Managing SNMP environments using Mobile SnmpSql

Raúl Oliveira, Fabien Berger, Jacques Labetoulle
Corporate Communications
Eurecom Institute
Sophia Antipolis, France

Abstract

Mobile Agents concept become an alternative method for structuring software that can manage networks and distributed systems. In this paper, we argue that the use of Mobile Agents, in conjunction with an appropriate Data Manipulation Language (DML) could provide an optimal solution to some management scenarios. For example, to manage remote networks through low bandwidth leased lines, or to manage networks of mobile hosts. The paper presents a DML like SQL (SnmpSql), specially designed for SNMP based environments, and its application to handle SNMP data in some conventional client/server modes. Then the mobility step is given, exposing situations where the usage of Mobile Agents bring clear benefits, namely for reduction of management data traffic. The paper analyses also how the association of SnmpSql with Mobile Agents, can be used to help on itinerary and work (to perform on each destination) determination, based on SnmpSql query analyze.

1 Introduction

Mobile Agents concept become an alternative method for structuring software that can manage networks and distributed systems [1]. In the following, we argue that the use of Mobile Agents, in conjunction with an appropriate Data Manipulation Language (DML) could provide an optimal solution to some management scenarios. It is indeed well known that there are a lot of situations in Network and Distributed Systems Management, where an appropriate technological solution lacks. These scenarios occur for instance when established management protocols fail to solve a specific management problem or when the appropriate protocols seem too complex to deploy or standardize. Mobile agents can provide a simple solution that reuse the already accepted management paradigms and protocols.

In this paper we first expose two network management scenarios where current approaches are not satisfactory or are very complex to deploy. Then in section 2 we present our first experiences using a modified version of SQL (SnmpSql), as a data manipulation language, to manage SNMP environments. The section 3, presents an integrated usage of Mobile Agents and SnmpSql, which is our point of view is a powerful combination. The conclusions section ends the paper with some thoughts about the advantages of using data manipulation languages to handle management information.

Remote Network Management

Remotely managing SNMP networks through low bandwidth leased lines, or more generally across public networks, is still a concern of many corporations. The problem is essentially to reduce the amount of management information transported across these lines, while keeping the ability to perform complex management tasks [10].

The solution typically proposed, adopts OSI CMIP/CMIS agents playing a proxy agent role between the TMN world and the SNMP environment. The drawback of this approach is twofold. On one hand it is not clear that the scoping and filtering mechanisms used by the proxy agent are the most appropriate to select and filter SNMP data, and on the other hand the proxy solution imposes an information model conversion between the SMI (SNMP) [9][3] and GDMO (CMIP) [6][5]).

The structure of management information (SMI) in SNMP, because of its table oriented design, allows one to model every type of relationship found in the relational model. Whereas using GDMO, it is difficult to model entities participating in relationships with cardinality distinct from "one-to-one" or "one-to-many". However, it has to be said that these relationships, as specified in GDMO, supposes containment relationships which it is not always the case.

0-8186-8476-3/98 $10.00

Finally, because to model complex relationships GDMO it is not enough, it is necessary to use the GDMO extensions or guidelines proposed in the General Relationship Model (GRM) [7]. Therefore, when relationships other than containment are used in SNMP MIBs, the model conversion necessarily conduces to a very cumbersome mapping.

Realizing that OSI proxy agents were mostly used because of their advantages in scoping and filtering information, Mobile Agents emerge as the solution specially adapted since they do not require any information model mapping. While, still reducing the management traffic, by just shifting the data processing from one side of the network to the other. If a data manipulation language like SQL is used by the Mobile Agents, they can be designed to get the desired information from the remote native management environment, implementing filtering and scoping using Boolean Algebra operations.

Managing mobile hosts

The problem with managing wireless networks is that hosts are not always reachable. Most management solutions based on traditional manager/agent paradigm have indeed not been conceived to handle the case where the managed hosts cannot be reached. The fact that Mobile hosts are not reachable it is not anymore a problem while using mobile agents, since Mobile Agent can wait until a physical connection is reestablished to travel to the managed host or come back to the management system.

