MODELLING NETWORK AND SYSTEM MONITORING OVER THE INTERNET WITH MOBILE AGENTS

Antonio Liotta[*], Graham Knight, George Pavlou

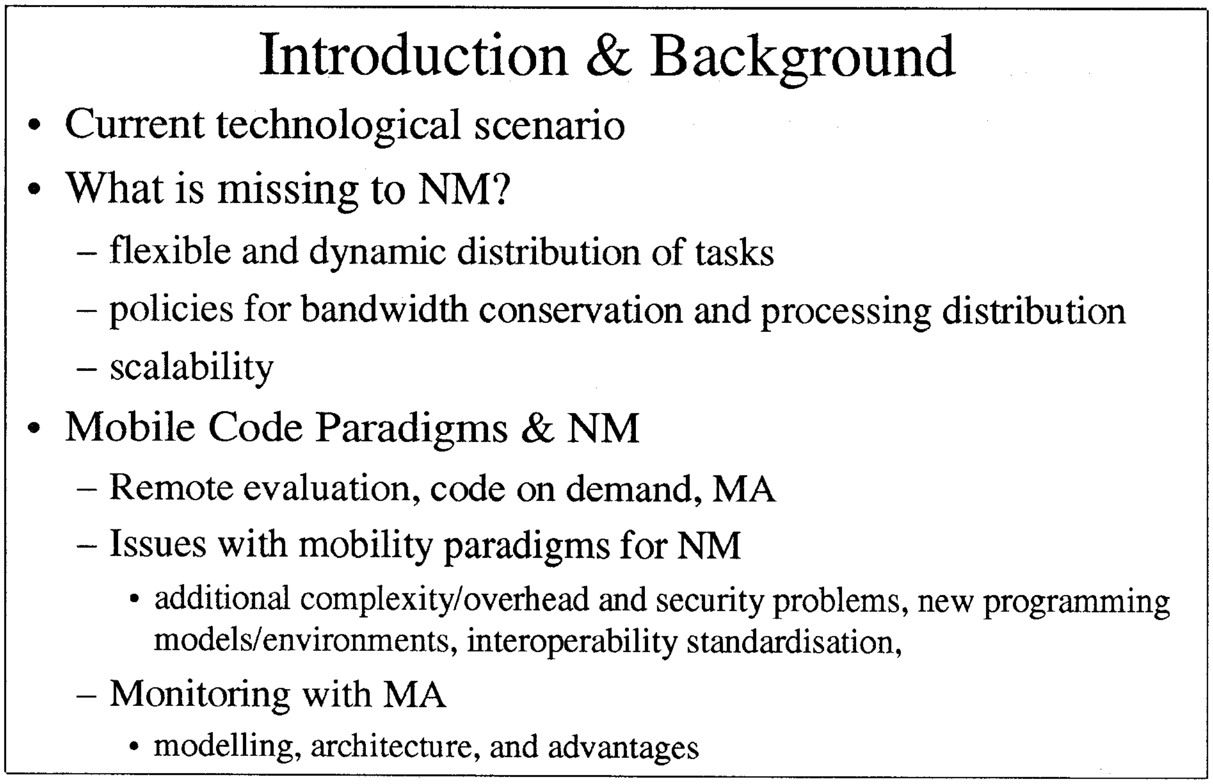
Department of Computer Science - University College London
Gower Street - London WC1E 6BT
Phone: +44-171-419-3679, Fax: +44-171-387-1397
E-Mail: A.Liotta@cs.ucl.ac.uk

ABSTRACT: Distributed Network Management is gaining importance due to the explosive growth of the size of computer networks. New management paradigms are being proposed as an alternative to the centralised one, and new technologies and programming languages are making them feasible. The use of Mobile Agents (MAs) to distribute and delegate management tasks is a particularly promising approach to dealing with the limitations of current centralised management systems which appear to be lacking flexibility and scalability.

This paper is focused on the impact that mobile code paradigms can have on distributed network and system monitoring. A dynamic, hierarchical management model based on a delegation paradigm is adopted and an MA-architecture for monitoring operations is proposed. Finally, possible uses of the proposed model and architecture for pursuing seamless and timely monitoring are discussed.

Keywords: Management by Delegation, Mobile Agents, Distributed Network Monitoring, Monitoring Modelling.

[*] During the course of this work, Antonio Liotta was supported by a research grant from Hewlett-Packard.



Introduction & Background

- Current technological scenario
- What is missing to NM?
  - flexible and dynamic distribution of tasks
  - policies for bandwidth conservation and processing distribution
  - scalability
- Mobile Code Paradigms & NM
  - Remote evaluation, code on demand, MA
  - Issues with mobility paradigms for NM
    - additional complexity/overhead and security problems, new programming models/environments, interoperability standardisation,
  - Monitoring with MA
    - modelling, architecture, and advantages

Introduction

Both data and telecommunications networks are undergoing dramatic changes . The data networks scenario is shifting from a Local Area Network (LAN) level to a world wide one, scaling up very rapidly. The telecom scenario is inevitably shifting from analog to digital transmissions. Furthermore, these two worlds, which used to be almost completely separated in the past, now seem to be rapidly converging [23]. At the same time, with the general acceptance of the Internet, the communication paradigm seems to be changing as well. New, so called "push technologies" that might provide means to extend the client-server "pull" model dynamically, are becoming popular.

As a consequence of this sensational evolution of both computer and communications technologies, network managers are now demanding Distributed Network Management (DNM) [24] in contrast with more traditional centralised or weakly distributed hierarchical management models [2, 4]. Traditional management has, in fact, proven to lack flexibility [6, 7, 8, 10, 14, 17] and scalability [11, 13, 21, 22] and fit badly with the current technological trends - which are characterised by an increasing availability of distributed computational resources contrasted by a moderate increment of bandwidth available over the Internet. For instance, in the SNMP management, which is mostly based on the client-server communication paradigm, tasks are statically defined, tend to use up a significant portion of network bandwidth, and often result in an ineffective distribution of computational load [11]. Therefore, traditional management tends to exploit the most critical resources of large scale networks - that is, network bandwidth and latency - and not to take full advantage of the resources that seem to becoming widely and inexpensively available in the network elements - that is, processing capability and memory.

