Chapter 6

VI. EVALUATION

Section VI.1 - Introduction

In this chapter we are going to present and analyse the data obtained from running the previously described scenarios. Based on such data we will elaborate on the causes and reasons for these figures, on the goodness of the different trade-offs assumed during the design and on the effects that the implementation of some of the initial framework requirements has over the performance of the system.

Furthermore, we will spend some sections discussing how will the framework perform under different scenarios. We will justify this expected performance based on the numbers obtained from the previous ones.

Finally, in addition to this exhaustive analysis of the evaluation data obtained, we will also compare the results obtained with previous projects dealing with the management of active networks, in particular with FAIN. We will discuss the different performance figures and, when applicable, explain their causes.

The evaluation chapter is organised in two sections. The first one explains the criteria that have been followed to evaluate the performance of the system as well as what system aspects have been evaluated. The second section presents the obtained evaluation data and the comparison to the available FAIN performance results.

Section VI.2 – Evaluation criteria

The overall goal of this evaluation is to assess the framework proposed, and more specifically, to assess the goodness of the different approaches taken during the design and the technical solutions used in the proof-of-concepts implementation.

In order to realise this assessment we must first specify the criteria that rules such evaluation. In this section we describe these criteria, which is composed by two main sets. The first one encloses those aspects that will help to establish whether the initial functional requirements that were specified for the management framework in our Thesis objectives have been fulfilled and to which extent. This set will be called 'functional criteria'. The second set is focused to analyse the performance of the system when realising the designed functionality. This set of criteria will be called hereafter 'statistical criteria' as

their assessment is based on the analysis of statistical data recompiled when running the scenarios. Often, the results obtained when evaluating the statistical criteria will be used as input for the evaluation of the functional criteria.

In the remaining of this section we will enumerate the different criteria contained within the two sets and define, for each of them, to what we refer exactly and what will be taken into account when evaluating the criteria.

1st Functional Criteria

A Flexibility

With flexibility we refer to the ability of a management system to cope with different managed network needs: different types of managed devices, different services, resources, etc.

In particular, when assessing the flexibility of the system we will pay special attention to:

· The system ability to manage heterogeneous networks that consists of different types of passive, programmable and active routers.

· The system ability to take advantage of the technical capabilities and technologies of the managed devices not only to better manage these devices but also to enhance the management performance (e.g. by reducing management traffic, etc.).

· The possibility of distributing different components of the management system in different machines so that the overall system performance can be enhanced.

· The system capability of creating different management infrastructures so that it can be best adapted to the network operator necessities.

· The system support for policy group processing and types of group processing strategies supported (e.g. atomic, best effort...).

B Extensibility

Extensibility can be seen indeed as part of the system flexibility property, and hence to be included within the flexibility criteria. However, its relevance is so high in a management system oriented to the management of active and programmable networks that it is worth to include it as a separated criterion.

By extensibility we refer to the ability of the system to change dynamically its behaviour, adapt to new services or managed resources and even to cope with modifications in the managed network as additions or removals of network elements in the managed infrastructure.

C Delegation

The delegation criterion is used to assess the delegation functionality included in the framework. By delegation we refer to the ability of the network operator owning the management system to allow some of his customers to realise certain management actions over the resources they have got allocated. The process of assigning these 'access rights' to customers is what we call Delegation.

More specifically, in relation to the delegation ability of the system we will assess the extent to which the management functionality can be delegated, the level of granularity of the delegation capability, the simplicity of the solution adopted and the computational cost associated to it

D Scalability

Scalability refers to the management framework design and the distribution of the management functionality in such a way that the system can cope with increasing number of managed devices and user requests.

In relation to the scalability criterion we will pay a special attention to the scalability of the system in terms of increasing the number of managed devices. The justification for this is extensively discussed in the following section.

E Security

Within the thesis objectives it has not been taken into account security, which is considered as out of the scope of this thesis. However, the framework designed still considers some minimum security aspects that will also be assessed. In the final chapter of the thesis we will elaborate around further security mechanisms that could be used to enhance the overall framework security.

F Interworking

We will assess the interworking criterion by analysing the framework features that allow the interworking with other systems. Particularly, we will assess the simplicity and the level of standardisation offered by these interworking features.

G Portability

By portability we understand the ability of the system to work on different types of machines (e.g., SUN, Intel) and with different Operating Systems (e.g. Solaris, Linux, Windows). This criterion will help us to assess to which extent the implemented code is portable.

2nd Statistical Criteria

A Processing time and CPU

The processing time to which we refer in this criterion is the time spent in every step of the policy processing. We will evaluate the total processing times for a policy under different situations as well as the times needed to realise particular policy functions, such as delegation, validation of a policy against an XML Schema, policy translation, etc.

