CHAPTER 5

TWO COMPLEMENTARY POLLING SCHEMES FOR IMPROVING MANAGEMENT SCALABILITY

5.1. INTRODUCTION

The core framework introduced in the previous chapter brings about all the benefits associated with the application of Mobile Agents (MA) in Network & Systems Management (NSM), i.e. dynamically customisable management services, local filtering of management information, etc. However, it suits NSM applications that involve high selectivity values for collected data and relatively small set of devices, concentrated in limited number of management domains. In other words, its scalability is questionable as it represents a 'flat' model, whereby a single MA object is launched from the manager platform and sequentially visits all the managed Network Elements (NE), regardless of the underlying topology (see Figure 5.1). The scalability problem is twofold:

- First, in large networks the *round-trip delay* of the MA greatly increases as the overall travel time depends on the number of hops realised by the MA.
- Second, the *network overhead* imposed by the MA transfers grows exponentially with the network size; the slope of the overhead curve becomes steeper in the case of high selectivity values (see Figure 4.21).

Furthermore, when utilising the flat model approach for the management of a large set of devices, the problem of collected data *inconsistency* arises. In particular, when the administrator requests a 'snapshot' of the managed devices status at a given time, the use of a single multi-hop MA does not represent an appropriate solution. This is due to the non-negligible time intervals between the acquisition of each data sample from every NE. For instance, the data values collected from the first and the last itinerary hosts will refer to distant time instants, affecting the consistency and reliability of extracted statistics.

The 'flat' model has been adopted by the majority of Mobile Agent Platforms (MAP) developed for NSM applications, e.g. [KU97, NIC98, BEL99, PIN99, PUL00b]. All the aforementioned works involve frequent MA transfers, when the collection of management statistics is considered, giving rise to scalability and data consistency concerns for the reasons highlighted above. Hence, these platforms are not appropriate for network monitoring and performance management, which represent the main application areas in this thesis.

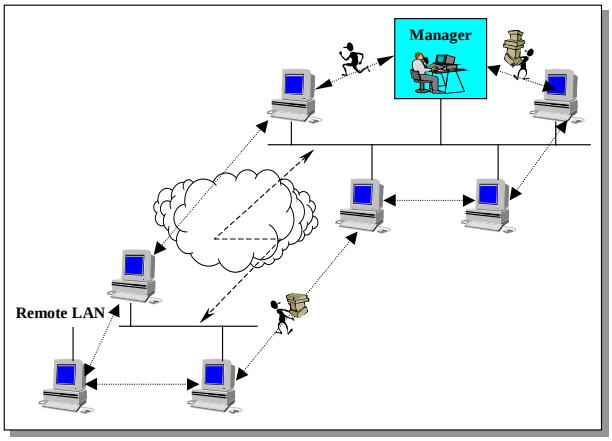


Figure 5.1. 'Flat' MA-based polling

Performance management involves gathering and logging data generated by devices, which may be analysed off-line or in real-time. That process helps in measuring the performance, throughput and availability of network resources. The advantage of analysing the data in real-time is that it allows sophisticated Network Management Systems (NMS) to foresee a possible congestion or failure and take preventive measures before the actual error occurs. On the other hand, collected data may be used to build daily, weekly or monthly graphs/reports to assist the administrator in network planning. In such cases there is no need for real-time NSM data and, hence, an alternative, complementary polling mechanism should be applied.

Therefore, we propose two MA-based polling schemes intended to provide efficient means for obtaining both real-time and off-line management data. The first approach, called *Get 'n' Go* (GnG), is used for the collection of real-time data; the network is partitioned into several domains and a single MA object is assigned to each of them. In every Polling Interval (PI), this MA sequentially visits all NEs within its network domain and obtains the requested information before returning to the manager. The second polling scheme, termed *Go 'n' Stay* (GnS), targets the acquisition of data to be analysed off-line, where the need to obtain data in short time frames is no longer an imperative. Thus, we introduce a method where an MA object is 'multicasted' to all monitored devices; the MA remains there for a number of PIs and collects an equal number of samples. Collected data may be delivered to the manager using various approaches, namely through the broadcasted MA objects themselves, clones of these

objects or RMI invocations. The infrastructure described in Chapter 4 has been extended so as to support the introduced polling schemes.

Two papers describing the design and implementation of the two polling schemes have been published in the proceedings of the *3rd International Workshop on Intelligent Agents for Telecommunication Applications* (IATA'99) and the *IEEE Global Communications Conference* (Globecom'99). Full references are given in Appendix A.

This chapter is organised as follows: Section 5.2 describes the design and implementation details of the two introduced polling schemes. Section 5.3 presents a performance analysis regarding the network overhead associated with their use, with experimental results reported in Section 5.4. Section 5.5 discusses the prons and cons of utilising the two polling schemes in practical management applications and Section 5.6 summarises the chapter.

5.2. POLLING SCHEMES : DESIGN & IMPLEMENTATION

5.2.1. Get 'n' Go Polling Scheme

Recently reported NSM-oriented MAPs assume small or medium-sized networks with flat topology structures. Hence, as the number of managed devices grows, the network becomes increasingly unmanageable. This is a consequence of having a single MA responsible for obtaining NSM data from all devices in every PI, giving rise to serious scalability concerns. In addition, the order in which managed nodes are visited is typically arbitrary.

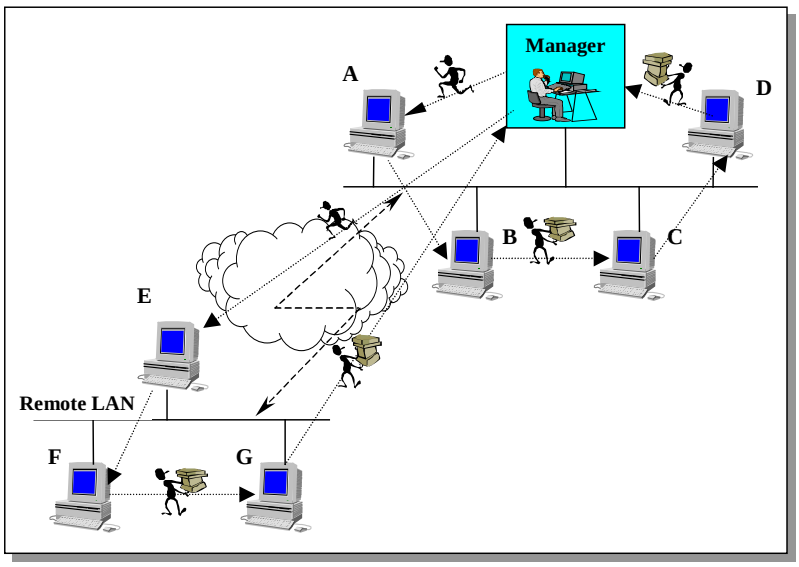


Figure 5.2. The GnG polling scheme

This represents a significant problem when the management of remote LANs is considered, as a travelling MA may have to be transferred several times across expensive and low-bandwidth WAN links during its lifetime. Even worse, when the managed network spans *s*

subnets ($s > 1$) connected to the manager site through low-bandwidth WAN links, the overall response time will increase rapidly, as the multi-hop MA will have to traverse a slow link at least $2 \times s$ times.

As previously mentioned, the concept behind GnG polling is to partition the managed network into several logical/physical domains. For instance, in Figure 5.2, an MA object polls the devices of the remote LAN, whereas a second MA is assigned to the network segment local to the manager host. Hence, flat MA-based management can now be thought of as a special case of GnG polling, where the managed network comprises a single management domain.