As a result, network managers are keen to consider new management paradigms, which can fulfil the scalability, flexibility, and robustness requirements of the current and future massive distributed systems. The *Management by Delegation* (MbD) *paradigm* [10] seems to have achieved a widespread interest [6, 9, 15, 16], to have triggered many similar investigations [7, 8, 17, 18, 19, 20], and to be a particularly promising approach to deal with the limitations of traditional management systems.

Mobile Code Paradigms and Network Management

Mobile code paradigms have recently raised interest as a new family of programming languages, usually referred to as mobile code languages (MCLs) [1], have become popular over the Internet. A review of mobile code paradigms that can be considered to be relevant to distributed network management applications can be found in [2, 4]. The authors identify two categories of mobility: *strong mobility* as the ability of an MCL to allow an execution unit to move both its code and its execution state; and *weak mobility* as the ability of an execution unit on a host to bind dynamically code coming from another host.

Carzaniga *et al* [3] also classify three categories of mobile-code paradigms: *Remote Evaluation* (REV), *Code on Demand* (COD), and *Mobile Agents* (MAs). In the REV paradigm a client can send the code describing the service to the server. The server owns the resources and provides an environment to execute the client's code. In contrast, with COD, the client downloads, links dynamically, and executes the code he requires from a server. So, the client owns the code and the server owns the resources.

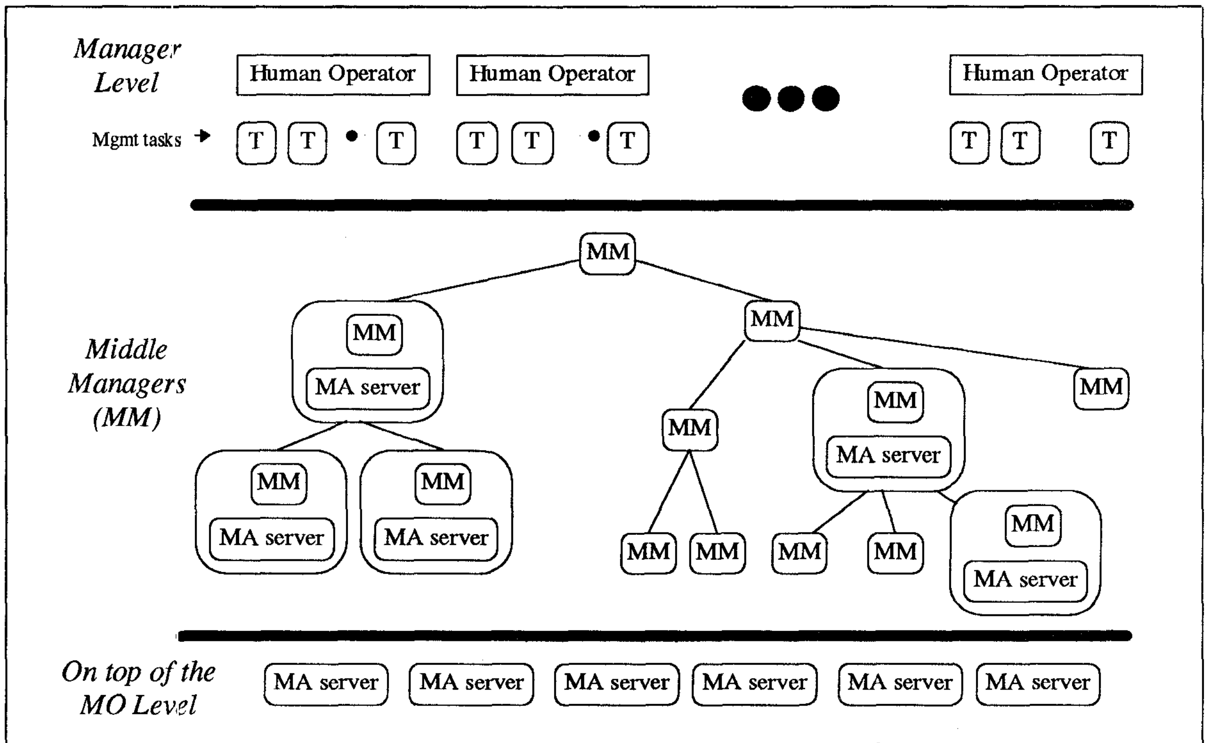


Figure 1. A hierarchical, dynamic Management Model.

Finally, within an MA paradigm the MA owns the code to perform a service but does not own the resources needed to accomplish it. Furthermore, the MbD paradigm can be integrated into this classification and considered to be a REV paradigm with remote control capabilities, as proposed in [4].

The adoption of mobile code paradigms can affect profoundly the way management is done [2, 12, 18, 20], enabling a dynamic distribution of management intelligence, and leading to a more rational exploitation of the newly and increasingly available technological resources. A proper use of these paradigms may lead to more scaleable, flexible, robust, timely and seamless management systems [2].

However, despite the great deal of interest of researchers aiming at an effective merge between distributed network management and code mobility paradigms, this merge still poses a number of very interesting, as well as hard, problems to tackle. Mobile-code-based management systems promise to be quite complex both to design and maintain, in contrast with the relative simplicity of static centralised or distributed models. Additional security issues have to be considered too. Finally, some of the other open issues include new suitable programming models, integration and interoperability with legacy systems, design and standardisation of a feasible infrastructure and its basic services (to manage mobile management tasks).

As a result of these numerous open issues we decided to focus this paper on a more specific management area, that is, network monitoring. The aim is to consider the impact that mobile code paradigms can have on monitoring and the need for semantically rich modelling and for a feasible MA-based hierarchical architecture. It is, then, focused on some of the issues related to the use of MAs and on two interesting application. First, the use of migrating MAs for implementing policies aiming at minimising bandwidth consumption by monitoring tasks. Then, the application of MAs for time-critical monitoring operations over the Internet.

A Hierarchical, Dynamic Management Model

In Figure 1 we show our view of some of the properties an MA-based management model should have in order to try to exploit the advantages of mobile code paradigms. We term "MA server" an execution environment supporting MAs. We assume that this environment is a layer built on top of the Managed Object (MO) level [26].