We believe that Mobile Agents can play the same primordial role in network and distributed systems management as in applications using the client/server paradigm for their interactions (e.g. remote database access)[8].

Using Mobile Agents to manage mobile hosts seems at this point an obvious solution, but the question that remains to answer is if we need to adopt a complete and new management framework to manage mobile hosts networks. In the following, we will show that, using Mobile Agents, the existing SNMP management instrumentation can be reused without major changes.

2 Distributed data manipulation using SnmpSql

Distributed data manipulation is a good example where Mobile Agents technology can be used efficiently. In a typical scenario the Mobile Agent travels from one host to the another with a predefined task of obtaining some information out of a database installed in the remote host. Once the agent gets the required data it comes back. The simplest way of implement this situation, can be made though a Mobile Agent travelling to the remote host with query specified in any DML, and travelling back with the information satisfying the query.

In the relational databases field, Structured Query Language (SQL) is the mostly used DML to describe Boolean Algebra operations such as unions, intersections, joins, projections and selections. In an SNMP based management environment we think that SQL can be used with the same advantages as for distributed relational databases. The same premise in [2] was used to create virtual tables in SNMP agents.

The SNMP environment, with its MIBs distributed over the network, can be indeed viewed as a distributed relational database environment. Adopting well known principles of vertical and horizontal fragmentation, from distributed databases discipline [4], it is then possible to apply the same principles to SQL queries to be used over the distributed SNMP tables.

The power of such an approach is easily understood by the following example. A network engineer wanting to obtain all rows of a distributed SNMP table, vertically fragmented through a set of hosts on a determined domain, has just to define a query over the desired table, mentioning that the logical table belongs to a domain instead of belonging to a single host. Conventional management APIs do not provide such a level of abstraction over the managed objects [1]

This approach lead us to the design of an API that handles user issued SQL queries intending to manipulate SNMP data, and which we named SnmpSql. Three implementations around SnmpSql concepts were developed providing different ways of using it or benefiting from a relational abstraction, inherent to an DML like SQL, to monitor SNMP based environments.

Java class The first and simpler approach is just a Java class that any program can use to get access to the SNMP world using the SnmpSql principles. This solution allows any Java program to easily encapsulate this class, without major programming difficulties. This class opens an SNMP session, parses the SnmpSql queries, issues the appropriate SNMP protocol primitives, and gives back as a result a virtual table, corresponding to the query initially passed an instance of this class.

[1] It will not be easy to imagine the same abstraction applied to the OSI management framework, mainly because of the rigid management information tree (MIT).

Remote Java instance Another solution was developed to provide remote access to an instance of the SnmpSql. In this case an IDL interface was developed allowing any kind of CORBA client to invoke methods on the remote instance of the SnmpSql class. This solution delegates all data manipulation, specified in SnmpSql query, to be done in the remote server. Therefore all filtering and scoping is performed in the remote server, before any data is transmitted over low bandwidth links.

Browser SnmpSql The third and last possibility offers to a network manager a relational oriented way to browse an SNMP based network environment. From a Java enabled browser it is possible to download an applet from which it is possible to gain access to a remote SnmpSql instance [2].

Figure 1 shows how we implemented this last situation presented above, which and in fact includes features from the all three cases. The SnmpSql Engine is the Java class referred in the first case. The middle level management application represents the remote server from where an instance of the SnmpSql class can be accessed through CORBA.

