A Simple Typology of Distributed Network Management Paradigms

Jean-Philippe Martin-Flatin
Telecommunication Services Group,
TCOM Laboratory,
Swiss Federal Institute of Technology (EPFL),
1015 Lausanne, Switzerland

Email: jpmf@tcom.epfl.ch

Simon Znaty
Multimedia Networks and Services Dept,
École Nationale Supérieure des
Télécommunications de Bretagne,
35512 Cesson Sévigné, France

Email: znaty@rennes.enst-bretagne.fr

30 July 1997

Abstract

Over the past few years, network management has steadily evolved from a centralized model, where all the management processing takes place on a single network management station, to distributed models, where management is distributed over a number, potentially large, of nodes. Among distributed models, one, epitomized by the SNMPv2 and CMIP protocols, has been around for several years, whereas a flurry of new ones, based on mobile code, distributed objects or cooperative agents, have only recently emerged. This paper reviews all major network management paradigms known to date, and proposes a simple typology to classify them.

Keywords: Distributed Network Management, Mobile Code, Management by Delegation, Distributed Objects, Intelligent Agents, Cooperative Agents.

1 Introduction

Network management has thrived on centralized or weakly distributed hierarchical models for many years. Soon after the advent of open systems in the second half of the 1980's, proprietary solutions gradually gave way to two open protocols, SNMP and CMIP, in the first half of the 1990's. These protocols primarily addressed what was then perceived as the most critical feature lacking in existing network management systems: interoperability between multiple vendors. SNMP was widely adopted by the IP world to manage LANs, WANs, Intranets, etc. In parallel to this wide-scale development, CMIP, richer but more complex than SNMP, found a niche market in the telecommunications world, as the ITU-T decided to adopt the OSI model as the basis for its Telecommunications Management Network (TMN) model.

Despite the lack of competition between these two protocols, which looked set to rule their separate markets for many years, the use of both SNMP and CMIP has been questioned in the recent past, together with their common underlying models. Why is it that more and more network administrators are now demanding Distributed Network Management (DNM), when the same people were happy with centralized or weakly distributed hierarchical models a couple of years ago? What triggered this sudden and massive shift toward DNM?

The answer, in our view, is twofold. First, DNM addresses what traditional models fail to provide: scalability, flexibility and robustness. These three features, identified by Goldszmidt [7] to motivate the use of his own model, Management by Delegation (MbD), can actually justify any form of DNM. Second, progress in distributed applications technologies (CORBA, intelligent agents...) and languages (Java, TeleScript, KQML...) since CMIP and SNMP were devised, suggested new ways of organizing network management. By transfer of technologies, the network management community was suddenly overwhelmed with an avalanche of new tools coming from the artificial intelligence and the software engineering communities. Research is currently going in all directions, and it is increasingly difficult to tell in which one network management is currently heading. Will the IP and telecommunications worlds adopt new management paradigms in the future? What paradigm will eventually win?

If it is true that these questions are hard to answer, we can at least try to find some elements of the answer. In engineering, we learn that a good way of unveiling trends in an apparent chaos is through classification. But although the literature offers many examples of typologies of organizational structures in other research fields, oddly enough, fairly little has been published recently in the area of network management. Many authors present the traditional approaches [17, 9, 2], others focus on just some of the new distributed paradigms [1, 16]; but none of them considers the whole range of network management paradigms.

The objective of this paper is to fill this gap, and to compile a comprehensive typology classifying all major network management paradigms known to date, whether they have been successfully implemented already or whether they are still confined to the research community. To do so, we first define a common terminology in section 2. We then define a simple typology in section 3 and, based on it, review all network management paradigms in sections 4, 5, 6 and 7.

2 Terminology

Before we proceed with the review of network management paradigms, we must first acknowledge that the network management community has not fully converged on a common terminology yet. Most people agree that the centralized model is characterized by a single Network Management Station (NMS), concentrating all the management processing, and a collection of agents limited to the role of dumb data collectors; but there are different views on several other definitions. For example, some authors motivate the use of their new distributed model by criticizing the centralized model, but overlook hierarchical models; others simply ignore the cooperative model. To address this confusion, we therefore propose the following terminology.