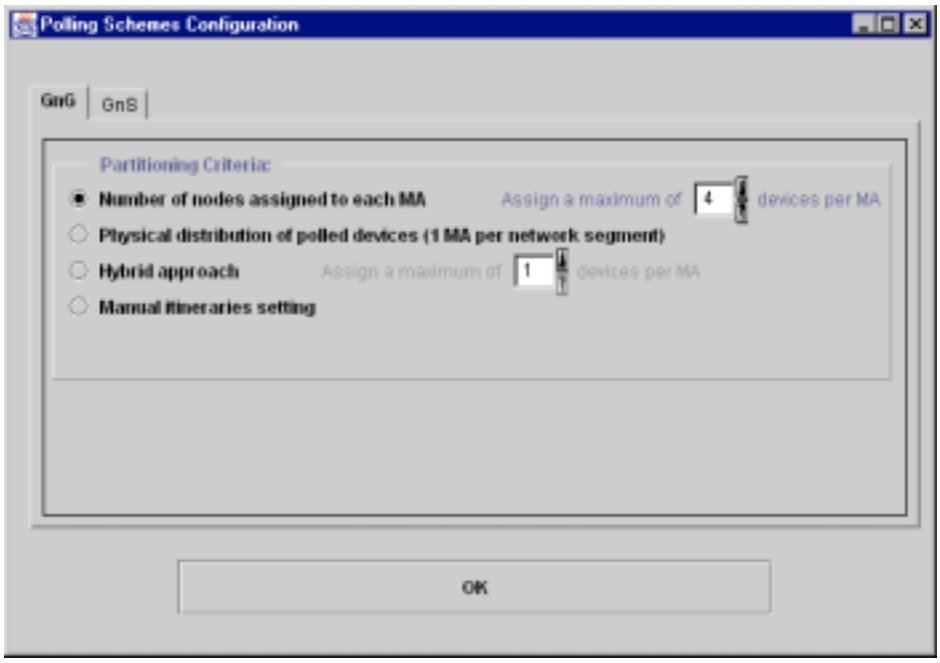


Figure 5.3. GnG polling: Selecting the partitioning criterion

The partitioning criteria are specified by the administrator, during the configuration of the monitoring task parameters. These criteria may be the following:

(a) the number of nodes assigned to each MA, i.e. equal distribution of managed hosts among individual MAs;

(b) the physical distribution of polled devices, i.e. one MA object assigned to each network segment;

(c) a hybrid of these two approaches, i.e. individual MA objects do not visit NEs located on different segments, but there is an upper limit of hosts assigned per MA. In other words, a given network segment may be split in more than one logical domains with every individual domain assigned to a different MA;

(d) manually specified itineraries.

The criterion which is more appropriate for a given management task and its corresponding parameters are selected through the GUI shown in Figure 5.3.

Should criterion (a) is applied, a maximum number of devices assigned per MA is also specified. The number of MAs required per PI is then automatically evaluated and their individual itineraries either manually or automatically specified. In the case that the number of devices is not a multiple of the number of MAs used, the MA object, which is launched last will be assigned a smaller set of devices. This design decision has been made to minimise the overall delay, as the MA object launched first has a marginal time advantage over the one launched last. Hence, should N network devices are divided in d management domains (d MA objects used in every polling interval), some MAs will be assigned to $\left\lceil \frac{N}{d} \right\rceil$ NEs and the remainder to $\left\lfloor \frac{N}{d} \right\rfloor$ NEs[1]. For instance, should 3 MAs are required for the management of 8 NEs, the first two will be assigned 3 NEs and the third only 2. Criterion (b) caters for the case that managed devices are distributed among several subnets, with criterion (c) being more suitable when a large number of NEs is concentrated in specific segments.

With the GnG approach, the launched MA objects travel and perform their management tasks independently, whilst the number of devices they visit is limited, thereby minimising the overall response time. This introduces a high degree of parallelism in the data collection process and suggests GnG as a suitable polling scheme for the acquisition of real-time data.

As the number of devices assigned per MA is reduced, the volume of MAs required to conduct polling is increased and the journey time of each of them decreased. Nevertheless, the manager needs more time and consumes more CPU cycles to instantiate, launch and receive back this increased number of MAs; a side-effect of launching and receiving a higher number of MAs is that the traffic in the manager's network neighbourhood is also increased. The communication overhead is also affected, as short itineraries prevent MAs state from growing too much. However, for large number of domains, the number of MA transfers per PI increases proportionally, which in turn implies higher requirements on network resources. That represents an interesting trade-off, which signifies the need for a detailed investigation in order to identify optimal solutions.

[1] Precisely, $n = N - d\left\lfloor \frac{N}{d} \right\rfloor$ MAs are assigned to visit $\left\lceil \frac{N}{d} \right\rceil$ NEs and $d - n$ MAs to $\left\lfloor \frac{N}{d} \right\rfloor$ NEs.

5.2.1.1. Implementation of GnG Polling Scheme

GnG polling is carried out through *Polling Threads* (PT), implemented by the `Manager.PT` class. PTs are started and controlled by the manager application; each of them corresponds to a single monitoring task. They are created by the MAG tool, when defining the properties of the MA that carries out the task. In particular, the properties of a monitoring application are maintained as a *Polling Thread Configuration* (PTC) object (`Manager.PTC` class). In addition to keeping existing PTCs in memory, they are also serialised and stored in '*configuration files*'. In principle, the serialisation process is very similar to the one used by the MFC components during an MA migration; the two procedures differ only on that in the PTC case the serialised information stream is directed to a file rather than a network connection. Configuration files are updated every time that their corresponding PTC objects are updated and used at the manager application initialisation, in order to restore the PTCs state (through the inverse process of de-serialisation); that way, management tasks are defined only once.

PTs' functionality is determined by their respective PTCs, which comprise a description of the management application and include the polling parameters. Post their creation, PTs remain on 'waiting' mode until activated by the administrator. When activated, PTs define the polling task according to the requirements described in their respective PTC object and start the polling operation.

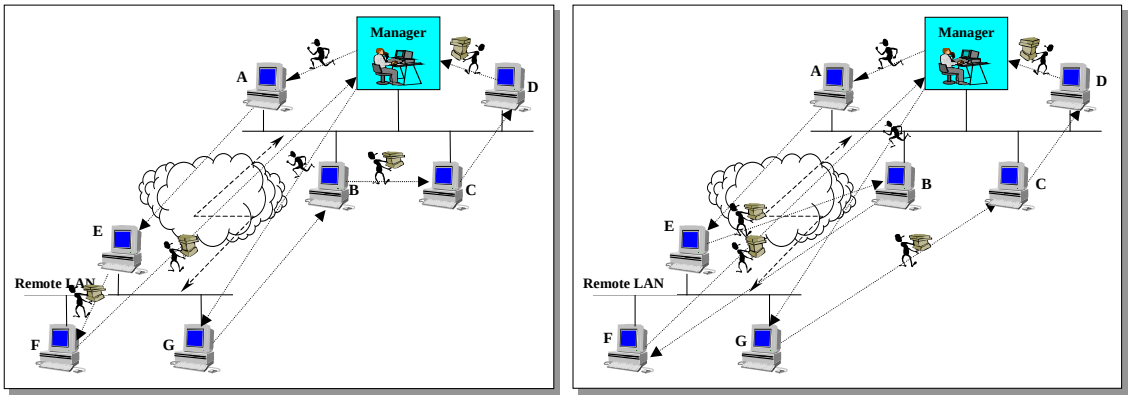
Specifically, when considering the GnG polling scheme, PTs instantiate and launch the required number of MAs (supplied with their corresponding itinerary) and then sleep for one PI; when that period elapses the same process is repeated. Meanwhile, a manager's listener daemon (Mobile Agent Listener thread) receives the MAs that return to the manager carrying their collected data. In a future extension, we will consider PTs synchronisation, so that monitoring tasks sharing the same polling frequency will not be initiated simultaneously. That will ensure that the traffic around the manager host is evenly distributed over time.

Polling properties may be modified at runtime. For instance, the PI may be adjusted, or new NEs may be added in the list of monitored devices. The discovery of a new active agent process triggers an automatic re-evaluation of the required number of MAs per PI and their itineraries; this procedure is transparent to the user.

A graphical table (within the manager's GUI) displays and allows modifications on existing PT properties, such as PT activation/de-activation, polling frequency and the number of devices assigned to each MA. That graphical component will be shown in Chapter 7 (Figure 7.13).