1. *Semantic richness*. We would distinguish between human manager level and Middle Managers (MM) level. The first level is basically a high-level interface between managers and the rest of the management system. This interface should provide a semantically rich model allowing for a high level definition of management tasks [4].

2. *Dynamic Hierarchical Management*. MMs are organised in a hierarchy resembling other management architectures (e.g., the OSI/TMN one [26]) and, thus, sharing similar advantages and disadvantages. However, despite the similarities, the manager hierarchy can be defined dynamically in contrast with the static solution adopted, for example, by the OSI/TMN. For instance, it is possible to define and deploy MMs and their respective scope dynamically.

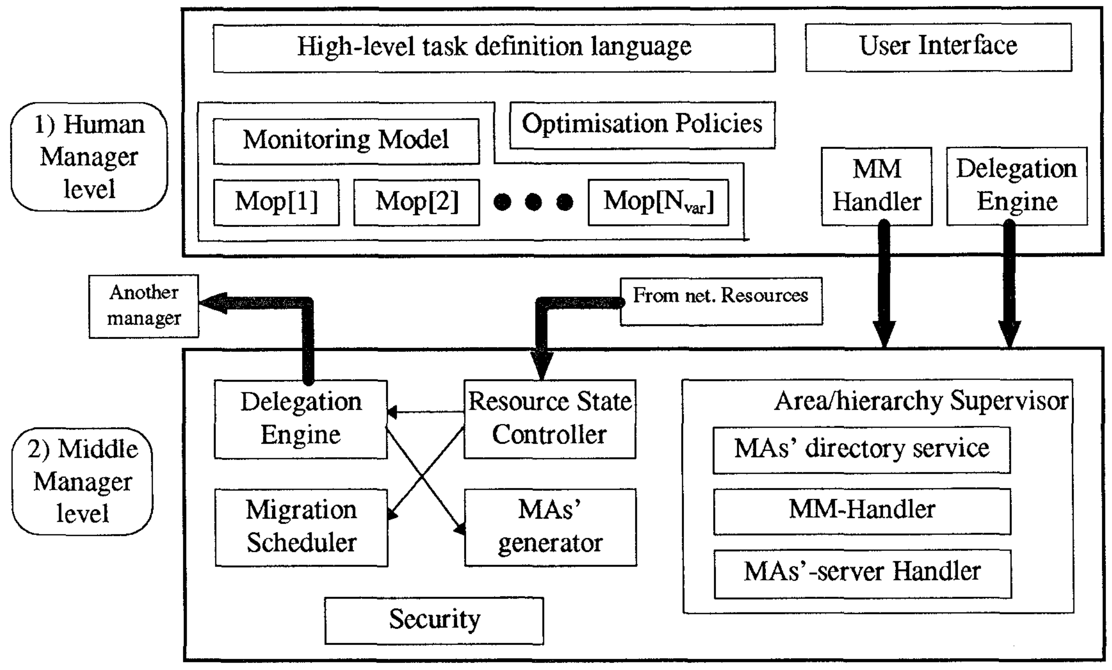


Figure 2. An MA-architecture for managers.

Consequently, the management granularity can be changed accordingly. An MM is in charge of managing MAs under its supervision. It monitors its MAs activities and enforces MAs monitoring optimisation policies. If, for example, an MM realises that more MMs could manage the same MAs more effectively, it can decide to generate other MMs and delegate part of its MAs management responsibilities accordingly.

3. *Dynamic delegation of management responsibilities.* An MM can dynamically delegate part of or all its management responsibilities to its lower MMs. For example, if it can decide to delegate part of the control it has on a set of MAs to other MMs.

4. *Dynamic delegation of management tasks.* Tasks are defined by human managers. Then, they are pre-processed and delegated to MMs. MMs can in turn decide to either execute them or to delegate them to one or more MMs. Tasks decentralisation is, thus, obtained through delegation of task, and it is i mplemented with MAs.

5. *Dynamic migration of management tasks (MAs).* Delegated MAs are under the control of their master MMs. MMs monitor MAs in order to find out their best location and trigger their migration accordingly. This is done with the main target of trying to minimise the impact that monitoring tasks have on the available bandwidth. Obviously, MAs can only be executed in nodes that have an MAs execution environment (MA server).

Therefore, it is of fundamental importance that tasks are modelled in a way which makes feasible their semantically rich definition by a human manager and their automation (through the delegation hierarchy), and allows for the introduction of migration policies targeted at an optimal use of the limited network resources.

An MA-Architecture

A possible 4-layered functional architecture for the hierarchical, dynamic, model described above is discussed in Figure 2 and Figure 3. There are two main groups of functions: managers functions (Figure 2) and MAs functions (Figure 3).

At the first level, *human managers* (or, simply, managers) can use a *high-level language* to define its management tasks (monitoring tasks in our context) and the *migration optimisation policies.* This requires the use of a semantically rich modelling of the monitoring activities [4]. The *monitoring model* should allow for the automation of monitoring tasks which is based on mobility and is controlled by delegation. It should also permit the enforcement of migration policies targeted at an optimal use of the network resources.

A possible way to model monitoring operation in such a mobility-aware manner is to consider it as made of three main parts. The first is needed in order to define the monitoring entities that will be involved (mainly their physical location and their unique identifier). A second part defines a set of parameters characterising the monitoring static features - e.g., the sampling period, S_p, the time interval between to polling samples. Finally, a third part can define the dynamic monitoring features - e.g., monitoring operation M_{op} - that is, the functions that can be changed regardless of the static ones.