First of all, why should DNM stand for *Distributed Network Management*, as we said earlier, rather than *Decentralized Network Management*, as others [1, 15] advocate? Our choice is motivated by common usage in other computer science research fields: for years, people have been referring to *Distributed Systems*, *Distributed Artificial Intelligence*, *Distributed Processing Environments*, etc. It therefore makes sense, in our view, to translate the acronym DNM into *Distributed Network Management*.

Decentralized management is to the enterprise world what distributed management is to computer science: a management paradigm based on the delegation of tasks to other entities. These entities are persons in the enterprise world, machines or programs in computer science.

Delegation is a generic word, used in both contexts to embody the process of transferring power, authority, accountability and responsibility for a specific task to another entity. In network management, delegation always goes down the network hierarchy: a manager at level (N) delegates a task, i.e., a management processing unit, to a subordinate at level (N+1); this is known as *downward delegation*. In the enterprise world, we can also have *upward delegation*; e.g., an employee delegates his tasks to his manager when he is off sick. Downward delegation and upward delegation are two kinds of *vertical delegation*, typical of hierarchical models. A *hierarchical model* is characterized by a multi-layer pyramid, comprising a *top-level manager* (at level 1), several *mid-level managers* (at levels 2, 3, etc.), and *operatives* at the lowest level. In network management, NMSs globally refer to the top-level and mid-level managers, whereas operatives are called *agents*. Orthogonally to vertical delegation, we have *horizontal delegation*, between two peers at the same level, typical of *cooperative models* used in Distributed Artificial Intelligence (DAI). DNM relies on either an underlying hierarchical model, a cooperative model, or a combination of the two: indeed, any model outside the realm of centralized models belongs to DNM.

Delegation is normally a *one-to-one relationship*, between a manager and an agent in a hierarchical management model, or between two peers in a cooperative management model. Arguably, delegation may also be considered, in rare cases, as a *one-to-many relationship*, where a task is delegated to a group of entities, collectively responsible for the completion of the task. One-to-many delegation is forbidden by most authors in enterprise management (see references in [11]). It could be envisaged in DAI. In network management, we propose to classify it as a form of cooperation, by coupling the hierarchical and cooperative models: a manager delegates a task to an agent, and this agent in turn cooperates with a group of agents to achieve this task. In the case of a *many-to-many relationship*, we are clearly in the realm of cooperation rather than delegation.

The meaning of NMS has shifted over the years from *Network Management System* to *Network Management Station*. The reason for this is clear: SNMP, when it was first released, assumed an underlying centralized model, characterized by a single network management station. The whole network management system was made of a management application running on a single workstation. Several years later, SNMPv2 adopted a hierarchical model, a la CMIP, where the network management system actually comprises multiple stations. Since we are now clearly in the days of DNM, we will translate NMS into *Network Management Station* throughout this paper.

To cope with legacy network devices, whose internal SNMP/CMIP agent does not support the capabilities described in strongly distributed models, we assume in this paper that old network devices make use of proxy agents if necessary. A *proxy agent* is a network management gateway, dedicated to a certain network device and external to it; it is located between the manager and the SNMP/CMIP agent, and is transparent

to the management application. It can for instance translate a CORBA request into SNMP/CMIP protocol primitives, and vice versa. When a proxy agent is used, the SNMP/CMIP agent embedded in the network device is called a *dumb agent*. Throughout this paper, when we refer to an *agent*, we may as well refer to the pair {dumb agent, proxy agent} or simply to the SNMP/CMIP agent. This proxy agent is sometimes referred to as *delegated agent*. This expression is ambiguous, since some people give this name to programs remotely transferred to an agent [8], so we will avoid using it.

Finally, the word *agent* has traditionally a different meaning in the DAI and network management communities. In order to avoid any confusion, we will speak of an *Intelligent Agent* (IAg) when we mean an agent in the DAI sense in this paper.