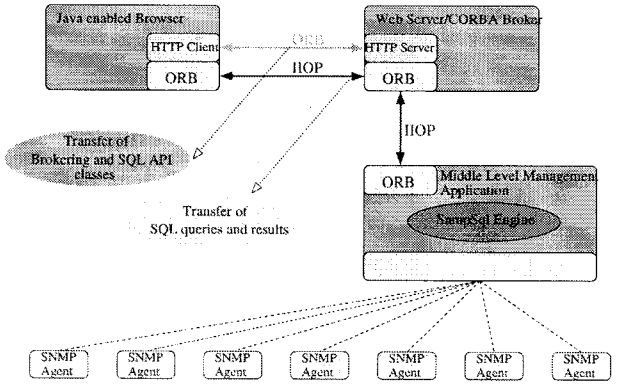

Figure 1: Global schema showing a Java enabled browser from where a network engineer can issue SnmpSql queries bringing to him SNMP data formated in a SQL style.

2.1 SnmpSql data manipulation features

In order to use SQL over SNMP based environments, we had to enhance the SQL, by adding to the base language the following management concepts: hostname (node), administrative domain, community names, and SNMP groups. These extensions to SQL, are indeed what became known as SnmpSql. As one of the advantages introduced by these extensions, it is now possible manipulate SNMP tables and groups in the same way. Of course, for a single agent query an SNMP group is viewed as a single row table. For what concerns a domain, then a SNMP group becomes a real table, where each table row corresponds to managed node.

SELECT attribute,[attribute],...,[attribute]
FROM Table,[Table],...,[Table]
WHERE <condition>

<Table>:=

<hostname>:[<community_name>]:
<snmp_group>.<table> |

<domain>:[<community_name>]:
<snmp_group>.<table> |

<hostname>:[<communit_name>]:
<snmp_group> |

<domain>:[<community_name] >:
<snmp_group>

Figure 2:Overview of the SnmpSql syntax.

Figure 2 gives an overview of the SnmpSql syntax, and shows the position in the query where are placed the new objects, that our DML needs to manipulate SNMP data. It has to be noticed that the user has not to provide the full identification of the SNMP objects (groups, tables, and variables). It is up to the API to complete the OID (Object Identifier) whether for the attributes in the clause SELECT or for the objects in the clause FROM.

From the figure above it is possible to identify two main cases (i,iii and ii,iv) that deal respectively with SNMP groups and tables.

SNMP groups are the first case handled either for a single host or across a domain. For the single host situation (**iii**) the query will return a single row table, whereas in the second situation (**iv**) it returns at least one row per host, depending on the condition specified in the WHERE clause.

SNMP tables are handled in the second case, there, the query returns a table from a single host

[2] A simple example of this browser is available at http://ia.eurecom.fr:15000/ClientApplet.htmlff

(i), or the union of a set of tables (ii) (one per host belonging to the specified domain). Of course, each resulting subtable respects the selection and projection specified in the SnmpSql query.

For all the above cases we are obliged to specify a community name as long as it is not the default one (public). The above examples show very simple queries to retrieve information found in SNMP groups and tables. These examples use basic Boolean Algebra operations (projection, selection). However, our aim is to filter and scope information in distributed SNMP MIBs. To achieve this objective the join operation has at least to be handled by the SnmpSql.

```
SELECT A_1,A_2,A_3,B_1,B_2,B_3
FROM Host_A:Group_X.Table_A,
Domain:Group_Y.Table_B
WHERE A_1=B_1

SELECT A_1,A_2,B_1,B_2, C_1,C_2,D_1,D_2
FROM Host_W:Group_X.Table_A,
Host_W:Group_Y.Table_B,
Host_Z:Group_X.Table_C,Host_Z:Group_Y.Table_D
WHERE (A_1=B_1 AND
B_2=ki) AND B_1=D_1 AND (C_1=D_1 AND
D_2=kn)
```

Figure 3:SnmpSql query showing a join operation between two tables in two different hosts

In figure 3 we show two examples of possible queries. The first query shows a join operation between a table from a MIB domain (host domain) and a table vertically fragmented over a set of hosts (SnmpSql domain). In our network management environment [3], this corresponds to an operation that allows a management application to navigate across resources, starting from a host that owns an overall view of a domain.