Processing time measures give us an idea of the CPU consumption and help us to assess the computational load of the framework. Although this parameter depends a lot on the way the system is implemented, it is useful as approximate information and especially as comparison data between the different policy processing steps.

In addition to the previous data, we will also provide information about the MANBoP packages that require more computation in average.

These statistics will be provided for the NL station when working alone, for the NL station when working over EL managers and for the EL stations.

B Memory

Within the memory criterion we consider both the size of heap consumed in the Java Virtual Machine and the disk space used by the management system. Particularly, we will detail the storage space needed by the most parts of the MANBoP Database.

C Bandwidth

We will measure the bandwidth consumed by the management framework under different circumstances and when realising different actions. With management bandwidth we refer to the number of bytes per second transmitted and received by the management stations to manage the network. The bandwidth information obtained will be used also as input for evaluating other functional criteria such as the flexibility or extensibility criteria.

Section VI.3 – Evaluation results

In the next paragraphs we present the main outputs from the analysis of the evaluation data. For obtaining this data, we have run one by one the scenarios described in the previous chapter. The statistical data for the evaluation has been obtained with two programs. The first one, a Profiler plugin [Profiler] for the Eclipse development platform [Eclipse], has been used to obtain the heap size, the CPU statistics, and the delays. The second program used has been the Ethereal traffic analyser [Ethereal] based on the libpcap [tcdump] library (winPcap [winpcap] for Windows machines).

The management stations where evaluation data is obtained are Intel 1.5 GHz Pentium IV computers with 500 MB of RAM.

The evaluation information obtained from running the scenarios has been ordered according to the evaluation criteria enumerated in the previous section. Along the following paragraphs we will present and analyse these data for each of the criteria.

1st Functional Criteria

A Flexibility

The system flexibility was measured mainly when running the first scenario. From this scenario we obtained several significant data. In the following lines we expose and analyse this information in relation to the flexibility criteria.

The first criterion we considered for evaluating the flexibility of the system is to assess the capacity of the framework for managing heterogeneous networks. This capacity was tested by creating a testbed that consisted of two FAIN active routers, one ABLE active router, one CISCO 7200 router and one CISCO 2600 router.

The framework run over all these managed devices and was able to manage them seamlessly. This can be seen with the times needed for managing each of these routers, which vary very little except for the actual enforcement interactions that depend exclusively on the managed devices.

In the table below we show the actual times used for processing a policy over each of the managed nodes in the testbed. This time has been decomposed in time until policy enforcement, and policy enforcement time.

Managed Node	FAIN ANN	CISCO 2600	ABLE
Dynamically installable component involved	ServicePC_0_FAIN	QoSPC_0_CISCO2600	BWMM_0_ABLE
Time for processing the policy	511ms	520ms	631ms
Time for downloading the component	400ms	281ms	361ms
Time for the enforcement	4477ms	3575ms	1382ms
Total time	5388ms	4016ms	2364ms

Table 6 - 1. Times for managing different types of network nodes

The times shown in the table are for the part of the scenario with a management infrastructure composed by only a network-level manager. The times in the element-level stations, which are the ones interacting with the managed devices in this case, for the network-level over element-level management infrastructure would be approximately the same.

During the execution of the scenario the MANBoP system also demonstrated been capable of managing more than one node of different types for correctly

processing a policy. In the step 11 of the scenario, the management system monitors an ABLE router to determine when a policy must be enforced, and finally enforces it over a CISCO 2600 router. The data obtained in the evaluation shows that this process is done again seamlessly: The NL management station spends 391 ms for starting the enforcement since it receives the information that the condition is met.

To assess the second flexibility criterion we observed how the management system interacted with the managed devices and particularly, whether the management system took advantage of the facilities and technologies offered by the managed device.

When running the scenarios the management system interacted with FAIN, ABLE and CISCO routers. In the case of the CISCO routers it used the CLI facility of CISCO routers to manage them through a telnet connection. However, it is for the FAIN and ABLE routers where the system takes more advantage of active network technologies and of the facilities that these routers offer.

When managing FAIN routers the management system installs the Monitoring Meter and Policy Consumer components inside the active node. During the scenario execution, we can observe that two Policy Consumers (i.e. QoSPC and ServicePC) are installed inside the FAIN ANNs. In this way, we significantly reduce management traffic, since these components will just receive the request and return the result to the manager, while all the interactions with the managed device are done inside the FAIN ANN. Also, the manager reduces its computational load since part of its components run inside FAIN ANNs. In the first part of the first scenario (when running with the network-level-only management infrastructure), we observed that the network-level manager loads only in its machine the PCC, QoSPC_0_CISCO2600 and DelegationPC_0_MANBoP components. The QoSPC_0_FAIN and ServicePC_0_FAIN components are loaded in the FAIN ANNs. Hence, the component load of the network-level management station, in terms of dynamically installable components, is reduced from 5 to 3. This would also be true for the element-level managers, although in the scenario this cannot be observed due to the testbed configuration (the element-level managers running in the same machines as the FAIN ANNs).