3 A simple typology of network management paradigms

With these definitions in mind, we can now present a simple, intuitive typology of network management paradigms. This classification, based on the underlying organizational model, obeys 3 principles. First, we ought to separate centralized paradigms from distributed paradigms. Second, we should try to isolate what is inherently different between traditional and new paradigms. Third, we should distinguish paradigms relying on vertical delegation from those based on horizontal delegation.

All DNM technologies, regardless of their idiosyncrasies, can be classified in two broad types: weakly and strongly distributed technologies, which implement respectively weakly and strongly distributed paradigms. *Weakly distributed paradigms* are characterized by the fact that network management processing is concentrated in a handful of NMSs, whereas the numerous agents are limited to the role of dumb data collectors (in an Intranet, we typically have 1 or 2 orders of magnitude between the number of agents and the number of NMSs). Typical examples of weakly distributed network management are the hierarchical models incarnated by CMIP and SNMPv2. *Strongly distributed paradigms*, on the other hand, decentralize management processing down to each and every agent: management tasks are no longer confined to NMSs, all agents and NMSs take part in the network management processing. Many strongly distributed technologies have been suggested in the recent past, which can be grouped into 4 types. The first 3, mobile code, distributed objects and Web-Based Enterprise Management (WBEM), are based on vertical delegation, and thus assume an underlying hierarchical paradigm. The fourth type, intelligent agents, is based on horizontal delegation, and assumes a cooperative paradigm.

This simple typology is depicted in Fig. 1, and comprises four types of network management paradigms:

- centralized paradigms
- weakly distributed hierarchical paradigms
- strongly distributed hierarchical paradigms
- cooperative paradigms

	centralized paradigms	hierarchical paradigms	cooperative paradigms
not distributed	SNMPv1, SNMPv2c		
weakly distributed		RMON, SNMPv2, CMIP, HTTP	
strongly distributed		WBEM, mobile code, distributed objects	intelligent agents

Fig. 1 : Simple typology of network management paradigms

Based on this typology, we will now review the different paradigms shown above, and present the existing technologies for each paradigm.

4 Traditional paradigms

The traditional paradigms encompass the centralized paradigms, based on SNMPv1 or SNMPv2c, and the weakly distributed hierarchical paradigms, based on SNMPv2, RMON, CMIP, or CMIP derivatives like TMN. They are all well-known and therefore not presented in this paper (see [11] for an overview and references). With these management paradigms, the semantics offered to the network administrator are entirely dependent upon the protocols used underneath: the abstraction levels presented to the network administrator have a one-to-one mapping to the protocol primitives. This is not inherent in the protocols themselves, but due to the way they are traditionally used.

Let us point out that in Fig. 1, SNMPv2c is classified as a technology supporting a centralized paradigm, unlike its predecessor, SNMPv2, which is classified as a technology supporting a hierarchical paradigm. This is because the concept of *party* was left out in SNMPv2c, which rendered the Manager-to-Manager MIB obsolete, and made hierarchical management with multiple levels of managers impossible.

5 Web-based network management paradigms

Since the World-Wide Web (WWW) is now ubiquitous, several proposals (see references in [11]) have been made to use the Web technology in network management. This resulted in very different approaches, which we collectively group under the heading Web-Based Network Management (WBNM). Some of them are based on weakly distributed hierarchical technologies, others on strongly distributed ones. WBNM therefore overlaps two of the types presented in our simple typology, and does not constitute a type per se. All Web-based paradigms require an HTTP server be present in every agent, a feature already offered by some vendors today.

Web-based weakly distributed hierarchical technologies use HTTP instead of SNMP, or in conjunction with it. The use of HTTP instead of SNMP became realistic with the

advent of HTTP 1.1, which supports long-lived TCP connections (HTTP 1.0 does not, so it is less efficient than UDP-based SNMP). Within HTTP packets, MIB data can be either encoded in a specific MIME type, or embedded in an HTML structured document. Device-specific command lines can also be encoded the same way; so, when commands provided by the command line interface have no SNMP equivalent, there is no longer a need for `expect` scripts to emulate interactive telnet sessions. Network management security, a highly controversial issue at IETF, can also entirely rely on Web security technologies, which a lot of people are currently working on to secure business transactions over the Web.