5.2.1.2. Optimal MAs Itinerary Planning

Despite the popularity of MA-based management applications, methodologies for designing efficient MA itineraries have received little attention. In particular, the number of hops realised by a multi-hop agent is not the only metric to evaluate the communication overhead of MA-based operations. The order in which MAs visit the NEs is also a crucial factor; slight changes on the agents' itineraries may result in dramatically variant management costs. The scenario illustrated in Figure 5.2 represents an ideal case in terms of the WAN link utilisation. That is because the link is traversed only twice per PI; the situation could be worse if partitioning criterion (a) was applied. For instance, network partitioning could be performed in such a way that the two MA objects would be required to visit NEs located in both the segment local to the manager station and the remote LAN. This is pictured in Figure 5.4a, with the first MA following the itinerary AEF and the second GBCD. In this case, the WAN link is traversed four times per PI. In a more pessimistic scenario, one of the MAs would not visit the remote LAN hosts in sequence, but traverse the link twice on each direction (itinerary: AEBF), as shown in Figure 5.4b. Apparently, itineraries scheduling lacks a mechanism that would guarantee minimal use of links connecting individual management domains, in other words *optimal itineraries planning* (OIP) is required.



(a) (b)

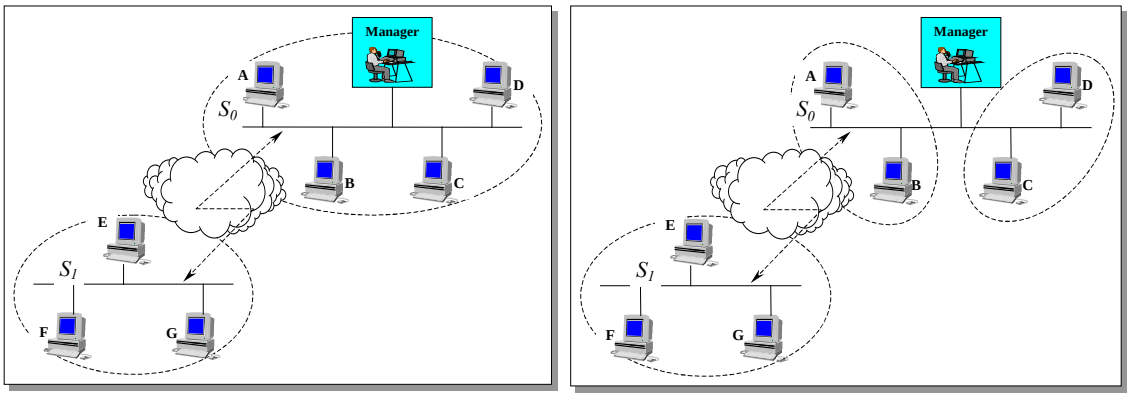
Figure 5.4. Non-optimised partitioning scenarios

Interestingly, the OIP problem exhibits many similarities with the Multi-point Line Topologies or Constrained Minimum Spanning Trees (CMST) problems. A CMST is a Minimum Spanning Tree (MST)[2] with the additional constraint on the size of the subtrees rooted on the centre. CMST algorithms are typically used in graph theory, with the main application area being network design problems. The most well known heuristics efficiently dealing with this problem are *Esau-Williams* (E-W) and *Sharma*'s algorithms [KER93]. These

[2] A MST is defined as a tree (i.e. a connected graph without cycles) with the least total distance, cost, or some other measure of delay or reliability.

algorithms propose near-optimal structures that minimise the total cost when a given network of N terminals is examined. Being heuristics, these algorithms cannot guarantee real optimal solutions; the latter would require the use of complex and time-demanding mathematical methods, such as Integer Linear Programming (ILP) [KER93]. However, for a relatively small number of terminals, the output of heuristics and ILP formulations typically coincide. The input of E-W and Sharma algorithms comprises the number N of terminals to be connected, the name of the central network site, constants $t_{i,j}$ and $c_{i,j}$ $(1 \le i, j \le N)$ that denote the traffic requirement and the cost for connecting terminals i and j respectively, and finally the aggregate cost C_{max} that sets a maximum limit for the cost of each individual multi-point line (subtree).

In principle, an algorithm dealing with the OIP problem would be a variation of a CMST algorithm. In particular, the managed network V would be a union of its s individual segments: $V = \{S_o, S_1, .., S_s\}$. The centre of the managed network (graph) would be the manager station (denoted as terminal *0*), located in segment S_0. The traffic constants would be: $t_{i.j} = 1, \forall i, j$, in order to eliminate their effect on the proposed solution. In addition, the cost constants would be $c_{i.j} = 0$ when $i \in S_k$ and $j \in S_k$ for any subnet S_k, while $c_{i.j} > 0$ when $i \in S_k$ and $j \notin S_k$. Clearly, in the latter case, the cost $c_{i.j}$ will not be a constant value but will reflect the cost of data transfers between two hosts located in separate network segments.



(a) (b)

Figure 5.5. Applying an OIP heuristic to propose (a) two, or (b) three near-optimal itineraries.

Using as case study the simple network topology illustrated in Figure 5.2, the employment of the algorithm briefly described above would prioritise the inclusion of the hosts located on segment S_1 in a single management domain, for both the cases that two or three optimal MA itineraries were requested (see Figure 5.5). In the extreme case that only one itinerary was requested, the application of the OIP algorithm would ensure that the hosts located in S_1 would be visited successively, namely the WAN link would only be traversed twice. Therefore, the algorithm described above would not only provide near-optimal clustering of the managed

network in management domains, but also near-optimal solutions regarding the order in which individual MA objects should visit their assigned NEs.

Should the output of the algorithm not satisfies the requirements set by the administrator (regarding the desired number of NEs assigned to each MA), the C_{max} parameter could be adjusted (decreased), until a solution that incorporates the desired number of itineraries would be obtained. Alternatively, an additional parameter C_{min} could be used in conjunction with C_{max}.

The application of an OIP algorithm presupposes that the manager platform (which runs the algorithm) has detailed knowledge of its managed network topology, i.e. the physical location NEs. A simple managed network topology model is described in Chapter 6. It should be noted that the algorithm dealing with the OIP problem has not been implemented yet, but will certainly be considered among possible optimisations of our framework in future extensions.

5.2.2. Go 'n' Stay Polling Scheme

GnS polling introduces an alternative approach to performance management, targeting data intended for off-line analysis. Within this approach, the mobility feature MA objects is not fully exploited, as the MAs intended to carry out decentralised management tasks are multicasted to the managed devices where they remain until their task is completed. This reduces the need for MA transfers, which are restricted on deploying MAs to remote devices. In addition, the proportion of *useful* management information returned to the manager is substantially increased, since NSM data are not necessarily delivered in every PI.

GnS approach may be useful for gathering performance information from a number of NEs, deliver performance reports on scheduled basis and event-driven notifications when performance thresholds are crossed. The user can analyse the performance reports and notifications obtained to determine utilisation trends, isolate performance problems, and possibly solve them before they cause non-reversible degradation of network performance. In this way performance monitoring can also aid in providing traffic flow predictions (per hour, day or month), long-term capacity and topology planning, identifying bottlenecks and congestion points, monitor quality of service (QoS) parameters, etc.

In principle, GnS polling scheme comprises a direct application of single-hop or constrained mobility (see Section 3.5.5.2). A prototype that implements a constrained mobility platform (CodeShell) has been introduced in [BOH00c] and described in Section 3.6.1.9.

5.2.2.1. Implementation of GnS Polling Scheme

The operation of GnS polling scheme is also controlled by PTs. Its implementation is much simpler than that of GnG approach, as the itinerary of multicasted MAs basically consists of only one host, while MAs are typically deployed only once during the monitoring task's lifetime.

Upon reaching its destination, an MA is instantiated and its execution subsequently started as a separate thread. Having the information of the PI duration embedded into its state, the MA obtains raw performance information from the local legacy system through interacting with the Service Facilitator (SF) component. The obtained data are processed, with the resulting higher-level information being encapsulated. The MA then 'sleeps' for a period defined by the PI duration (the `sleep()` method of the `java.lang.Thread` class is invoked). When this interval elapses, the MA 'wakes up', resumes its execution and obtains another data sample.