In the case of the ABLE routers the management system also takes advantage of the facilities that they offer. Although, for the ABLE routers this is not done by installing Monitoring Meters or Policy Consumers inside the ABLE active router (not permitted in these routers), but by creating components that take advantage of the facilities they offer.

In the first scenario, we observe that the monitoring of the ABLE routers is done by a Monitoring Meter component running inside the management station. This Monitoring Meter component creates and sends active packets to the ABLE router in the testbed. These active packets will monitor the throughput in one of the router's interface and only when a certain threshold

is exceeded it warns the Monitoring Meter. In this way, the management station takes again advantage of the facilities of the managed device to reduce its management traffic and even its computational load, since all the polling process is done by the active packet inside the ABLE router and not by the management station.

In the table below we compare the CPU and the management traffic needed to monitor the throughput every four seconds on the managed device using an ABLE active packet against using CLI commands.

	Using CLI commands	*Using ABLE active packet*
CPU[31]	Around 100ms for every polling	Around 300ms just once
Management traffic	1453bytes/sec	Just one active packet

Table 6 - 2. Comparative table between monitoring using ABLE facilities or not

The third criterion for evaluating the flexibility of the system was to assess the ability of loading different components of the management system in different machines. This property, in addition of allowing the distribution of the management functionality between different machines might help to sort out different problems that might appear when managing a network.

The fact that the whole MANBoP management system is implemented in CORBA simplifies a great deal the fulfilment of this criterion.

When running the first scenario we observe that, apart from dynamically installing Monitoring Meters and Policy Consumers on some ANNs (i.e. FAIN ANNs), the management system is capable of installing these components in any kind of machine used with that goal. It needs just to change the corresponding line of the underlying topology configuration file introduced in the bootstrap. In the first part of the first scenario (when running with a network-level-only management infrastructure), we have configured the network-level MANBoP manager to install the Monitoring Meter devoted to the monitoring of the ABLE node on the kubrick.upc.es machine and not in the manager station as would be by default. The reason for this change was that while setting up the testbed we noticed that active packets going to the ABLE node were passing through the kubrick.upc.es machine, but this machine was not forwarding them. We solved this problem by changing the network-level manager configuration file for installing the monitoring meter component generating the active packets at kubrick.upc.es. In this way we solved the problem and showed once more the flexibility of the designed framework. In Figure 6 - 1 and Figure 6 - 2 we can see the distribution of Monitoring Meters and Policy Consumers used in the scenario among the different machines of the testbed.

This capability of the management system can be used also to reduce the load of a network-level manager working alone without the need to change the

[31] The CPU is measured as the number of milliseconds the process is using it.

infrastructure to a more distributed one (e.g. with element level managers). In this case, the network-level station reduces the number of components loaded in the station and to a great extent also the management traffic and the computational load. However, with this solution all policies must still be evaluated at the network-level station, hence in cases of high load a more distributed management infrastructure (where the evaluation of policies is distributed between different management stations) will still be advisable and enhance the overall management system performance. Nevertheless, this might be a very good intermediary solution for a network operator.