[3]For each SnmpSql there are a domain MIB, where the composition of the domain is stored, as well as summary information about all the systems systems and services offered in the domain

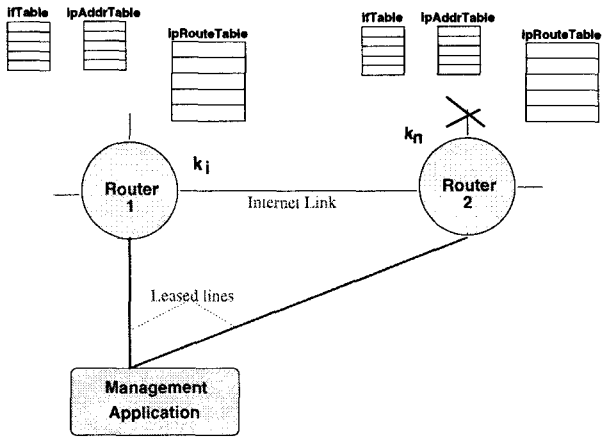

Figure 4: Shows that if the interface **kn** on router 2 is broken, then a certain number of route destinations, going out through interface **ki** on router 1, will also be affected.

The second case on the same figure allows the management application to obtain with a single query, two internal joins on each host and a second external join between two virtual tables belonging to each host. This complex query can be applied to determine from two cascaded routers, what are the affected route destinations on the first router (interface with IP address k_i), if one interface (IP address k_n) on the second router is down. This situation is depicted in figure 4, while a SnmpSql query allowing to obtain the desired data is presented in figure 5 .

It has to be said that this situation while providing a good abstraction level to the management application developer, still obliges to a complete transference of the tables from each SNMP agent back to the management application. This for sure has a high cost over low-bandwidth leased lines.

SELECT $Router_1$.ipRouteDest
FROM $Router_1$:interfaces.ifTable, $Router_1$:ip.ipAddrTable, $Router_1$:ip.ipRouteTable, $Router_2$:interfaces.ifTable, $Router_2$:ip.ipAddrTable, $Router_2$:ip.ipRouteTable,
WHERE
($Router_1$.ifIndex=$Router_1$.ipAdEntIfIndex **AND** $Router_1$.ifIndex=$Router_1$.ipRouteIfIndex **AND** $Router_1$.ipAdEntAddr=K_i)
AND
($Router_2$.ipRouteDest= $Router_2$.ipRouteDest)
AND
($Router_2$.ifIndex=$Router_2$.ipAdEntIfIndex **AND** $Router_2$.ifIndex=$Router_2$.ipRouteIfIndex **AND** $Router_2$.ipAdEntAddr=K_n)

Figure 5:Evaluation of route destinations affected on the of the first router, by a broken interface in the second router. Practical application of the second query in figure 3

In figure 6 an alternative solution is proposed where two SnmpSql servers should be placed in host systems near to the managed systems. However now, only two intermediate result tables are transmitted, being this solution a great advance compared to the previous regarding bandwidth utilization. In fact these SnmpSql servers play the role that OSI proxy agents use to play when remote SNMP environments are being managed by an OSI manager.

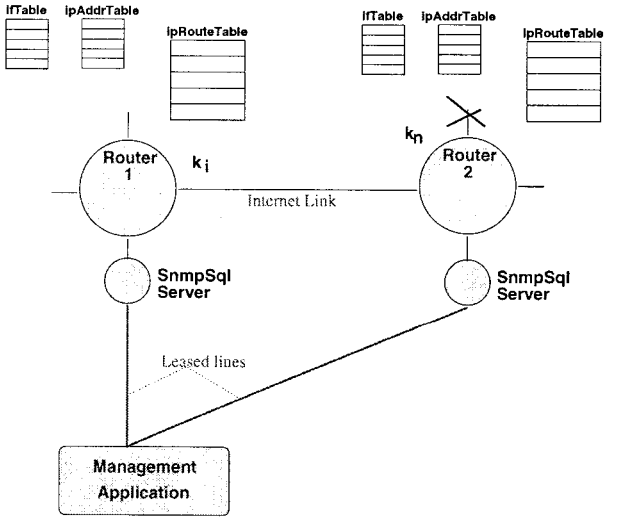

Figure 6: Usage of SnmpSql servers in the same role that OSI proxy agents have been used