MM-handlers and *delegation engines* have the same function at both the 1st and 2nd layer. The former can dynamically generate or kill a new MM in the management hierarchy. The latter can delegate to an MM part or all of its management responsibility and tasks. The results of the monitoring tasks in which the manager is interested in are then reported through the *user interface.*

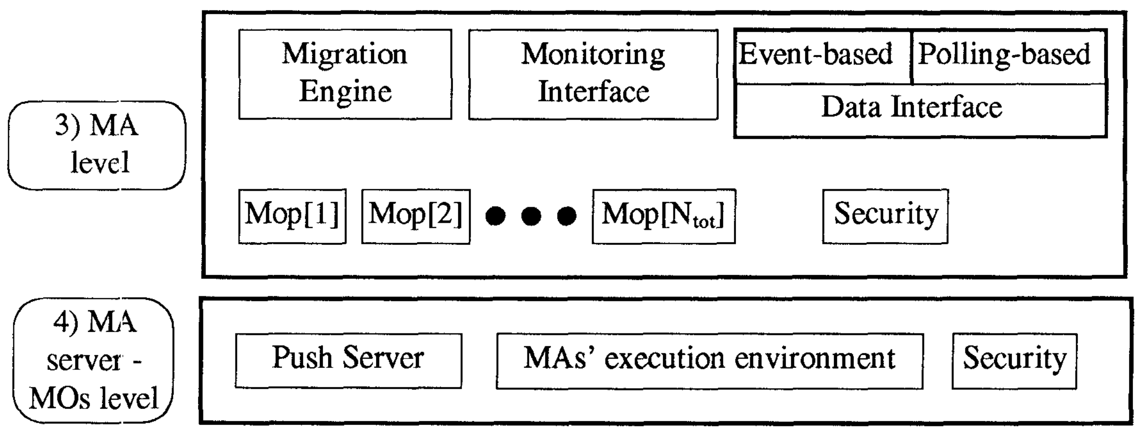


Figure 3. MA architecture: the MA Layers.

The MM *supervisor* analyses the monitoring tasks before they are deployed as MAs in order to find out whether other MAs are carrying out operations which might be relevant to the new tasks. The supervisor can query the MAs listed in its *directory service* in order to find out relevant MAs. If it finds out that an existing MA is already monitoring an entity he is interested in and that the respective monitoring parameters are compatible, it will inform the *MA-generator* accordingly. In this case the MA-generator will generate an MA which subscribes to the pre-existing MA data interface rather than monitoring the actual managed objects.

The MA migration capability is of fundamental importance to target seamless and timely monitoring. A migration decision should consider both the optimisation policies and the state of the available network resources. The smarter is the decision, the better the result can be. However, it is not straightforward to figure out "where" the "intelligence" should be placed, whether in MMs or within MAs. Trying to deal with this issue we distinguish two different blocks: the *migration scheduler* and the *migration engine*. The first plays the smart part of the migration process. It is able to check the resource availability and to enforce the migration policies according to the current active tasks. It is also able to accept new policies through a delegation process.

Location of MAs Intelligence

The migration engine, instead, plays the mechanical part of the migration process: if the scheduler triggers a migration action the engine is able to implement it. Therefore, both the intelligence and the autonomy are coded in the scheduler, whereas the migration mechanisms are in the engine. It seems quite obvious that the migration engine has to be put into the MA. However, it is not so obvious to decide where the scheduler should be placed. Two solutions can be considered:

1. *Scheduler in the MM.* An MM can monitor and control one or more MAs in order to apply placement optimisation policies. An MM has a relatively global view of the MAs activities and, thus, it might be able to take better migration decisions. On the other hand this solution will introduce an MM-to-MA communications overhead. Besides, MAs are not completely autonomous entities as MM have to control them. Thus, MMs represent a single point of failure.
2. *Scheduler in the MA.* MAs are fully autonomous and the MM-to-MA communications overhead can be reduced significantly. However, an MA tends to have a local knowledge which might represent an obstacle to effective migration decisions (that should have a more global view). Moreover, smart MAs will be more complex, both to design or maintain. They will also be bigger in size and, thus, introduce a greater migration overhead. An MA could have a more global view, but it would require extra services, like the ones provided by the MM area supervisor. Besides, it could be difficult for an MA to maintain a consistent view of the world. This would, again, increase its complexity and size. Another drawback of this solution is that in order to change the migration policy every single MA should be modified.

In our view when a hierarchical, dynamic, management model (such as the one we described above) is adopted, the migration scheduler should be put within the MM. This would allow the application of global migration policies by MMs at higher level. The migration overhead could be reduced by increasing the number of levels in the management hierarchy and delegating the migration responsibilities to lower level MMs. Finally, this solution would still allow fully autonomous MAs. This can be obtained by introducing an MM whose management scope is restricted to a single MA.

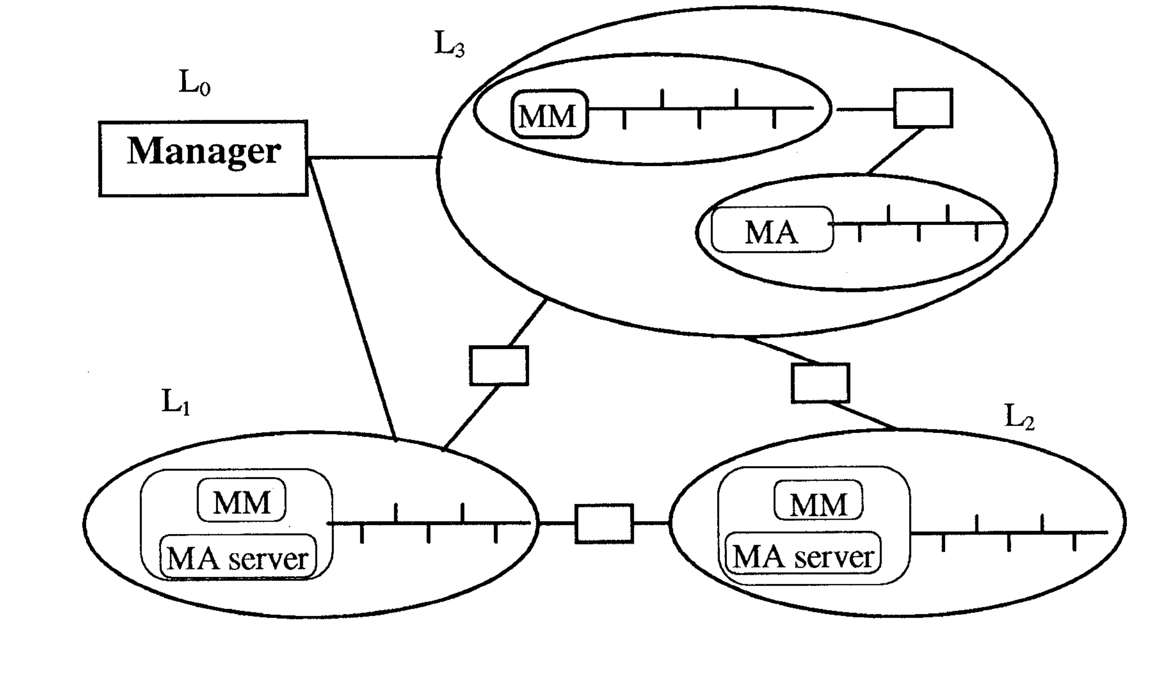


Figure 4. Mobility and Network Load.