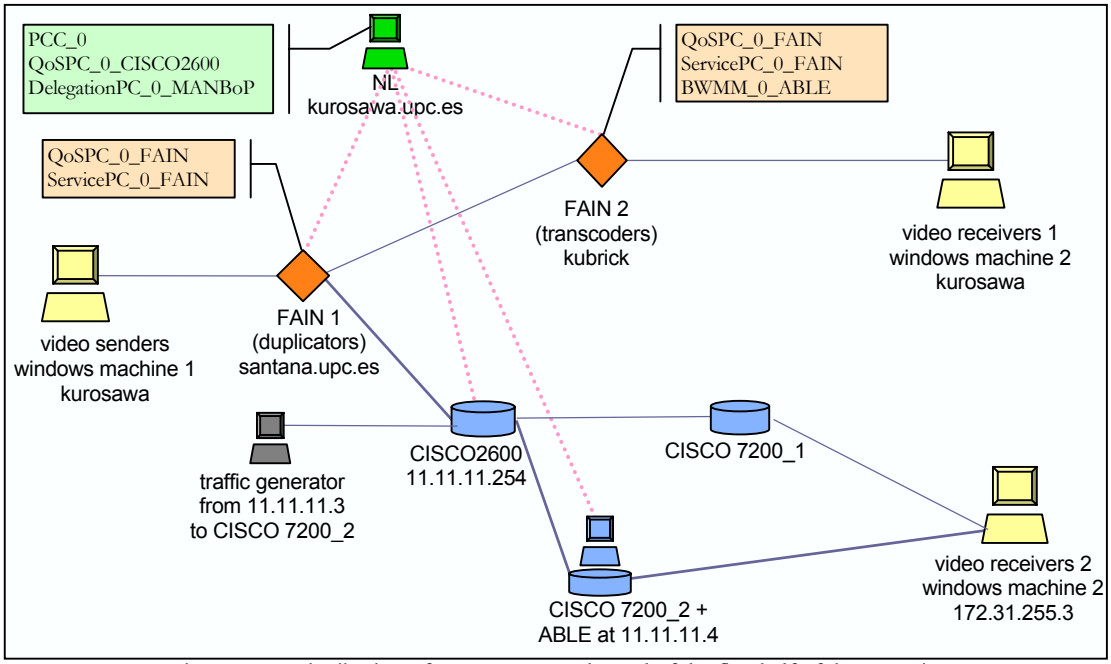

Figure 6 - 1. Distribution of components at the end of the first half of the scenario

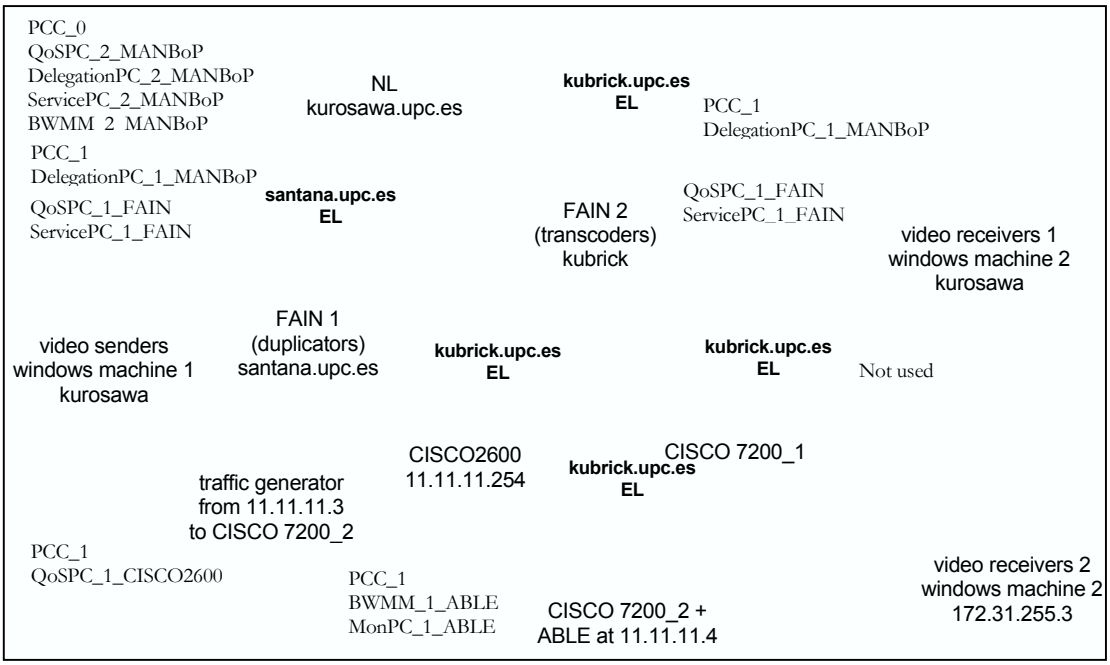

Figure 6 - 2. Distribution of components at the end of the first scenario

The fourth criterion for evaluating the flexibility of the system is assessing the ability of the system for creating different management infrastructures based on the network operator needs (i.e. network topology, number of users, etc.). This was one of the initial requirements for the framework design and is one of the main goals of the first scenario. In this scenario, we can see how exactly the same network and the same service can be managed either with a network-level-only management infrastructure or with a network-level over element-level management infrastructure. Furthermore, the creation of the management infrastructure is a very simple process since the basic code for every management station is the same: the MANBoP framework.

When running the scenario we first created a network-level only management infrastructure by instantiating the MANBoP framework with two configuration files including respectively the managed topology, and the underlying devices. The underlying devices file contains information needed to download the Monitoring Meter and Policy Consumer components appropriate for these devices and where they should be placed.