3 SnmpSql mobile extension

A simple analyze over the queries presented in the previous section, reveals that a considerable amount of information must travel across the network, to allow a management application to perform the overall desired operation. One way to reduce this network overhead is to shift part of the data processing to the managed hosts or near them, as was made in the end of last section.

Up to here to shift part of the data processing and reduce management data traffic, it was not necessary to use mobile agents technology, and it was enough to send SnmpSql queries from the manager to SnmpSql server. The step that we intend to take in this section is to send the agents out of the management application with the global SnmpSql query, and let them decide the best trajectory policy by analyzing the query they carry with them.

To provide SnmpSql with mobility features, we identified three trajectory policies. The choice of a policy depends on various aspects of the manipulated data such as the amount of information, join operation optimization, and overall operation performance. Of course, these different aspects cannot be all optimized for the same query, and so it is the role of the management application to decide which trajectory policy to apply.

Choosing from one of the different trajectory policies influences the division of the global SNMP-SQL query into smaller queries. The smaller queries will be either carried by Mobile Agents to the remote hosts or used to assemble the pre-fetched intermediate tables, in order to build the final table, corresponding to the global query.

3.1 Star trajectory policy

The star trajectory policy can be used when all managed nodes are at the same distance from a determined point, and it is possible to decompose a global query in n parts, each one applying to one of the hosts implicit in the original query.

This is the case, for instance, when the management application sends a Mobile Agent to the remote managed environment, and it arrives to a Mobile Agent server at the entrance of the remote environment. There the Mobile Agent decides to splits itself in several Mobile Agents, each one now traveling with a new query that applies only to a single managed host, in order to retrieve the desired data from the SNMP agent in that host. The parent Mobile Agent must wait that all its sons come back to reassemble the result table

and then go back to the sender management application.

For the star trajectory, the query carried by all Mobile Agents is the same, if the target is to manipulate a table vertically fragmented over a set of SNMP agents. The queries are distinct if the objective is to reassemble a table horizontally fragmented, or if the global query specifies a join operation made of different SNMP tables belonging to distinct managed hosts [4].

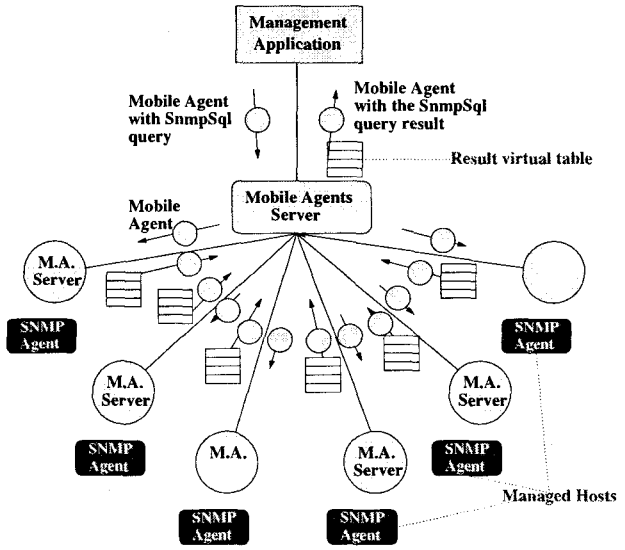

Figure 7: Trajectory policy for which one Mobile Agent is sent per managed SNMP agent, or eventually per group of managed agents

The decision of whether or not this trajectory policy should be used depends on the tradeoff between the average virtual table size of each subquery, and the code size of the Mobile Agent. The virtual table size itself depends on both the size of each particular SNMP table, and on the condition expressed in the WHERE clause of the query.