Two Web-based strongly distributed hierarchical technologies have been in the spotlight for the past year. The first one, promoted by Sun, IBM and many network equipment vendors, is based on Java. Since most enterprises buy vendor-specific, NMS-dependent add-ons like CiscoWorks to manage their network equipment, some people suggested to save the cost of the NMS (where both the software and the underlying Unix workstation are expensive) by using Web browsers on cheap PCs. To do so, equipment vendors, rather than support multiple platforms for each add-on, need only provide a single device-specific, platform-independent management applet, the so-called *embedded management application*. The applet, written in Java, offers a GUI very similar to the add-ons, and allows a network administrator to manage a network device with Java Remote Method Invocations, rather than SNMP. Sun's Java Management API (JMAPI) offers a set of tools and guidelines to build these applets. In this scheme, vendors save the cost of supporting many add-ons on multiple platforms, and the loss of revenue incurred by scrapping add-ons is covered by selling these HTTP servers on a per-device basis.

The second paradigm, called Web-Based Enterprise Management (WBEM), takes a more radical approach by replacing all existing protocols and object models with new ones. The point is to integrate the Desktop Management Interface (DMI), used to manage cheap desktops, with SNMP, used to manage network equipment and expensive workstations. This framework, promoted by a consortium led by Microsoft, is based on a new object model, the HyperMedia Management Schema (HMMS), a new protocol, the HyperMedia Management Protocol (HMMP), and a new environment to manage elements as objects, the HyperMedia Object Manager (HMOM). The Desktop Management Task Force (DMTF) is currently specifying schemata for the Common Information Model (CIM), based on HMMS. It is also working on SNMP/CIM, DMI/CIM and CMIP/CIM proxies, in order to integrate WBEM with existing management protocols and object models.

6 Strongly distributed hierarchical paradigms

Although weakly distributed hierarchical paradigms address several shortcomings of the centralized models, they still often prove too limited in practice. They do not cope well with mobile computing, and only partially address the need for scalability. They also lack flexibility and robustness, two features that network administrators, learning from experience, have now come to demand. To address this, new, strongly distributed technologies have emerged, based on mobile code or distributed object paradigms.

6.1 Mobile code paradigms

Picco, Vigna et al. [3] made a detailed review of mobile code paradigms used by distributed applications, which applies equally well to DNM. They define *strong mobility* as the ability of a Mobile Code Language (MCL) to allow an execution unit, (i.e., a Unix process or a thread) to move both its code and its execution state to a different host: the execution is suspended, transferred to the destination host, and resumed there. *Weak mobility*, on the other hand, is the ability of an MCL to allow an execution unit on a host to bind dynamically code coming from another host: the code is mobile, but there is no state preservation. By analyzing all existing MCLs [3, 4], they identified 3 different categories:

- *Remote Evaluation* (REV): when a client invokes a service on a server, it does not only send the name of the service and the input parameters: it also sends the code along. So the client owns the code needed to perform the service, while the server owns the resources and provides an environment to execute the code sent by the client.
- *Code on Demand* (COD): a client, when it has to perform a given task, contacts a code server, downloads the code needed from that server, links it in on the fly (dynamic binding) and executes it. So the client owns the resources and the server owns the code.
- *Mobile Agent* (MA): an MA is an execution unit able to migrate autonomously to another host and resume execution seamlessly. Conceptually, an MA can migrate its whole virtual machine from host to host: it owns the code, not the resources.

The REV paradigm can be seen as an extension of the Unix command `rsh`. The COD paradigm, conceptually, looks very much like Video on Demand. As for the MA paradigm, the choice of its name is unfortunate, but alas reflects a clash in the terminologies used by the distributed applications and DAI communities. This clash has confused a number of people, who liken the concepts of mobile code, mobile agent and intelligent agent. In DAI, a mobile agent is a full-blown intelligent agent, as we define it in section 7, with an extra property: mobility. In this sense, one could argue that there is much more to a mobile agent than just a mobile program and a preserved state.