In the second half of the first scenario we created a network-level over element-level management infrastructure by instantiating one element-level MANBoP instance per managed device with the appropriate configuration files and a network-level MANBoP instance. The differences of the configuration files for the network-level MANBoP instance in relation with the network-level-only management infrastructure reside on the fact that the underlying devices configuration file (see appendix B) now contains

information about the element-level MANBoP managers. Many other management infrastructures can be created depending on the network operator needs. However, the big advantage of this framework property is that all these infrastructure can be created by the network infrastructure by instantiating always the same code (i.e. MANBoP framework) just with different parameters depending on the position within the management infrastructure where it is going to run. Hence, the network operator has a whole range of solutions available to manage its network using a single piece of code.

Finally, the last criterion that must be analysed to assess the flexibility of the system is the support of policy group processing functionality. This functionality is tested in the first scenario by introducing two groups of four policies each. The first group is oriented to the creation of a Virtual Active Network (VAN) to the service provider, while the second one is introduced by the Service Provider to deploy and configure the webTV service (i.e. duplicator and transcoder) in its VAN. The two groups are processed as atomic groups; that is, either all policies of the group are enforced correctly or none is. Other group processing strategies are also supported, such as: best effort, sequential enforcement, etc.

When running the first scenario we observe that policy groups are processed correctly. Group policies were stored until the correct enforcement confirmation of the previous policy of the group arrived. If a not-successful result was received the removal of those group policies that were previously enforced was requested. Finally, the successful enforcement message only appeared when all group policies were enforced correctly.

The implications in terms of processing times of the policy group processing functionality are small. The times spent in policy group processing tasks for a policy, as measured in the scenario, are shown in the table below. We include also in the table the total policy-processing time and the time percentage consumed in group processing tasks over the total time. These times are obtained in the network-level management station for the two management infrastructures demonstrated. They are obtained at the network-level station only because it is there where the policy groups were processed in this scenario. However, other groups might be processed at the element-management level as well depending on how they are defined.

	Policy	Group processing delay	Total policy-processing time	Percentage over the total policy processing time
Network-level-only management infrastructure	1st VAN policy	160ms	6179ms	2,58%
	2nd VAN policy	90ms	1923ms	4,68%
	3rd VAN policy	110ms	611ms	18,00%
	4th VAN policy	90ms	18737ms	0,48%
	1st Service policy	90ms	5388ms	1,67%
	2nd Service policy	110ms	7090ms	1,55%
	3rd Service policy	70ms	5818ms	1,20%
	4th Service policy	70ms	10615ms	0,66%
	Average	98,75ms	7045ms	3,85%
Network-level over element-level management infrastructure	1st VAN policy	90ms	1782ms	5,05%
	2nd VAN policy	100ms	1612ms	6,2%
	3rd VAN policy	101ms	812ms	12,44%
	4th VAN policy	150ms	671ms	22,35%
	1st Service policy	60ms	581ms	10,33%
	2nd Service policy	80ms	431ms	18,56%
	3rd Service policy	70ms	530ms	13,21%
	4th Service policy	80ms	421ms	19,00%
	Average	91,37ms	855ms	10,68%

Table 6 - 3. Policy group processing statistics

The assessment of the MANBoP system flexibility has drawn satisfactory results and presents the framework as a useful tool for network operators to manage heterogeneous networks in terms of flexibility. The framework permits the management of active, programmable and passive routers without, for this reason, loosing any of the advantages that can be obtained when using the capabilities offered by these technologies. Furthermore, the flexibility of the MANBoP framework has appeared as an interesting capability for solving many possible problems that might appear when managing a network.

In relation to the creation of different management infrastructures based on the MANBoP framework is quite a simple process since it is only necessary to start the same piece of code in the different stations with the appropriate configuration files. The remaining processes for achieving the entire functionality are automatic.

As will be described later on this chapter, the scalability scenario shows how the possibility of easily create different management infrastructures can be very helpful for solving different problems, particularly scalability problems, in a cost-effective way.

In what refers to the policy group processing functionality, the performance penalty paid for supporting this functionality is small when compared with the flexibility that this mechanism adds to the management framework. The added flexibility derived from including policy group processing functionality is huge since it allows even a higher automatism of management tasks. Without policy group processing functionality it would be the user of the

management system (network operator or service provider), the responsible of introducing policies one by one and removing them if an interrelated policy was not enforced successfully.

B Extensibility

The extensibility properties of the system are used almost on every single step of every scenario. Nonetheless, it has been during the first scenario where we have analysed extensibility-related data in order to assess this property.

In the following paragraphs we expose the main outputs from the extensibility assessment for each of the two extensibility criteria defined.

In relation to the ability of adding new management functionality criterion and in what refers to the functional assessment, we have observed that the system is capable of detecting when a new component is needed (even XML Schemas), what kind of component is it and where it must be installed. With this information, it simply contacts the CIA component, which does the work. This extensibility property is essential for the management of active and programmable networks, since it allows the management system to adapt to new services or even hardware resources that might have been added dynamically on the nodes. Furthermore, the extensibility property is also very useful for the management of passive nodes because it allows to update the management functionality with a newer version and to have loaded in the management stations only those functional domains that are really used.

