node and continue in another with respect to the volatile nature of context information [17].

5 Conclusions and Future Work

The fact that context may be acquired from multiple distributed and heterogeneous sources as well as the fact that it changes dynamically make it a really hard work to build context-aware services. Our approach to develop a middleware for efficient provisioning of context-aware services is based on using Active Technologies on top of fixed and mobile networks. The exploitation of ANs and MAs paradigms in the field of context-awareness has been examined, highlighting the different prospects of each one.

The motivation behind ANs comes from the emerging concern on how to best adapt the existing networks in order to accommodate new type of applications such as context-aware ones. On the other hand, MAs paradigm pays less attention to the communication model that is ultimately based on high-level abstractions, exploiting it to implement distributed applications. As a consequence, a combined approach that exploits both the ANs and MAs paradigms is considered to be most fruitful based on the concept that MAs paradigm could be inspired by the ANs communication model in order to access network resources more efficiently. Alternatively, each context-aware application could use for some tasks the one paradigm and for some others the other, depending on the estimated efficiency achieved in each case. A performance evaluation both from an application and network point of view will be regarded, in terms of time and communication complexity respectively. Finally, the idea of exploiting special navigation patterns for coping with navigation and synchronization aspects of an active process will be taken into account, as it is considered essential for performance improvement.

Being inspired by both technologies, in the long term, it may be appropriate to merge these two paradigms, to form an even stronger vision of how we can exploit the potential of networks to the fullest [15].

References

1. M.Weiser, "The Computer of the 21st Century", Scientific American, Vol. 265, No. 3, pp.66-75, September 1991
2. M. Satyanarayanan, "Challenges in Implementing a Context-Aware System", editorial introduction in IEEE Pervasive Computing, pp.2, July-September 2002
3. CONTEXT: Active Creation, Delivery and Management of efficient Context Aware Services, IST-2001-38142-CONTEXT, http://context.upc.es
4. IETF Policy Core Information Model Extensions, http://www.ietf.org/internet-drafts/draft-ietf-policy-pcim-ext-08.txt, May 2002
5. Sixto Ortiz Jr., "Active Networks: The Programmable Pipeline", IEEE Computer Magazine, Vol.31, No.8, pp.19-21, August 1998
6. S. Karnouskos, "Realization of a secure active and programmable network infrastructure via mobile agent technology", Computer Communications, Vol. 25, Issue 16, pp.1465-1476, October 2002
7. R. Kawamura, R. Stadler, "Active Distributed Management for IP Networks", IEEE Communications Magazine, pp.114-120, April 2000
8. ABLE: "Active Bell Labs Engine", http://www.cs.bell-labs.com/who/ABLE
9. D. Razz, Y. Shavitt, "An Active Network Approach for Efficient Network Management", IWAN '99, July 1999, Berlin, Germany
10. D. Razz, Y. Shavitt, "Active Networks for Efficient Distributed Network Management", IEEE Communications Magazine, 38(3), pp.138-143, March 2000
11. H.S. Nwana, D.T. Ndumu, "A brief introduction to software agent technology", N. Jennings, M. Wooldridge (Eds.), Agent Technology Foundations: Applications and Markets, Springer, Berlin, 1998
12. IKV++ Technologies AG (2002), Grasshopper MAP, http://www.grasshopper.de
13. OMG MASIF Standard, http://www.fokus.gmd.de/research/cc/ecco/masif/
14. SNMP research, http://www.snmp.org
15. M. Collier, "Mobile Agents and Active Networks: Complementary or Competing Technologies?", Technical Report of the Broadband Switching & Systems Laboratory
16. Ichiro Satoh, "A Mobile Agent-based Framework for Active Networks", in Proceedings of IEEE Systems, Man, Cybernetics Conference (SMC'99), pp.71-76, IEEE, October 1999
17. H. Liu, M. Nodine, "On Applying Mobile Agents to Network Management", in Proceedings of INET '96, Montreal, June 1996