6.2 Mobile code technologies

6.2.1 MbD

Among all the mobile code technologies that recently appeared, two ones focused on DNM: Management by Delegation and active networks. Both of them follow the REV paradigm. Goldszmidt's Management by Delegation (MbD) framework [7, 8] set a milestone in the network management research field, by demonstrating for the first time the full potential of DNM. The whole idea of MbD can be summarized in one sentence: "delegation can be used to move management functions to the data rather than move data to these functions" [8]. To the micro-management syndrome, MbD answers with large-scale distribution. To rigid servers, i.e., servers offering services defined once and for all at design time, it brings dynamic extensibility (*flexibility* as

Goldszmidt puts it), i.e., the ability to dynamically extend services by remote applications. The delegation process is fairly simple: the client sends a program, the *delegated agent*, to the server, the *elastic server*, using the Remote Delegation Protocol (RDP); this delegated agent is dynamically linked in by the elastic server; then its execution by the server is either immediate, or delayed and controlled via a scheduling system. This is made possible by a multi-threaded run-time environment providing a "software backplane where delegated programs are loaded and executed as threads in a shared address space" [8]. Processes running in this environment are known as *elastic processes*. An elastic process is "an incarnation of a program that can be modified, extended and/or contracted during its execution" [8]. There is no fixed format for delegated agents: they may be scripts, binary programs, or anything.

In 1996, two working groups were created, one by IETF and another by ISO, in order to integrate MbD, or rather a derivative of MbD, in their respective frameworks. So far, this has resulted in an Internet draft defining the Script MIB [10], and an ITU-T draft (X.753) defining the Command Sequencer management function.

6.2.2 Active networks

An active network is a network whose nodes can perform computations on packet contents, and possibly modify them. Two approaches to active network technology are possible. The evolutionary path, called the *programmable switch approach*, keeps the existing packet format and provides a mechanism for downloading programs to dynamically programmable nodes [18, 21]. The revolutionary path, also known as the *capsule approach*, considers packets as miniature programs that are encapsulated in transmission frames, and executed at each node along their path [19].

Tennenhouse's approach in active networks is similar to Morgenstern's approach in active databases: in active networks, the code is moved into the network device MIB or run-time environment, and controlled either remotely or locally; in active databases, programs are moved into the database, and rely upon internal or external triggers to get executed.

6.3 Distributed objects

Two frameworks based on distributed object technologies have been proposed by industry: one uses JMAPI, which we presented in section 5, and another uses CORBA. The Joint Inter-Domain Management (XoJIDM) group, jointly sponsored by X/Open and the Network Management Forum (NM Forum), was created to provide tools that enable interworking between management systems based on CMIP, SNMP and CORBA. The SNMP/CMIP interoperability has been addressed by the ISO-Internet Management Coexistence (IIMC) group of the NM Forum, with the translation between the SNMP and CMIP services, protocols and information. CMIP/CORBA and SNMP/CORBA [12] interworking is tackled by XoJIDM, which addresses specification translation and interaction translation. The specification translation covers the process by which information specifications are converted. Algorithms are defined for the mapping between GDMO/ASN.1 and CORBA IDL [13] and between SNMP MIBs and CORBA IDL [14]. Interaction translation consists in either the mapping of CMIP

PDUs into one or more requests or replies on CORBA IDL interfaces, or the translation between SNMP PDUs and CORBA IDL requests/replies. The XoJIDM mappings allows CORBA programmers to write OSI or SNMP managers and agents without any knowledge of GDMO, ASN.1 and CMIP, and conversely GDMO, CMIS or SNMP programmers to access IDL-based resources, services or applications without knowing IDL.

7 Cooperative paradigms

Unlike centralized and hierarchical paradigms, cooperative paradigms are *goal-oriented*. What does this mean? With the REV paradigm, agents receive programs from a manager and execute them, without knowing what goal is being pursued by the manager. Managers send agents the 'how-to', and keep the 'why' for themselves. Conversely, with intelligent agents, managers send the 'why', and expect agents to already know the 'how-to'. In service management, IAgs are typically used in negotiation, for example to get the best deal for a multimedia session from competing service providers. In network management, IAgs may be used for pattern learning; e.g., they may dynamically learn what are the peak and slack hours of a Virtual Private Network (VPN) in an ATM network, and automatically readjust the bandwidth rented from the service provider so as to reduce the bill.