During the first scenario we observed that as policies were being introduced the components needed for their processing were downloaded: PCC, QoSPC, DelegationPC, ServicePC, BWMM, etc. Furthermore, these components where downloaded taking into account: the management level at which the MANBoP instance was working and the underlying devices they should work with.

The performance penalty paid for having this extensibility method instead of fix management functionality is not high. The main implication is the time needed for downloading a component. However, this time is only spent once (the first time a policy requires that component to be processed) and each time the component is re-downloaded because it was previously removed from the system for any reason. When a component is used frequently the implication of the download time on the overall behaviour of the system is negligible.

In the table below we can see the implication of downloading the needed component over the total policy processing time as well as its null influence when a second policy arrives to be processed. This information has been obtained from the first half of the first scenario (when having a network-level-only management infrastructure). Nevertheless, for whatever management infrastructure we have, these numbers are approximately the same in every management station.

	Policy	*Downloaded component*	*Download time*	*Total policy-processing time*	*Percentage over the total processing time*
Network-level-only management infrastructure Downloaded to NL manager at kurosawa.upc.es (co-located with the code server)[32]	1st VAN policy	PCC_0	181ms	6179ms	2,92%
		QoSPC_0_FAIN	881ms		14,26%
	2nd VAN policy	DelegationPC_0_FAIN	400ms	1923ms	20,80%
	3rd VAN policy	Already downloaded	0	611ms	0%
	4th VAN policy	Already downloaded	0	18737ms	0%
	1st Service policy	ServicePC_0_FAIN	400ms	5388ms	7,42%
	2nd Service policy	Already downloaded	0	7090ms	0%
	3rd Service policy	Already downloaded	0	5818ms	0%
	4th Service policy	Already downloaded	0	10615ms	0%
	CISCO policy	BWMM_0_ABLE	310ms	4647ms	6,67%
		QoSPC_0_CISCO2600	281ms		6,04%
Average			409ms	6778ms	9,68% (5,28%)[33]
Network-level over element-level Downloaded to NL manager at kurosawa.upc.es (co-located with the code server)	1st VAN policy	PCC_0	90ms	1782ms	5,05%
		QoSPC_2_MANBoP	786ms		44,10%
	2nd VAN policy	DelegationPC_2_MANBoP	401ms	1612ms	24,86%
	3rd VAN policy	Already downloaded	0	812ms	0%
	4th VAN policy	Already downloaded	0	671ms	0%
	1st Service policy	ServicePC_2_MANBoP	150ms	581ms	25,82%
	2nd Service policy	Already downloaded	0	431ms	0%
	3rd Service policy	Already downloaded	0	530ms	0%
	4th Service policy	Already downloaded	0	421ms	0%
	CISCO policy	BWMM_2_MANBoP	291ms	2274ms	12,80%
		Already downloaded	0		0%
Average			343ms	1012ms	22,52% (10,24%)
Network-level over element-level Downloaded at EL manager at santana.upc.es	1st VAN policy	PCC_1	251ms	5465ms	4,59%
		QoSPC_1_MANBoP	286ms		5,26%
	2nd VAN policy	DelegationPC_1_MANBoP	140ms	383ms	36,55%
	3rd VAN policy	Already downloaded	0	425ms	0%
	4th VAN policy	Already downloaded	0	5789ms	0%
	1st Service policy	ServicePC_1_MANBoP	231ms	4893ms	4,72%
	3rd Service policy	Already downloaded	0	5501ms	0%
Average			226ms	3742ms	12,78% (7,3%)

Table 6 - 4. Component downloading time statistics

The second criterion for assessing the extensibility of the MANBoP framework is the evaluation of the ability of the management system for supporting the addition or removal of devices to the managed topology. This one is an important criterion for every management system, since it is expectable that the managed network changes over time, but particularly interesting for the management of heterogeneous networks composed of

[32] The QoSPC_0_FAIN and ServicePC_0_FAIN component are downloaded at the FAIN ANN running in santana.upc.es

[33] The average percentage between brackets considers also the cases when no components are downloaded.

active and programmable routers and passive routers. The cause is the likely progressive deployment of this type of routers into the network.

During the first scenario, in the step 9, we try to introduce a policy involving a managed node that is not included within the managed topology of the system. The Policy Conflict Check (PCC) component refuses the enforcement of that policy. Then, before introducing that policy again we add the involved node in the managed topology using an 'administrative[34]' command supported by the Policy Consumer Manager (PCM) component. This command includes the new node within the managed topology and permits the enforcement of the previous policy.