It can happen that a Mobile Agent arriving at a Mobile Agent server decides not to go further and performs all queries from the Mobile Agent server. This case would occur for instance if the amount of information to pass across the remote network it is not so important.

3.2 Shortest path trajectory policy

Shortest path trajectory policy is used everytime a mobile agent having to visit n managed hosts, in order to evaluate the global query, can optimize the time required to visit all hosts. For instance by finding the path to which corresponds the minimum amount of time required to visit all hosts. In this case the most important is the time spent to obtain the result of a global query.

Therefore this type of trajectory policy can be applied when, for example, the objective is to reassemble a table vertically fragmented over a set of scattered mobile hosts. Being this set of hosts easily reached among themselves than from the management application. In this case, the performance of reassemble operation can be greatly improved if the agent travels across all mobile hosts, before coming back to the management application.

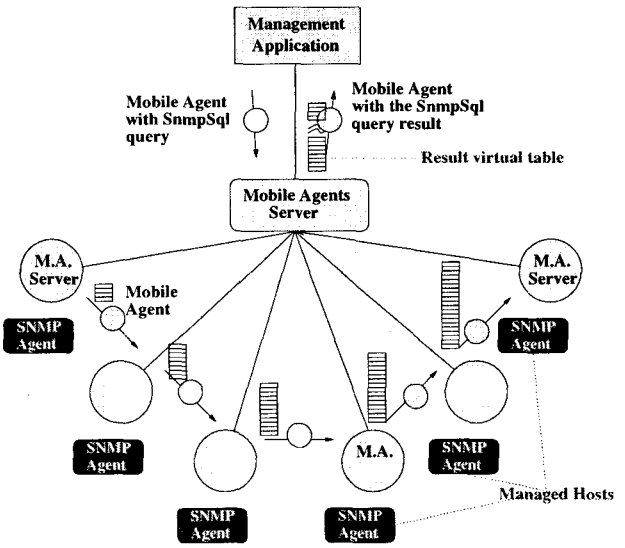

Figure 8: Shortest path trajectory policy

A query corresponding to this situation is the query on figure 3.2. At the Mobile Agent server, in the border between the fixed network and the wireless environment, the Mobile Agent inquires a domain server for the domain composition. Based on the answer, and if all hosts are mobile, the Agent decides to initiate a trajectory across all those hosts.

3.3 Iterative trajectory policy

This particular trajectory policy applies when the query, to be efficiently performed, requires that the agent carries along the path the intermediate results. The routers problem presented above (see Figure 4) is a good example of an application of this trajectory policy.

[4]Note: in order to join two tables, there must exist a relationship between two columns of those tables

```
SELECT A_1,A_2,A_3
FROM Domain:Group_Y.Table_a
WHERE A_2=kn
```

Figure 9: Typically case where the shortest path trajectory policy can be applied

For the routers problem, as we demonstrated above, a management application has two options. For the first option it retrieves all tables from both routers (SNMP agents) and then processes these tables in order to obtain the query result. In the second case the management application sends two partial queries to remote SnmpSql servers and the with the two partial results it build the final query result.

By using a Mobile Agent approach, the management application can still reduce the network used bandwidth.

First a Mobile Agent with the global SnmpSql query is sent to a Mobile Agent Server near the first router. The the Mobile Agent applies the subquery presented in the figure 10.

```
SELECT ipRouteDest
FROM Router1:interfaces.ifTable,
Router1:ip.ipAddrTable, Router1:ip.ipRouteTable
WHERE (ifIndex=ipAdEntIfIndex AND
ifIndex=ipRouteIfIndex AND ipAdEntAddr=Ki )
```

Figure 10:$Router_1$ internal join operation, that gives as a result the list of all destinations associated with interface **ki** in the router 1.