The *monitoring interface* is another important module. It provides a view of the monitoring operations and their parameters. Managers can, thus, query an MA through this module in order to find out whether a new task can take advantage of its services. Finally, the *data interface*, presents the results of the MA current monitoring operations M_{op} to other MAs or to managers. There are two ways to report this information. One is via notifications of events to the entities that have subscribed for them. Another is via polling. Any entitled MA can in fact collect the monitored data by polling the MA. At the last layer the *MA-server* has the capability to accept requests of MAs activation from managers (*Push server*), to check out security constraints, and to provide a suitable MA execution environment.

Optimization Mechanisms

The management model and architecture described above can be used to target a reduction of the impact that monitoring operations have on the available network bandwidth. In particular, the flexibility of this model allows for the application of several possible optimisations. Hierarchical management [22], delegation of management responsibilities [13, 16], and distribution of tasks via delegation [9, 12, 18], can all lead to a reduction of use of bandwidth for management purposes. However, when these three features are joint together - like in the proposed model - more sophisticated optimisation mechanisms might be thought.

We assume that a WAN is made of a collection of WANs and LANs which, in turn, are composed of different sub-networks. Thus, a WAN can be seen as divided into areas, made of sub-areas, and so on (Figure 4). From this perspective a monitoring task, defined by a human manager, can be analysed in terms of the areas, or "locations", involved in the monitoring activity and accordingly split into many sub-task. Similarly, MMs can be considered to be bound to specific areas, like the area managers described in [21]. The delegation process can forward monitoring sub-tasks all the way through the MM hierarchy till the proper area MM is reached. Given these assumption, the following bandwidth optimisation mechanisms can be adopted (Figure 4):

1. *Distribution via Delegation.* This split-and-forward process can represent a first bandwidth optimisation policy. For example, if a monitoring task targets two entities, placed in locations L_1 and L_2 respectively, it may be reasonable to split the given task into two separate tasks and, then, forward them to locations L_1 and L_2 respectively. This is basically a distribution of monitoring tasks via delegation [9]. The deeper is the MM hierarchy, the smaller is the monitoring loop (between MMs and monitored resources), and the better the optimisation in the use of the bandwidth.
2. *Avoidance of task duplication.* Each area MM is able to apply another optimisation policy. It checks out whether there is any other MA which is carrying out operations that might be relevant to the new sub-task. If this is the case the MA-generator creates an MA whose task is to get the information from the pre-existing MA. Thus, the situation where more than one MAs monitor the same entity is avoided whenever possible. This in turn may lead to a further reduction of network load. For example, if two MAs monitor the same entity via polling, half of the bandwidth used up for this task can be saved.
3. *Adaptation through migration.* MM migration schedulers can monitor the available resources and the current active monitoring tasks under their supervision. If they detect that an MA is operating through a congested link or detect a failure they can migrate it into a better location.

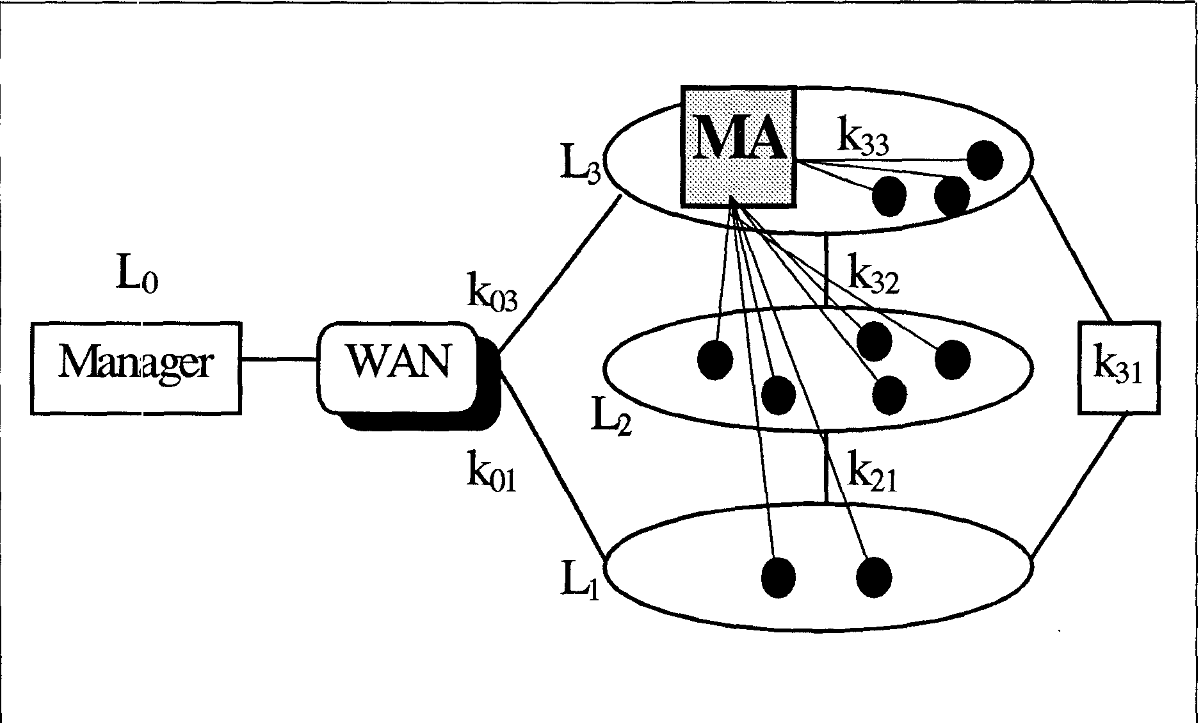


Figure 5. Cost Functions for MAs

4. *Change of management granularity.* Delegation of monitoring responsibilities among different MMs can allow to change the granularity of the MAs optimisation policies. If a set of MAs are supervised by lower level MMs then the migration policies are applied at a smaller granularity (only local information is used). More global optimisation policy can be applied delegating responsibilities at higher level MMs.