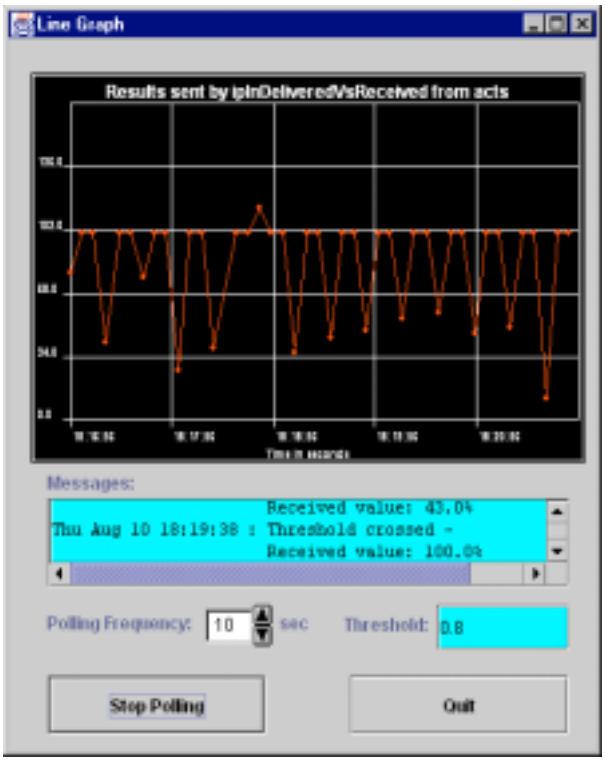


Figure 5.6. Graphical representation of statistics returned by MAs employing GnS polling

Collected samples are delivered to the manager station either in scheduled basis or in case that a monitored performance indicator exceeds a pre-determined threshold. In the latter case, a notification is instantly generated and sent to the manager platform. This functionality resembles the operation of metric objects [ISO93] and the summarisation function [ISO92], as specified in the corresponding standards. The data returned to the manager can be graphically displayed using graphs updated in real-time. This is illustrated in the graph shown in Figure 5.6, which depicts the fluctuation of a simple performance metric that combines two MIB

objects over time (the metric value is evaluated at the managed devices by the MA objects that return the processed data samples).

MA objects either remain on their hosting NEs for a given number of PIs and then request the disposal of their execution thread, or execute in permanent basis (in the latter case the monitoring task will only be terminated at the event of a manager's request). Collected data may be delivered to the manager using three alternative approaches:

(a) through the multicasted MA objects themselves: PTs will repeat the deployment of the MAs to the NEs, given that the execution of the monitoring task is to be continued (see Figure 5.7a);

(b) through clones of the originally deployed MAs (see Figure 5.7b): MA clones are constructed through an explicit invocation of the `clone()` method of the `java.lang.Object` class;

(c) through an RMI call (see Figure 5.7c): the `returnData()` method of the `RMI.ManagerRmiServer` class is invoked.

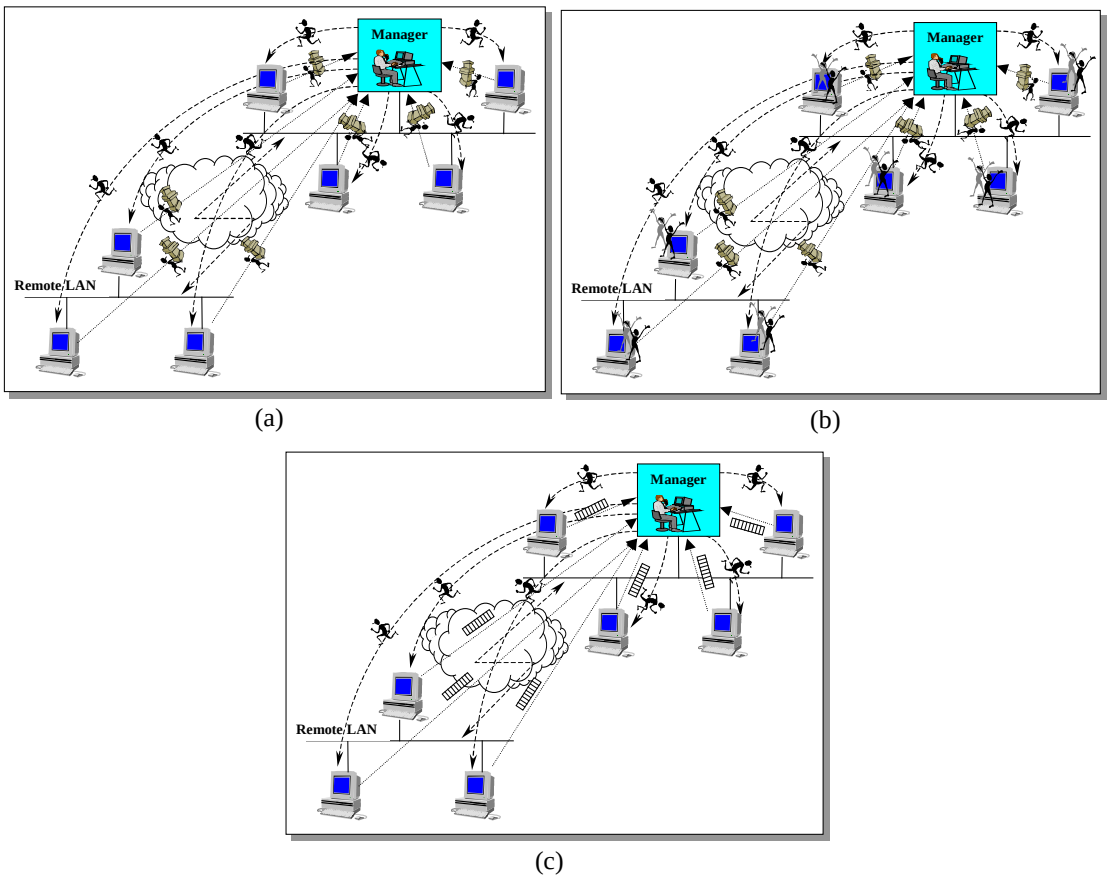


Figure 5.7. Approaches in GnS polling: data delivery through (a) the multicasted MA objects, (b) clones of the multicasted MAs, (c) RMI calls

When applying approach (a), the supervising PT multicasts at regular intervals an MA object to all monitored hosts. The MAs then remain active on the hosts for p PIs (where p is specified by the administrator). When the p PIs elapse, the MAs return to the manager to deliver the acquired samples. Meanwhile, PTs suspend execution for a period given by the product of PI and the number p of PIs that MAs remain on the managed devices ($PI \times p$). When this period expires, they resume operation and the process is repeated. In the extreme case that p=1, the GnS scheme performs similarly to SNMP-based polling and identically to GnG, when each MA is assigned to a single device. This data delivery method certainly implies a higher management cost, as it involves a larger number of MA migrations. However, it has been implemented for evaluation purposes. In approaches (b) and (c), the data returned to the manager either by MA clones or RMI invocations are discarded from the data folder of the MAs that remain on the NEs to continue their monitoring tasks; that way the usage of memory resources by locally executing MAs is moderated.

The properties of GnS polling are specified through the GUI shown in Figure 5.8.