When having a management infrastructure in various levels (network, element, etc.) the add-node administrative command must be done at the network-level management station that will include it within its managed topology and also on those element-level management stations of neighbouring nodes.

The performance implication of the node addition is really small, especially if we take into account that it will be used only sporadically. The time spent for the node addition in the scenario is 191ms at the first half (network-level only infrastructure) and 221ms at the second half (network-level over element-level infrastructure).

Summarising the extensibility evaluation, we can state that it fulfils the requirements introduced at design time. Moreover, the performance implications of the extensibility mechanisms are not significant on the overall system behaviour while the gains obtained from it are enormous, particularly taking into account that extensibility support is a must on a management system for active and programmable networks.

C Delegation

The assessment of the delegation capabilities of the system is based on the results obtained from running the first scenario. In this scenario the service deployment and configuration functionality are delegated to the WebTV service provider with several restrictions.

One criterion for evaluating the delegation capability of the system is the analysis of the extent to which the management functionality can be delegated. In the way the delegation capability has been designed and implemented in MANBoP, creating restricted XML Schemas for users (see Chapter 4 for more information), the functionality that can be delegated for users (i.e. service providers) is: any management functionality. To delegate the entire management functionality the network operator will simply need to grant access to the service provider to all functional domains; that is to all

[34] With administrative we refer in this case to a command oriented exclusively to the administrator of the system.

XML Schemas without any restriction. Furthermore, the network operator can even delegate to the service provider the functionality to allow the service provider delegating part of its functionality to its customers, although this scenario is not foreseeable.

The delegation of the full management functionality is certainly not advisable. However, in some particular cases the network operator might desire to grant full access to certain functional domains to a particular service provider.

Another important aspect for evaluating the delegation capability of the system is the granularity of the delegation. That is, what is the minimum unit of functionality that can be delegated? In MANBoP, the minimum unit that can, potentially, be delegated is a single instance of a policy. That is, a policy with only one possible value in each of its fields.

The third criterion to evaluate the goodness of the delegation solution proposed is to analyse the complexity of the solution adopted. The framework designed bases the delegation capability in the XML validation mechanism of a policy against its corresponding XML Schemas. We have taken advantage of the fact that there are many implementations of XML validators freely available that realise this task quite efficiently. In MANBoP we have chosen the SUN Multi-Schema Validator [SunMSV] to realise this task.

By leaving policy validation to a specialised code, we first assign the most cost-expensive task, and the one executed more often, to a code implemented with the target of completing the task in an efficient way, and second, we ease enormously the implementation of the delegation functionality of the system. This is due to the fact that the only part that needed to be implemented was that responsible of the creation of the restricted XML Schemas according to the delegation policies received. The implementation of this part is very simple, since it is limited to add or modify some strings on the original XML Schema for that functional domain, while the functional cost associated to this part is not really relevant since it is done only once when the delegation is realised.

In the tables below we summarised the statistical data recompiled during the execution of the first scenario containing information about the computational costs associated to the delegation mechanism. The first table includes the validation time for all policies in the scenario and its influence over the total policy processing time. The second table includes the time spent for delegating the functionality to the webTV service provider in the scenario.

	Policy	Validation time	Total policy-processing time	Percentage over the total processing time
Network-level-only management infrastructure	1st VAN policy	140ms	6179ms	2,27%
	2nd VAN policy	211ms	1923ms	10,97%
	3rd VAN policy	130ms	611ms	21,27%
	4th VAN policy	120ms	18737ms	0,65%
	1st Service policy	140ms	5388ms	2,60%
	2nd Service policy	91ms	7090ms	1,28%
	3rd Service policy	80ms	5818ms	1,37%
	4th Service policy	80ms	10615ms	0,75%
	CISCO policy	180ms	4647ms	3,87%
Average		130ms	6778ms	5%
Network-level over element-level	1st VAN policy	140ms	1782ms	7,86%
	2nd VAN policy	241ms	1612ms	14,95%
	3rd VAN policy	150ms	812ms	18,47%
	4th VAN policy	171ms	671ms	25,48%
	1st Service policy	110ms	581ms	18,93%
	2nd Service policy	100ms	431ms	23,20%
	3rd Service policy	70ms	530ms	13,21%
	4th Service policy	90ms	421ms	21,38%
	CISCO policy	70ms	2274ms	3,08%
Average		126ms	1012ms	16,28%
Element-level	1st VAN policy	271ms	5465ms	4,96%
	2nd VAN policy	84ms	383ms	21,93%
	3rd VAN policy	152ms	425ms	35,76%
	4th VAN policy	112ms	5789ms	1,93%
	1st Service policy	110ms	4893ms	2,25%
	3rd Service policy	70ms	5501ms	1,27%
Average			3742ms	11,35%