A Cost Function for MA Configurations

Although mobility can be beneficial for the purpose of a monitoring task, overheads introduced by MAs and MMs - e.g., due to their deployment and management - should be accounted for very carefully. Different configurations for a set of monitoring MAs can result in a dramatically different network load. Besides, a simple centralised, polling-based, solution might result as being the least bandwidth-consuming configuration for a class of monitoring tasks.

It is, then, crucial to define "Cost Functions" that can help "configuring" MAs in the best way and estimating the corresponding overheads. As an example of the potentiality of the monitoring model and functional architecture introduced above, we now present a simple cost function that can be used to characterise the cost of an MA configuration in terms of network load.

Several simplifying assumptions are made for the purpose of this example (Figure 5). We consider a single "poll" to be the collection and analysis of a set of "n" values. The interval between "polls" we term the "analysis period" (A_p). The intervals between collection of individual values within a poll is the "sampling period" (S_p). Clearly $A_p >= n * S_p$. We term "observation period" (O_p) the lifetime of a monitoring operation; "data resolution" (b_s) the number of bits used to code each sample.

We use "cost coefficients" $k_{i,j}$ to weight the network load, injected between locations L_i and L_j by a given monitoring task, according to the network properties (e.g., type of network, availability of bandwidth, congestion state, etc.). For example, if we consider the cost coefficients to be proportional to the reverse of the link capacity, in Figure 5 we generally have:

$k_{01} \approx k_{03} >> k_{31} \approx k_{32} \approx k_{21} >> k_{33}$

A simple "poll" interaction between a managing location, L_0, and a managed location (L_1, L_2 or L_3) is modelled as a special case of an MA-based interaction. This "poll" is equivalent to an MA configuration with the MAs themselves being deployed in L_0. The communication load associated to the MA deployment is, then, accounted for zero in this case.

A simple cost function, characterising the bandwidth consumption of a given configuration of monitoring MAs can be expressed as:

$C_{conf} = C_{depl} + C_{coll} + C_{deliv}$ (Equation 1)

where the three terms represent the cost for deploying MAs, collecting data from the monitored locations to the MAs, and delivering data from MAs to management locations respectively. If we suppose that there is only one MA to configure and deploy, the above cost terms can be expressed as follows:

$C_{depl} = k_{0x} * B_{MA,act}$ where $B_{MA,act}$ is the bandwidth used up in order to transmit the MA from the managing location, L_0, to the remote monitoring location, L_x (L_3 in Figure 5);

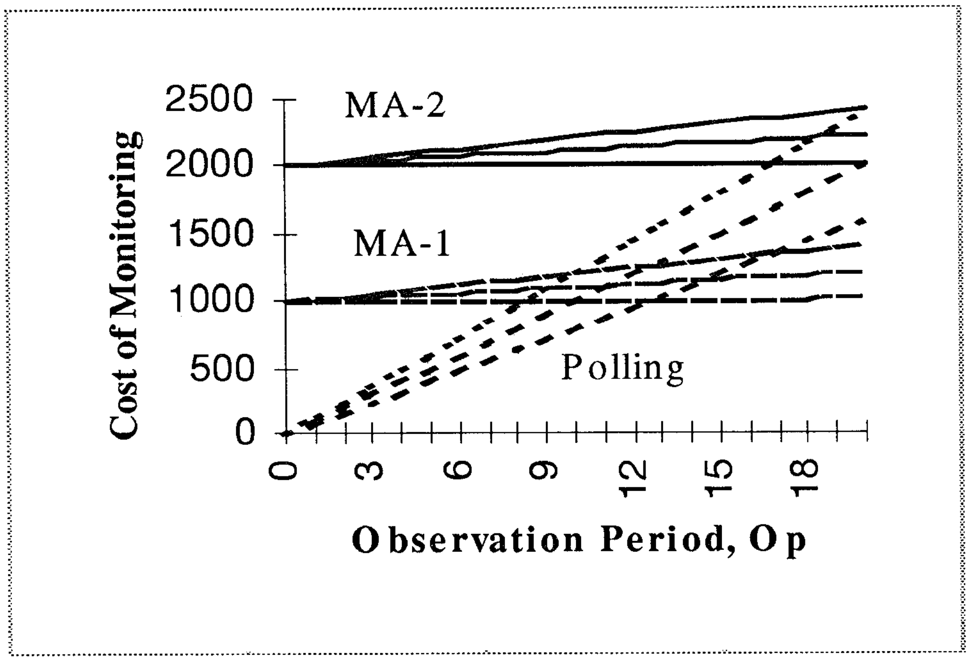


Figure 6. Costs for Monitoring against MAs life-time

$C_{coll} = \sum_i k_{x,i} * \left(b_{s,i} / S_{p,i} \right) * O_p$ where the Σ is extended to all the monitored objects;

$C_{deliv} = k_{x,0} * \sigma * B_{coll,x}$ where $B_{coll,x}$ represents the total amount of monitoring data collected by the MA and σ is the "selectivity" of the MA that is the ratio between the amount of data the MA delivers to the manager, $B_{deliv,x}$ – i.e., traffic between L_x and L_0 – and $B_{coll,x}$. The terms of σ may vary considerably, according to the kind of monitoring operation. For example:

a) For a "logging" activity a manager may require a report of B_{summ} bytes, summarising the monitored data, every A_p seconds, so

$B_{coll,x} = \sum_i \left(b_{s,i} / S_{p,i} \right) * O_p$; $B_{deliv,x} = B_{summ} / A_p * O_p$

b) For a "polling-based alarm detection" activity the MA polls the relevant entities and filters out the collected data in order to generate alarms. Alarms of size b_{alarm} are generated with probability P_{alarm}, then

$B_{coll,x} = \sum_i \left(b_{s,i} / S_{p,i} \right) * O_p$; $B_{deliv,x} = P_{alarm} * b_{alarm}$