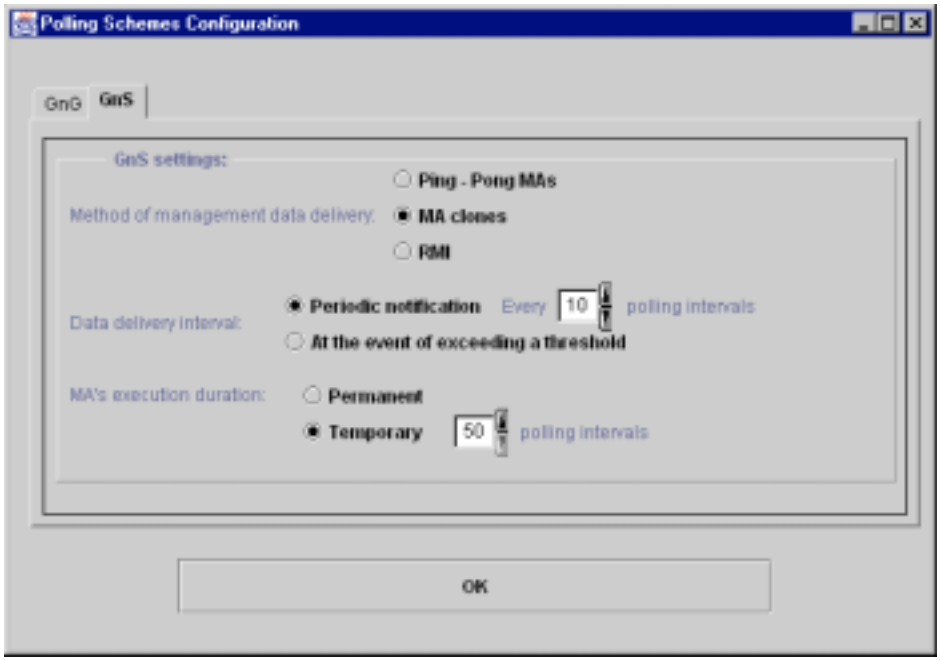


Figure 5.8. Configuring the properties of GnS polling scheme

In the case of updating a monitoring task that employs the GnS scheme, the manager first requests the termination of executing MAs (through RMI calls to their hosting MAS servers) and subsequently deploys their updated version whose execution is immediately started.

It is noted that all the properties required to configure either GnG or GnS polling are declared in `MCode.Monitor` class, which extends the generic `MCode.MA` class. In other words, `Monitor` class defines the functionality of MA objects intended to perform decentralised monitoring tasks. Monitoring MA classes need to extend the `Monitor` class.

The administrator can modify the GnG/GnS parameters in real-time and also to change the polling scheme from GnS to GnG and vice versa, depending on the managed network traffic conditions and the types of management information required, without disrupting the monitoring procedure. The transition from GnS to GnG triggers an automatic network domain segregation process, which satisfies the chosen partitioning criteria.

5.3. PERFORMANCE ANALYSIS

In this section, simple mathematical formulations are devised in order to model the network overhead associated with the employment of the two proposed polling schemes. This performance evaluation essentially extends the evaluation of Section 4.5.2, using the same definitions, symbols and formalisms.

Considering the GnG scheme, we make the assumption that managed devices are equally distributed among individual MAs. Hence, the resulted overhead for GnG polling when segmenting the network into *d* domains would be,

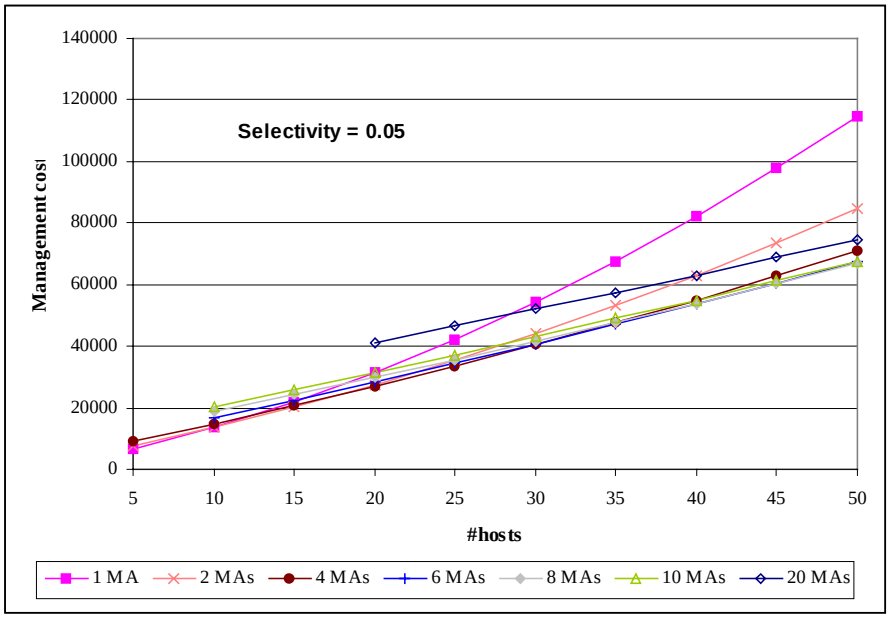
$$B_{GnG} = N * (C + O_T) + d * p * \sum_{h=1}^{h_{tot}} (ST_h + O_T), \text{ where } 1 \le d \le N \text{ and } h_{tot} = \left\lceil \frac{N}{d} \right\rceil + 1 \qquad (5\text{-}1)$$

where h_{tot} represents the number of hops for each MA object and the rest of the symbols hold their usual meaning. The first term of the equation describes the overhead imposed when multicasting the MA code to all MASs, whilst the second represents the bandwidth consumed by the MA state transfers between the manager and the polled devices (each MA is assigned to $\left\lceil \frac{N}{d} \right\rceil$ NEs, hence h_{tot} transitions in total, including the return to the manager). Substituting ST_h from Eqn. (4-7), we obtain:

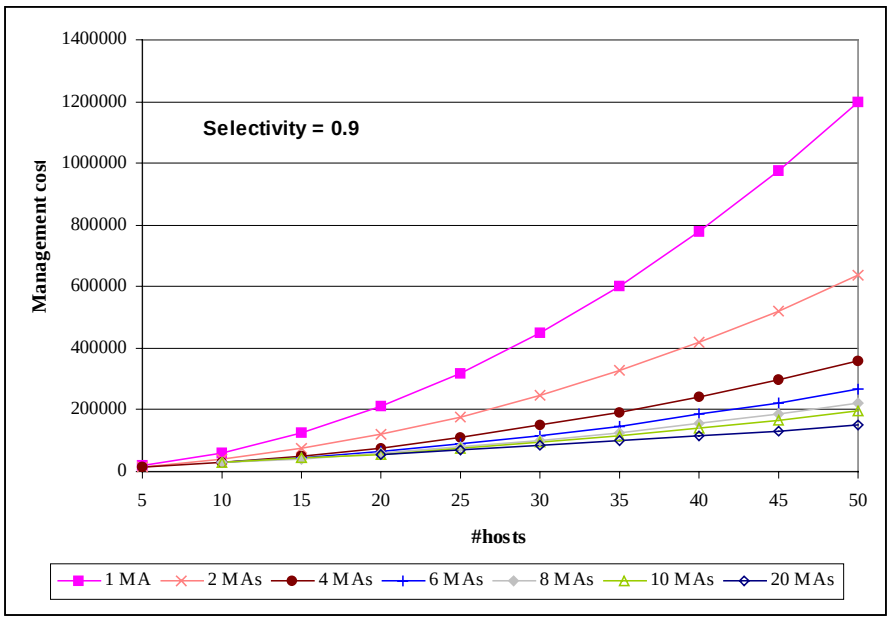
$$B_{GnG} = N * (C + O_T) + d * p * \left[(ST_0 + O_T) * h_{tot} + \sum_{h=1}^{h_{tot}} (dS * h) \right], \; 1 \le d \le N \qquad (5\text{-}2)$$

where *dS* accounts for the state size increment experienced on every MA hop ($dS = \sigma * b$). The conclusions inferred for multi-hop MA-based polling are valid for GnS polling also, although a larger number of MAs (equal to *d*) are transferred on each PI. In particular, the GnG scheme involves $d * h_{tot} \approx N + d$ MA migrations, compared to *N*+1 involved in flat MA-based polling. However, the former is expected to perform better in case that the managed network is fairly large, selectivity values are high or large amounts of data are collected from each host. In that case, the network segmentation will result in shorter itineraries, enforcing travelling MAs to return back to the manager station before their state becomes prohibitively large.