In order to understand the advantages of using the iterative trajectory policy, we assume that the tables containing IP routes destinations, in each router, have an average of 10000 destinations, and these destinations are equally distributed through all interfaces. For an eight interfaces router the Agent will carry to the second router a intermediate result table of 10000/8 rows.

```
SELECT ipRouteDest
FROM Router2:interfaces.ifTable,
Router2:ip.ipAddrTable, Router2:ip.ipRouteTable
WHERE (ifIndex=ipAdEntIfIndex AND
ifIndex=ipRouteIfIndex AND ipAdEntAddr=Kn
)
```

Figure 11:$Router_2$ internal join operation,that gives as a result the list of all destinations associated with interface **kn** in the router 2

Arriving at the Mobile Agent server near to the second router the Mobile Agent obtains the second virtual table by issuing the query presented in figure 11. With the two virtual tables the Mobile Agent performs the last join operation represented in figure 12, and finally comes back home with a table whose estimated number of lines is around 1250/8, assuming the 1250 destinations of the interface **ki** of the first router are equally distributed through all the interfaces in the second router.

```
SELECT VirtualTable1.ipRouteDest
FROM VirtualTable1,VirtualTable2
WHERE (VirtualTable2.ipRouteDest=
VirtualTable1.ipRouteDest)
```

Figure 12:External join operation, that in fact performs an intersection of the two partial queries.

The previous example shows us that it is possible to reduce the data transported across the leased lines or the public network by a ratio of approximately of 1/128 by using Mobile Agents. While with remote SnmpSql servers the best ratio obtained was 1/8.

4 Conclusion

The work presented throughout this paper shows that SQL is a powerful data manipulation language, that can be easily adapted to handle distributed management information for SNMP based environments, as the developed SnmpSql is a clear example. We also showed that DMLs have finally a great affinity with the Mobile Agent concept. Therefore it is not surprisingly the successful association between a Mobile Agents platform and the SnmpSql. Specially for

what concerns a less dependency of rules, to evaluate itineraries and tasks decomposition, on heuristics that otherwise should be carried by the Mobile Agents. Concerning the reduction the traffic of management data, the examples shown are very clear regarding the advantages of Mobile Agents usage. Nevertheless, it has to be said that the great improvement was done with the creation of SnmpSql, and that is why in this article a complete section was dedicated to it.

As final comment we think that Mobile Agents are an interesting approach to be used in management frameworks, not only because of the performance advantages, but essentially by the flexibility and robustness provided for the development of management applications.

References

[1] S. Appleby and S. Steward. Software agents for control. In P. Cochrane & D.J.T. Meatley, editor, *Modelling Future Telecomunications Systems - BT Telecommunications Series.* Chapman & Hall Publisher.

[2] Kazushige Arai and Yachiam Yemini. Mib view language (mvl) for snmp. In *Fourth International Symposium on Integrated Network Management*, pages 455–465, 1995.

[3] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol (SNMP), RFC 1157, May 1990.

[4] Stefano Ceri and Giuseppe Pelagatti. *Distributed Databases Principles and Systems.* McGraw-Hill Internacional Editions, 1984.

[5] Management
Information Protocol Specification - Common Management Information Protocol, ISO/IEC 9596-1, ITU X.711.

[6] Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects, ISO/IEC 10165-4, ITU X.722.

[7] ISO/IEC JTC 1/SC 21 – Information Technology – Open System Interconnection – Data Management and Open Distributed Processing – Structure of Management Information – Part 7 : General Relationship Model., mar 1994.

[8] Colin G. Harrison, David M. Chess, and Aaron Kershenbaum. Mobile Agents: Are they a good idea? Technical report, IBM Research Division, T. J. Watson Research Center, March 1995.

[9] M. Rose K. McCloghrie. Structure and Identification of Management Information for TCP/IP-based Internets, RFC 1155, IAB, 1990.

[10] T. Magedanz, K. Rothermel, and S. Krause. An emerging Technology for Next Generation Telecommunications. March 1996.