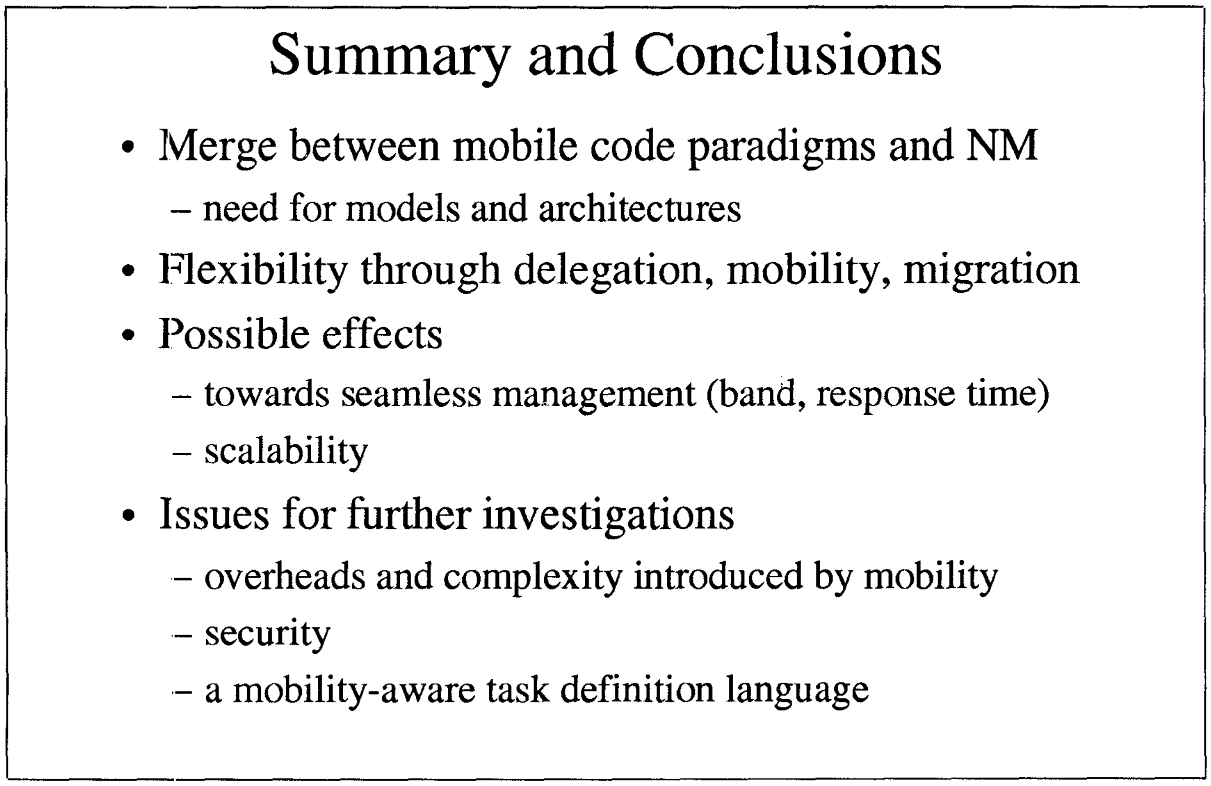
c) For an "event-based alarm detection" activity the MA collects events of size b_{event} and probability P_{event} and generates alarms as in the previous case, then

$B_{coll,x} = P_{event} * b_{event}$; $B_{deliv,x} = P_{alarm} * b_{alarm}$

In Figure 6, three families of lines derived from equation 1 and corresponding to 3 different MA configurations are plotted. The slops of the lines depend mostly on the involved cost coefficients and on the relative weight of the 3 terms of equation 1. It is important to notice that the slops relative to a "polling" configuration – i.e., MA in L_0 – can be much greater that the ones corresponding to an actual MA configuration – e.g. MA in L_1, L_2 or L_3 – due to the high contribution of the second term in equation 1. The graph shows, however, that very short monitoring activities tend to be less bandwidth consuming if performed using polling. In contrast MA-based monitoring tends to be a better choice if O_p is as long as to compensate the bandwidth overhead due to the transmission of the MA.

The ideal situation would be to design MA-based monitors with cost lines characterised by a much smaller slope than the one their corresponding polling-based system would have. This can be achieved if the MA monitors a sufficiently high number of variables and/or if it performs a sufficiently high-level processing or filtering of raw data.

From the above analysis it appears evident that the decision on whether to use an MA-based approach or a centralised one is a trade-off decision that relies on two main things: the observation period and the level of pre-processing that MAs can perform on raw data - i.e., the MA selectivity. If they are both relatively high then the MA approach is usually more convenient than the polling one, in terms of bandwidth consumption. Finally, it is worth noting that the same cost functions can also be used to apply MAs migration policies.



Summary and Conclusions

- Merge between mobile code paradigms and NM
  - need for models and architectures
- Flexibility through delegation, mobility, migration
- Possible effects
  - towards seamless management (band, response time)
  - scalability
- Issues for further investigations
  - overheads and complexity introduced by mobility
  - security
  - a mobility-aware task definition language

Improving Timeliness through Mobility

The management model and architecture described before can be also used to improve the timeliness of monitoring operations. We consider this topic from the special perspective of the Internet, as time-critical operations are particularly problematic as well as interesting in this scope.

Timeliness depends on end-to-end delay. End-to-end delay over the Internet can be expressed in terms of fixed and variable components [5]. The main fixed components are transmission delay at a node and propagation delay on the link. The main variable components are processing and queuing delay at nodes. The overall delay is lower-bounded by the link latency, but it varies dynamically [5] and it is often unpredictable. Transmission delay increases, roughly, linearly with packet size. Queuing delay increases, roughly, linearly with packet rate, and fluctuates rapidly over small intervals [5]. Consequently, as the overall end-to-end delay fluctuates in a dynamic and, often, unpredictable way - e.g., depending on the traffic injected by the network users - time-critical tasks cannot always be supported by the Internet. For example, a performance monitoring task characterised by a relatively small sampling period might not be able to meet also real-time (RT) constraints unless supported by a suitable RT infrastructure.