This is graphically illustrated in Figure 5.9, which depicts the dependence of management cost to network size and scalability values. As in Section 4.5.2, we assume an initial MA state size of 400 bytes, 1000 bytes of retrieved data and that *no* data compression is performed. Therefore, for low selectivity values (σ = 0.05, i.e. only the 5% of the originally obtained information is returned to the manager), using a large number of co-operating MAs (e.g. 20) does not represent an appropriate solution, at least for small and average-sized networks (see Figure 5.9a).



(a)



(b)

Figure 5.9. Management cost for GnG polling scheme, as a function of the network size and the number of management domains (i.e. number of MAs used per PI), for selectivity equal to (a) 0.1, or (b) 0.9

The distinction is clear in the case of high selectivity values (see Figure 5.9b) where the separating gap between flat MA-based management (using one MA) and GnG polling is magnified. It is also evident that there is an optimal number of MAs associated with a given network size, which minimises the management cost. For σ = 0.9, that number is 3 MAs for a network of 5 managed hosts, 6 MAs for 10 hosts, 8 MAs for 15 hosts, 10 MAs for 20 hosts, etc. In our current prototype, the decision regarding the number of management domains is made by the administrator, however, in a future extension we intend to provide automated adaptation of that number to the size of the managed network in order to minimise the associated network overhead.

Similarly to GnG, the network overhead for GnS polling scheme would be:

$$B_{GnS} = N * (C + O_T) + N * (ST_o + O_T) + B_{del} \tag{5-3}$$

where the first term of the summation represents the bytecode distribution, the second the actual MAs deployment and the third the bandwidth wasted for data delivery. Only the last term exhibits dependence on the time interval over which the monitoring task is executed. Assuming that data are periodically delivered every p' PIs through the same MA objects that were originally multicasted, B_{del} is given by:

$$B_{del} = N * [2 * (ST_0 + O_T) + dS'] * \left\lceil \frac{p}{p'} \right\rceil \tag{5-4}$$

The MAs return p' data samples, which result in a state size increment of dS'. Following the delivery of management information, a new set of MAs will be multicasted again (hence the *2* multiplier). For a large p', the number of MA transfers is minimised and GnS mode becomes more lightweight in terms of bandwidth consumption. In the case of delivering data through clones of the original MAs, B_{del} becomes:

$$B_{del} = N * (ST_0 + dS' + O_T) * \left\lceil \frac{p}{p'} \right\rceil \tag{5-5}$$

Namely, the deployment of a new set of MAs every time that data are delivered is not further required. Accordingly, data delivery through RMI is modelled as follows:

$$B_{del} = N * Rmi(dS') * \left\lceil \frac{p}{p'} \right\rceil \tag{5-6}$$

where $Rmi(dS')$ denotes the traffic generated by an RMI call that passes as an argument a data vector of size dS'.

A quantitative evaluation of response time for MA-based operations is not attempted here as the adoption of GnG introduces additional complexity which makes its mathematical

modelling very hard, whereas modelling response time for GnS polling would be meaningless since reducing latency is not a prime objective for this scheme.

5.4. EXPERIMENTAL RESULTS

In this section, we describe a number of experiments aiming at evaluating the performance of the introduced polling schemes and comparing it against flat MA-based polling. The physical distribution of managed devices is assumed as arbitrary to these experiments. Similarly to the previous chapter, latency and network overhead measurements are presented.

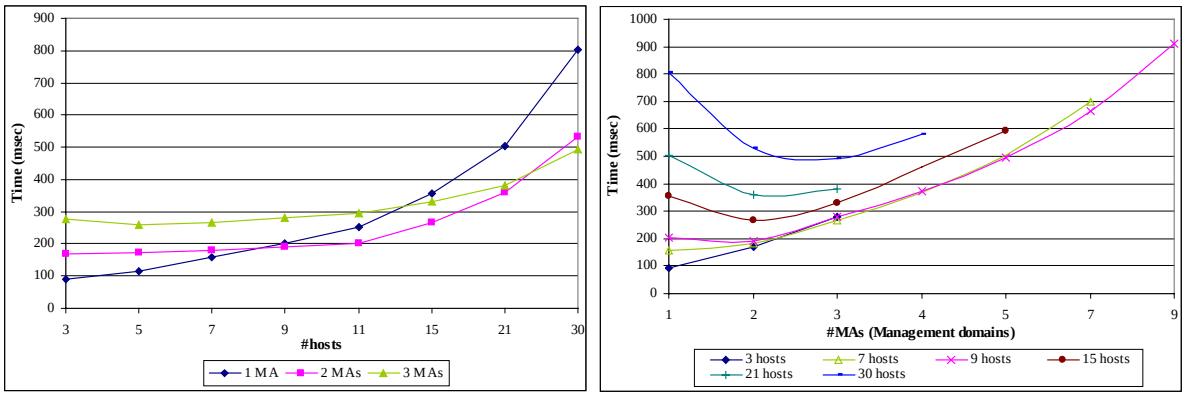
5.4.1. Response Time Measurements

One of the main motivations that led to the design of GnG polling scheme has been to reduce the response time of flat MA-based polling implementations. Hence, the investigation of the factors affecting latency is of great importance. These factors include the network size, the amount of collected data and the number of MAs employed in each PI. TCP is used for MA transfers, while no data compression is applied. Practically, we have repeated the experiment of Section 4.6.1.2, which investigates the performance of multi-hop MAs, i.e. flat MA-based polling. Each travelling MA sequentially visits the polled devices included into its management domain, obtaining a certain amount of information at each point of contact, which can either be 50 (see Figure 5.10) or 2000 bytes (see Figure 5.11), before returning to the manager station to deliver their collected data.

Figure 5.10a shows that the flat approach (using a single MA) does not scale well as the number of NEs increases. This makes it necessary to partition the managed network into several domains in order to maintain low overall response time. In particular, it is shown that in the case that only 50 bytes are collected from each host, using 2 MA objects improves efficiency for networks including more than 13 devices. Likewise, 3 co-operating MAs perform better for networks of 24 managed devices or more. Interestingly, the respective thresholds differ in the case that the amount of collected data equals 2000 bytes. In particular, the transition to two or three co-operating MAs models would improve performance for a number of hosts bigger than 4 and 9 respectively (see Figure 5.11a). This is due to the faster growth of MAs state size, which in turn affects their transfer latency.

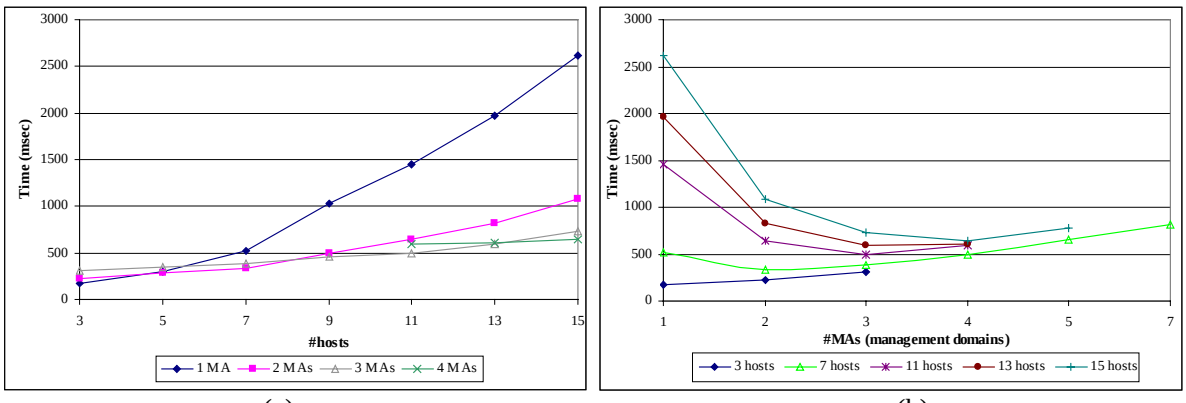
Depending on the managed network size, the optimum number of domains (number of MAs working in parallel) can be determined from the minimum point of the corresponding curves of Figure 5.10b and Figure 5.11b. All curves displayed in these figures are convex, suggesting that the optimal solution always lies between the extremes of segmenting the network in very few or too many management domains. Namely, it is shown that assigning a small number of

devices to each MA, i.e. launching a large number of MAs per PI typically increases the overall response time. That is, although the individual journey times decrease, the time required for the manager to instantiate, launch and receive back these MAs dominates. Thus, for a network of 15 devices, when collecting 50 bytes from each host, response time is minimised when partitioning the network in 2 domains, i.e. assigning 8 NEs to the first MA and 7 to the second. Ideally, the manager application (through the PTs) should dynamically adjust the number of domains according to the current number of managed devices, aiming at maximising efficiency. The conclusions reported in this section agree with the simulation results presented in [RUB00], which investigates strategies for reducing the delay involved in MA-based performance management applications.