Cooperative models were only recently considered by the network management community. They originate from DAI, and more specifically from Multi-Agent Systems, where people are modeling complex systems with large groups (known as *societies*) of IAgs. This research field is fairly recent and much hyped, so a consensus on the terminology has not been reached yet. For Wooldridge and Jennings, IAgs are defined by four properties [20]:

- *autonomy*: an IAg operates without the direct intervention of humans, and has some kind of control over its actions and internal state
- *social ability*: IAgs cooperate with other IAgs (and possibly humans) to achieve their goals, via some kind of agent communication language
- *reactivity*: an IAg perceives its environment and responds in a timely fashion to changes that occur in it
- *pro-activeness*: an IAg is able to take the initiative to achieve its goals, as opposed to solely reacting to external events.

These authors consider all other properties as application-specific, e.g., mobility, veracity (IAgs do not knowingly communicate false information), and rationality (IAgs act so as to achieve their goals).

For Franklin and Graesser [6], IAgs must be reactive, autonomous, goal-oriented (i.e., pro-active, purposeful) and temporally continuous (i.e., an IAg is a continuously running process). They can also optionally be communicative (socially able), learning, mobile, flexible or have a character (e.g., modeled with beliefs, desires and intentions). Since we consider IAgs in cooperative paradigms in DNM, the social ability should be a mandatory property of IAgs.

In DNM, we propose that IAgs should be:

- goal-oriented (= pro-active)
- autonomous
- reactive
- socially able
- temporally continuous

Since IAgs are based on cooperation, they are exposed to heterogeneity problems, and therefore badly need standards for agent management, agent communication languages, etc. Two consortia are currently working on such standards: the Foundation for Intelligent Physical Agents (FIPA) and the Agent Society. Among all the agent communication languages that sprang up in DAI [20], one, KQML [5], seems to be the most popular within the network management community.

8 Conclusion

To address the lack of classification in the fast moving realm of DNM paradigms, we presented in this paper a simple typology dividing up all network management paradigms into 4 different categories, according to their underlying organizational model: the centralized paradigms, the weakly distributed hierarchical paradigms, the strongly distributed hierarchical paradigms and the cooperative paradigms. We then reviewed these different management paradigms, presented their idiosyncrasies, and listed the main technologies supporting them.

Based on this typology, how could one determine what paradigm to use for a given management scenario? We tackled some elements of this issue in [11], where we introduced a model integrating all DNM paradigms. We are still working on this integrated model, and will start implementing it shortly. In the future, we will attempt to demonstrate that the coupling of hierarchical and cooperative paradigms can indeed address network administrators' perennial quest for ever richer semantics and ever more flexibility.

Acknowledgments

This research was partially funded by the Swiss National Science Foundation (FNRS) under grant 5003-045311. The authors wish to thank Jean-Pierre Hubaux for his valuable suggestions and comments, and Shawn Koppenhöfer for checking our English.

Bibliography

[1] M. Baldi, S. Gai and G.P. Picco. "Exploiting Code Mobility in Decentralized and Flexible Network Management". In K. Rothermel and R. Popescu-Zeletin (Eds.), *Mobile Agents: Proc. 1st Int. Workshop (MA'97), Berlin, Germany, April 1997*. LNCS 1219:13-26, Springer-Verlag, Berlin, Germany, 1997.

[2] R. Boutaba. *Une architecture et une plate-forme distribuée orientée objet pour la gestion*

intégrée de réseaux et de systèmes (in French). PhD thesis, Pierre & Marie Curie University, Paris, France, March 1994.

[3] A. Carzaniga, G.P. Picco and G. Vigna. "Designing Distributed Applications with Mobile Code Paradigms". In *Proc. 19th Int. Conf. on Software Engineering (ICSE'97), Boston, Massachusetts, USA, May 1997*.

[4] G. Cugola, C. Ghezzi, G.P. Picco and G. Vigna. "Analyzing Mobile Code Languages". In J. Vitek and C. Tschudin (Eds.), *Mobile Object Systems: Proc. 2nd Int. Workshop (MOS'96)*. LNCS 1222:93-109, Springer-Verlag, Berlin, Germany, 1997.