The reduction of the management network load obtained with MAs can reduce the overall response time. However, delay over the Internet will still vary dynamically and unpredictably due to competing traffic loads. Thus, the use of MAs on its own cannot guarantee RT responsiveness. Bandwidth reservation protocols, like the Resource ReSerVation Protocol (RSVP) proposed for the Internet [25], can help improving timeliness. Such protocols are meant to provide Quality of Service (QoS) commitments with regard to bandwidth, packet loss, and delay through the reservation of network resources along the data path.

In theory RSVP could be used in order to guarantee timeliness regardless of the paradigm adopted. However, a centralised, polling-based, paradigm would generally require a higher transmission rate than the one usually required by MAs (as noted in the previous sections). This, in turn, means that queuing delays associated with centralised polling are generally longer than the corresponding delays obtained with MAs. Alternatively, we could say that a centralised paradigm would generally require a reservation of a greater amount of resources than an MA paradigm in order to guarantee comparable queuing delays - that is, end-to-end delays - for time critical operations.

Conclusions

A dynamic hierarchical management model based on MAs provides an attractive perspective to the lack of flexibility and scalability of current centralised management systems. We have presented our view of a feasible model and architecture and shown some scenarios highlighting the impact that mobility and migration can have on monitoring. Most of the considerations can be extended to management in general. A key architectural issue we have tried to address concerns the placement of the migration intelligence. We think that migration decisions should be placed as close to the MAs as possible, but outside their boundary.

Acknowledgements
We are grateful to Hewlett-Packard for their encouragement and sponsorship in relation to this work. We would like to thank the researchers at HP Laboratories Bristol (HPLB) for providing interesting and stimulating ideas. Helpful suggestions also came from members of the Department of Computer Science at UCL and the Database Laboratory at Politecnico di Milano.

References
[1] G. Cugola, C. Ghezzi, G.P. Vigna, *Analysing Mobile Code Languages*. C. Tschudin and J. Vitek (Eds.), Mobile Object Systems. LNCS, Springer-Verlag, Berlin, Germany, 1997.

[2] M. Baldi, S. Gai, G.P. Picco. *Exploiting Code Mobility in Decentralized and Flexible Network Management*. First Int. Workshop on Mobile Agents, Berlin, Germany, April 1997.

[3] A. Carzaniga *et al*, *Designing Distributed Applications with a Mobile Code Paradigm*. ICSE'97.

[4] J-P Martin-Flatin, S. Znaty, *Annotated Typology of Distributed Network Management Paradigms*. DSOM'97, Sydney, Australia, 21-23 October, 1997.

[5] J.C. Bolot, *End-to-End Packet Delay and Loss Behaviour in the Internet*. SIGCOMM'93 Ithaca, N.Y., USA, Sept. 1993.

[6] G. Pavlou et alt., *Distributed Intelligent Monitoring and Reporting Facilities*. The British Computer Society, IEE and IOP Publishing. Distrib. Syst. Eng. 3, pagg. 124-135, 1996.

[7] A. Vassila, G. Knight, *Introducing Active Managed Objects for Effective and Autonomous Distributed Management*. IS&N 1995, Heraklion, Greece, October 16-20 1995.

[8] A. Vassila, G. Pavlou, G. Knight, *Active Objects in TMN*. ISINM '97, San Diego, USA, May 12-16 1997.

[9] K. Meyer, et al., *Decentralising Control and Intelligence in Network Management*. INM'95, Santa Barbara, CA, 5/1995.

[10] Y. Yemini, G. G. Goldszmidt, S. Yemini, *Network Management by Delegation*. INM II, Amsterdam 1991.

[11] G. Goldszmidt, Y. Yemini, *Distributed Management by Delegation*. Proceedings of the 15th International Conference on Distributed Computing Systems, June 1995.

[12] G. Goldszmidt, Y. Yemini, *Delegated Agents for Distributed System Management*. IFIP/IEEE DSOM 1996 Workshop. L'Aquila, Italy, October 28-30th 1996

[13] J-CH Gregoire, *Management Using Delegation*. Advanced Information Processing Techniques for LAN and MAN Management. Amsterdam: North Holland, 1993.

[14] J-CH Gregoire, *Models and Support Mechanisms for Distributed Management*. INM IV. New York: Chapman & Hall, 1995.

[15] F. Gagnon, J-CH Gregoire, *Implementation of Delegation in Distributed Network Administration*. CCECE 93.

[16] Maria-Athina Mountzia, Gabi Dreo-Rodosek, *Delegation of Functionality: Aspects and Requirements on Management Architectures*. IFIP/IEEE Workshop on Distributed Systems: Operations & Management, October 1996.

[17] T. Magedanz, *On the impacts of Intelligent Agent Concepts on Future Telecommunication Environments*. IS&N 1995, Heraklion, Crete, Greece, October 16-20 1995.

[18] T. Magedanz, T. Eckardt, *Mobile Software Agents: A new Paradigm for Telecommunications Management*. NOMS 1996.

[19] T. Magedanz et al, *Intelligent Agents: An Emerging Technology for Next Generation Telecommunications*? IEEE INFOCOM, San Francisco, California, USA. March 1996.

[20] T. Magedanz, *Towards "Intelligence on Demand" - On the Impact of Intelligent Agents on IN*. ICIN 96, Bordeaux, France, November 25-28, 1996.

[21] R. Kooijman, *Divide and Conquer in Network Management using Event-driven Network Area Agents*. Technical University of Delft, The Netherlands. May 1995.

[22] M.R. Siegl, G. Trausmuth, *Hierarchical network management: a concept and its prototype in SNMPv2*. Technical University of Wien, Austria. May 1995.

[23] M. Decina, V. Trecordi, *Convergence of Telecommunications and computing on networking models for integrated services and applications*. Special Issue of the Proceedings of the IEEE, 1997.

[24] A. Pras, *Network Management Architectures*. Thesis. University of Twente, The Netherlands. February 1995.

[25] P. White, *RSVP and Integrated Services in the Internet: a Tutorial*. IEEE Communications Magazine, May, 1997.

[26] Y. Yemini, *The OSI Network Management Model*. IEEE Communication Magazine, May 1993, Pagg.20-29.