(a) (b)

Figure 5.10. Polling response time of GnG polling in the case that 50 bytes are collected from each host, as a function of (a) the network size, (b) the number MAs launched per PI



(a) (b)

Figure 5.11. Polling response time of GnG polling in the case that 2000 bytes are collected from each host, as a function of (a) the network size, (b) the number MAs launched per PI

The information graphically presented in Figure 5.10 and Figure 5.11 is analytically presented in Table 5.1 and Table 5.2 respectively.

# Polled Devices	# MAs launched per Polling Interval													
	1		2		3		4		5		7		9	
	Average	SD	Average	SD	Average	SD	Average	SD	Average	SD	Average	SD	Average	SD
3	91.4	5.9	169.8	11.6	278.7	18.9	-	-	-	-	-	-	-	-
5	114.6	5.4	173.4	19.7	260.4	15.6	-	-	487.8	41.3	-	-	-	-
7	158.8	5.7	180.9	22.7	265.4	19.7	-	-	-	-	698.3	77.8	-	-
9	202.9	13.1	189.1	23.6	281.0	26.2	-	-	-	-	-	-	912.0	81.3
11	250.7	5.4	202.1	19.1	294.3	40.6	-	-	-	-	-	-	-	-
15	355.9	25.4	268.2	22.0	331.3	38.2	-	-	613.5	45.2	-	-	-	-
21	504.5	30.3	358.4	45.1	382.7	45.7	-	-	622.9	47.1	-	-	-	-
30	803.5	64.1	531.4	81.3	493.5	83.4	579.6	124.8	-	-	-	-	-	-

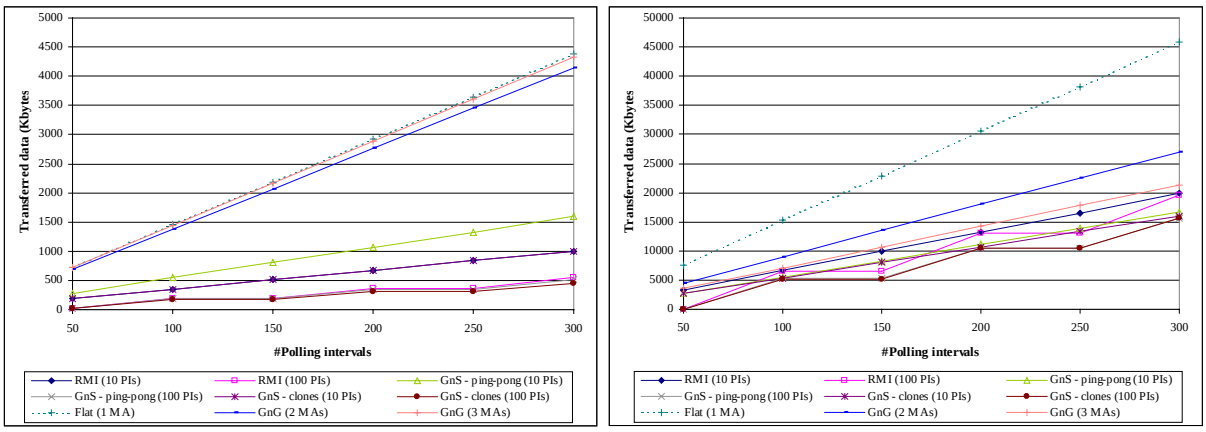
Table 5.1. Average polling time and standard deviation for GnG polling, for multi-hop MAs for various network sizes and numbers of management domains (50 bytes are collected from each host).

# Polled Devices	# MAs launched per Polling Interval													
	1		2		3		4		5		7		9	
	Average	SD	Average	SD	Average	SD	Average	SD	Average	SD	Average	SD	Average	SD
3	176.7	35.7	220.7	14.0	309.9	24.6	-	-	-	-	-	-	-	-
5	300.4	36.3	280.5	37.2	344.1	42.0	-	-	549.4	46.7	-	-	-	-
7	521.0	49.5	335.0	37.3	381.2	29.2	-	-	-	-	816.0	86.7	-	-
9	1030.3	46.8	495.9	26.5	454.8	33.3	-	-	620.2	-	-	-	1129.5	143.6
11	1452.3	36.3	638.5	28.6	496.7	48.1	596.9	49.9	-	-	-	-	-	-
13	1967.9	68.0	821.1	37.4	590.6	61.9	605.9	58.9	-	-	-	-	-	-
15	2611.5	71.8	1084.2	54.5	728.9	80.9	647.5	74.6	779.3	48.0	-	-	-	-

Table 5.2. Average polling time and standard deviation for GnG polling, for multi-hop MAs for various network sizes and numbers of management domains (2000 bytes are collected from each host).

5.4.2. Network Overhead Measurements

This section reports the results of a simple experiment that aims at measuring the network traffic incurred when the management of a small set of devices is involved. The experimental testbed comprises a PC that plays the role of the manager and 10 PCs in which a MAS server is installed, simulating managed devices. We compare the performance of GnG polling (for various network partitioning configurations) against that of GnS (for various data delivery frequencies).



(a) (b)

Figure 5.12. Network traffic incurred by GnG, GnS and RMI approaches as a function of the polling intervals for data samples equal to (a) 50 bytes, and (b) 2000 bytes.

# Polling Intervals	RMI (10 PIs)	RMI (100 PIs)	GnS (ping-pong, 10 PIs)	GnS (ping-pong, 100 PIs)	GnS (clones, 10 PIs)	GnS (clones, 100 PIs)	GnG (1 MA)	GnG (2 MAs)	GnG (3 MAs)
	Transmitted data on MAC layer, Kbytes (data sample: 50 bytes)								
50	183	20	281	20	183	20	729	689	722
100	346	194	543	184	347	165	1459	1379	1443
150	509	194	805	184	511	165	2188	2068	2165
200	672	369	1067	349	675	309	2917	2758	2887
250	836	369	1329	349	838	309	3647	3447	3608
300	999	544	1591	513	1002	454	4376	4137	4330
	Transmitted data on MAC layer, Kbytes (data sample: 2000 bytes)								
50	3326	20	2788	20	2690	20	7640	4502	3558
100	6632	6518	5557	5243	5361	5224	15281	9004	7117
150	9939	6518	8326	5243	8031	5224	22921	13507	10675
200	13245	13017	11094	10467	10702	10428	30562	18009	14233
250	16551	13017	13863	10467	13373	10428	38202	22511	17792
300	19858	19515	16632	15691	16043	15632	45843	27013	21350

Table 5.3. Comparison of GnG, GnS and RMI-based approaches in terms of network overhead: The volume of the transferred data on the MAC layer

In particular, network traffic is measured (by the Windump tool) for the cases that single data samples comprise 50 or 2000 bytes. The performance of GnG polling is investigated when employing 1 (flat management), 2 or 3 MAs. When considering GnS polling, data delivery frequency can either be 10 or 100 PIs, i.e. 10 or 100 data samples respectively are returned to the manager either through an RMI call or encapsulated into an MA state. Focusing to GnS polling, data deliveries are performed either by the originally multicasted MAs, their clones or RMI calls. The results graphically illustrated in Figure 5.12 are also analytically presented in Table 5.3.