[5] T. Finin, R. Fritzson, D. McKay and R. McEntire. "KQML as an Agent Communication Language". In N.R. Adam, B.K. Bhargava and Y. Yesha (Eds.), *Proc. 3rd Int. Conf. on Information and Knowledge Management (CIKM'94), Gaithersburg, Maryland, USA, November 1994*, pp. 456-463. ACM Press, 1994.

[6] S. Franklin and A. Graesser. "Is it an agent, or just a program?: a taxonomy for autonomous agents". In J. Müller, M. Wooldridge et al. (Eds.), *Intelligent Agents III, Proc. ECAI'96 Workshop (ATAL), Budapest, Hungary, August 1996*. LNAI 1193:21-35, Springer-Verlag, Berlin, Germany, 1997.

[7] G. Goldszmidt. *Distributed Management by Delegation*. PhD thesis, Columbia University, New York, NY, USA, December 1995.

[8] G. Goldszmidt and Y. Yemini. "Distributed Management by Delegation". In *Proc. 15th Int. Conf. on Distributed Computing Systems (ICDCS'95), Vancouver, Canada, May 1995*. IEEE Press, New York, NY, USA, 1995.

[9] H.G. Hegering and S. Abeck. *Integrated Network and System Management*. Addison-Wesley, Wokingham, UK, 1994.

[10] D.B. Levi and J. Schönwälder. *Script MIB. Definitions of Managed Objects for the Delegation of Management Scripts*. IETF Internet-Draft, 1st version, November 1996.

[11] J.P. Martin-Flatin and S. Znaty. "Annotated Typology of Distributed Network Management Paradigms". Technical Report SSC/1997/008, SSC, EPFL, Lausanne, Switzerland, March 1997.

[12] S. Mazumdar. "Inter-Domain Management between CORBA and SNMP". In *Proc. 7th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'96), L'Aquila, Italy, October 1996*.

[13] S. Mazumdar and T. Roberts (Eds.). *Translation of GDMO Specification into CORBA-IDL*. Report of the XoJIDM task force, August 1995.

[14] S. Mazumdar (Ed.). *Translation of SNMPv2 Specification into CORBA-IDL*. Report of the XoJIDM task force, September 1996.

[15] K. Meyer, M. Erlinger et al. "Decentralizing control and intelligence in network management". In A.S. Sethi, Y. Raynaud and F. Faure-Vincent (Eds.), *Integrated Network Management IV, Proc. 4th IFIP/IEEE Int. Symp. on Integrated Network Management (ISINM'95), Santa Barbara, CA, USA, May 1995*, pp. 4-16. Chapman & Hall, London, UK, 1995.

[16] J. Schönwälder. "Network Management by Delegation: from Research Prototypes towards Standards". In *Proc. 8th Joint European Networking Conference (JENC8), Edinburgh, Scotland, UK, May 1997*.

[17] M. Sloman. *Network and Distributed Systems Management*. Addison-Wesley, Wokingham, UK, 1994.

[18] J.M. Smith, D.J. Farber et al. *SwitchWare: Accelerating Network Evolution*. Technical Report MS-CIS-96-38, CIS Dept, University of Pennsylvania, USA, 1996.

[19] D.L. Tennenhouse and D. Wetherall. "Towards an Active Network Architecture". *ACM Computer Communication Review*, 26(2):5-18, 1996.

[20] M. Wooldridge and N.R. Jennings. "Agent Theories, Architectures and Languages: a Survey". In M. Wooldridge and N.R. Jennings (Eds.). *Intelligent Agents. Proc. ECAI-94, Workshop on Agent Theories, Architectures and Languages, Amsterdam, The Netherlands, August 1994*. LNAI 890:1-39. Springer-Verlag, Berlin, Germany, 1995.

[21] T. Yemini and S. da Silva. "Towards Programmable Networks". In *Proc. 7th IFIP/IEEE Int. Workshop on Distributed Systems: Operations & Management (DSOM'96), L'Aquila, Italy, October 1996*.