As expected, the presented results demonstrate a clear advantage of GnS over GnG polling scheme, especially when the monitoring tasks execute for long time periods. The separating gap is, however, reduced for large data samples (see Figure 5.12b). In both cases, flat MA-based polling exhibits the worse performance amongst all alternative approaches. Regarding

GnS polling, data deliveries through clones of the originally deployed MAs has been shown to represent the most efficient approach. This conclusion agrees with the results presented in Section 4.6.2, according to which our MA framework marginally outperforms RMI in terms of network overhead when considering simple data transfers between a pair of hosts. Furthermore, extra savings on GnS overhead may be achieved by applying more infrequent data deliveries (every 100 instead of 10 PIs).

5.5. DISCUSSION

It should be already evident that the decision regarding the choice of the appropriate polling scheme depends on a number of factors, such as the type of the monitoring task, response time restrictions, the estimated task execution period, the physical distribution of managed devices, etc. The following sections comprise an analytical comparison of the two polling schemes, discussing the advantages and trade-offs associated with their employment.

5.5.1. GnG polling scheme

GnG polling represents an efficient management model in the case that real-time operations are involved, namely when response time restrictions apply. Specifically, it has been shown that for relatively large networks, applying logical network partitioning and letting a set of MAs visiting their assigned nodes in parallel, may significantly improve efficiency in comparison with both flat MA-based model (launching a single MA) or deploying an MA to every device. In the latter case, each MA would typically be executing for a time negligible with respect to its deployment time; hence, the agent deployment overheads would be unacceptable for time sensitive applications [BOH00b].

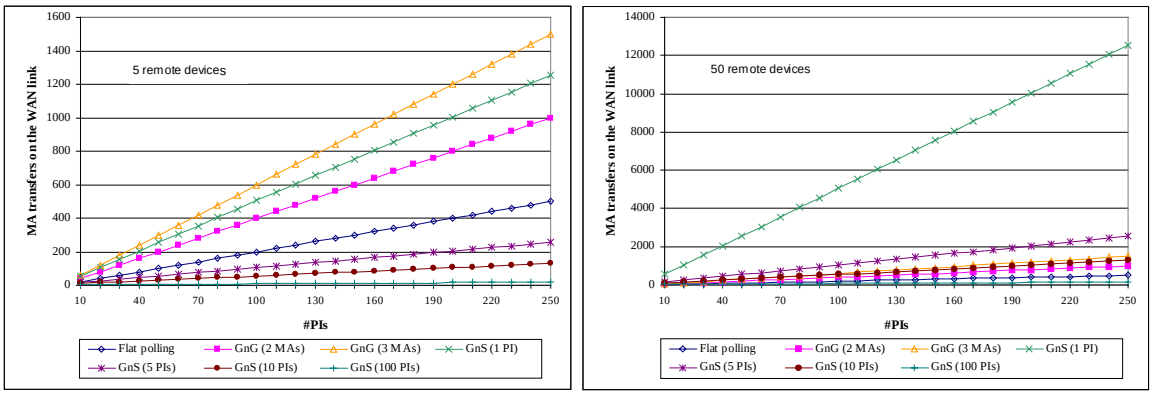
GnG polling also results in network overhead savings compared to the flat model, especially when the amount of data stored within MAs state is substantial. When the monitoring task is intended to run for a short time period, GnG polling may achieve network traffic savings even over GnS mode.

GnG polling also relaxes the management station from processing bottlenecks that would appear should GnS polling be applied for the management of a large number of NEs (in such case, an equal number of MAs would be instantiated and launched, resulting in overloading the manager platform). An invited side-effect of using a relatively small number of MAs is that network capacity around the manager platform is not saturated by the simultaneous generation and transmission of the agents, leading to more even utilisation of network resources. It should be acknowledged though that flat management answers the two aforementioned problems

(manager overloading and management traffic distribution) more adequately, as only one MA object is transmitted in every PI.

Furthermore, modifications of monitoring tasks are easier when employing GnG polling. That is because the corresponding MA classes are updated on the manager station, with the modifications taking effect on the PI following the update. On the other hand, when using the GnS approach, MA updates are carried out through deploying the new version of the modified MA classes; frequent modifications may result in a dramatic increment of network overhead.

Last, GnG polling may also reduce the utilisation of links connecting individual network segments. If, for instance, a topology similar to the one illustrated in Figure 5.2 is considered, the employment of GnG polling would allow the deployment of a single MA object sequentially visiting all the NEs located within the remote subnet, before returning to the manager station. If, on the other hand, GnS polling is employed, both MA deployments and data deliveries will make use of the WAN link. Figure 5.13 depicts the number of MA transfers over the WAN link for various GnG and GnS configurations. Clearly, when a large number of NEs is concentrated in the remote subnet, GnS polling should apply infrequent data deliveries to maintain low link utilisation (see Figure 5.13b).



(a) (b)

Figure 5.13. Number of MA transfers over the interconnecting link required for the management of a remote subnet including (a) 5, or (b) 50 devices

5.5.2. GnS polling scheme

GnS polling can be thought of as a natural evolution of the REV paradigm where the uploaded MA code does not only represent a remote service, but can also act as a fully autonomous software component. It is therefore particularly suited to programming and dynamically extending the capabilities of network devices.

With respect to monitoring applications, GnS polling represents an attractive distributed management model for off-line analysis of bulk management data. In particular, its use is

advantageous over GnG approach in cases that monitoring tasks are intended to run for relatively long time frames. Since it performs MAs deployment only once, it practically diminishes MA transfers through the network, reducing the associated traffic, given that data deliveries are not very frequent. For certain classes of monitoring applications, when the manager should be notified only on emergency situations, GnS scheme may reduce the bandwidth usage even further as the manager is notified only when certain thresholds, referring to performance metrics, are exceeded.

Furthermore, since management operations are performed locally, the polling frequency of monitoring tasks can be increased (i.e. the PI interval decreased) without putting any additional strain on network resources. That would allow observing instantaneous fluctuations of network & systems performance, providing more accurate statistics as the values of performance indicators would not be averaged over long time periods. Increments of polling frequency should however be regulated so as not to result in excessive consumption of system computational resources.

5.6. SUMMARY

This chapter introduced two polling schemes that aim at answering the scalability problems of flat MA-based management. Their implementation is based on the MAP presented in Chapter 4. The two polling schemes complement each other, in the sense that they jointly cover a wide range of monitoring tasks.

In particular, GnG polling is suitable for collecting on-line data and performing simple control and configuration tasks on several NEs. Response time is minimised by employing a number of MAs that travel within their assigned management domain, introducing a high degree of parallelism in the data collection process. This method may also be preferable when considering the management of remote LANs for short time range as it can reduce MA transfers over the interconnecting links. Concerning the off-line analysis of management data, we propose the GnS polling scheme. In this approach, MAs collect a larger amount of data before delivering performance reports to the manager, leading to drastic reduction of MA transfers. Hence, the selection of the appropriate polling scheme (and its associated parameters), is a compromise between network overhead, response time, execution period, etc, depending primarily on the type of management data to be collected. In both cases, semantic compression of NSM data can be applied to further reduce bandwidth usage. The performance analysis and results indicate a significant improvement in both response time and traffic overhead when comparing the introduced polling schemes to flat MA-based polling.

As a final remark, it should be emphasised that despite the improved scalability that GnG and GnS polling schemes introduce over centralised and flat MA-based management, the problem of managing remote subnets is not adequately addressed. Specifically, when physically and geographically dispersed managed network topologies are involved, the application of the two polling schemes implies a rather heavy utilisation of the interconnecting links. The management system has therefore increased dependency on network resources, whilst being prone to network failures. Furthermore, the manager station represents a single point of failure, as it is still responsible for managing the entire managed network, regardless of its size and topology structure. All these issues suggest that there is still space to further improve MA-based management scalability and flexibility. Concrete ideas for achieving such improvements are proposed in Chapter 6.