Table 6 - 5. Policy validation time statistics

Delegation task	Time at NL (NL-only infrastructure)	Time at NL (NL over EL infrastructure)	Time at EL
Create a new user	201ms	401ms	47ms
Assign access rights (create a restricted XML Schema) to a user	150ms	361ms	152ms

Table 6 - 6. Time statistics for delegation tasks

Concluding, the delegation mechanism designed and implemented in MANBoP appears as a good alternative for delegation of functionality in a policy-based management system using XML as language for expressing policies. Its main advantages are the wide syntactical and semantical flexibility that it offers, its simplicity and the fact that the most cost-expensive task is done by specialised code.

D Scalability

a Introduction

The scalability of a system is one of the most important evaluation criteria. Its goal is measuring the limits of the system and seeing how it behaves near those limits. With that concept in mind we have targeted the evaluation of the MANBoP framework scalability.

There are several situations that can lead a management system to a scalability problem; among them, the more relevant ones are increasing the size of the managed network and increasing the number of system's users. In both cases this leads to more computations, more management traffic, more memory used and bigger delays.

In the two scalability tests conducted we have opted to evaluate only the first one: increasing size of the managed network. There are two justifications for this decision. First, it certainly represents a much more realistic situation for the MANBoP framework since the users of the management system are the network operators and some service providers. This draws a scenario with a maximum number of users on the scale of tenths. Such scenario would not create scalability problems in the framework, especially taking into account that requests on the management plane are usually less frequent than on the control or data plane [ITU91]. Second, even if we decided to test the scalability of the system in terms of users by flooding the framework with policies, such a test would be more an evaluation of the system's implementation than of the framework's design. Nevertheless, at the end of the scalability evaluation section we elaborate in more detail around the scalability in terms of number of users.

b Scalability in terms of managed network size

The assessment of the system scalability is developed in the second scenario, which is targeted to this goal. In this scenario we progressively increase the number of different types of managed devices monitored. The limitation in terms of managed devices comes from the number of components that are downloaded into the management station and the computational resources, as well as management traffic, that these components consume. In the scenario, we have figured out the worst possible situation. That is, every new managed device being monitored is a different type of device than the previous one and hence, needs a different component to process the policy and monitor it. Furthermore, all monitored devices are passive routers so Monitoring Meter components cannot run at the managed device (as would have been the case with active or programmable routers), and should be loaded at the network-level management station.

The initial goal was that each Monitoring Meter (MM) component for each new managed device added in the scenario would realise a real monitoring of

the throughput in a CISCO router's interface, getting the throughput value every five seconds. The problem we faced, though, is that the CISCO router supported only a maximum of 30 connections. Therefore, with the data obtained from this first test, in terms of time spent by the MM in the pooling and management traffic generated, we created a MM component that will have the same computational load (i.e. CPU and memory) as the MMs doing the CISCO monitoring. It has been with the latter ones that we have completed the scalability test although we present in this section the data obtained in both cases.

The data shown in the tables below to evaluate the load of the network-level management station are three processing times (detailed afterwards), the size of the JVM's used heap and the management traffic generated. The management traffic could be measured, for obvious reasons, only on the telnet test. However, it is easily deducible the time for the 100 components, since each component generates management traffic for 1453bytes/sec. Therefore, 100 components would generate, if they could keep the polling rate, a management traffic with a throughput of 145Kbytes per second. The three processing times (or delay times) are the time for downloading the Monitoring Meter component, the time for completing a polling cycle and the time for enforcing a policy on a CISCO router. This last processing time is measured for each ten components loaded. The reason for taking this extra time measure is that the previous two processing times have a great variance over time because their use of the CPU is relatively small. Hence, even when the station is quite loaded, if the MMs happen to develop all their actions in the period of time during which they have the CPU assigned to them, they can carry them out quite fast. On the other hand, if they do not conclude their task and need to wait to have the CPU assigned to them again they spend much more time for concluding their tasks. For this reason, we also provide the five-sample average time for the first two processing times.

The policy enforced on a CISCO router is exactly the same as the one used in the first evaluation scenario. Its processing time is a much more time-consuming task. Therefore, the task cannot be completed in one turn of CPU when the machine is loaded and the behaviour is much more stable. The fact that it spends around 16 seconds to process the policy has to do with the time needed to interact with ABLE and with the CISCO routers, where after every command the thread must sleep for around half a second before sending a new command. We have also modified the code that executes the monitoring for this CISCO policy so that it always informs that the condition is met to avoid that the monitoring influences on the total enforcement time.

The figure below includes the absolute and average download times for the 30 components that could be loaded in the telnet test.

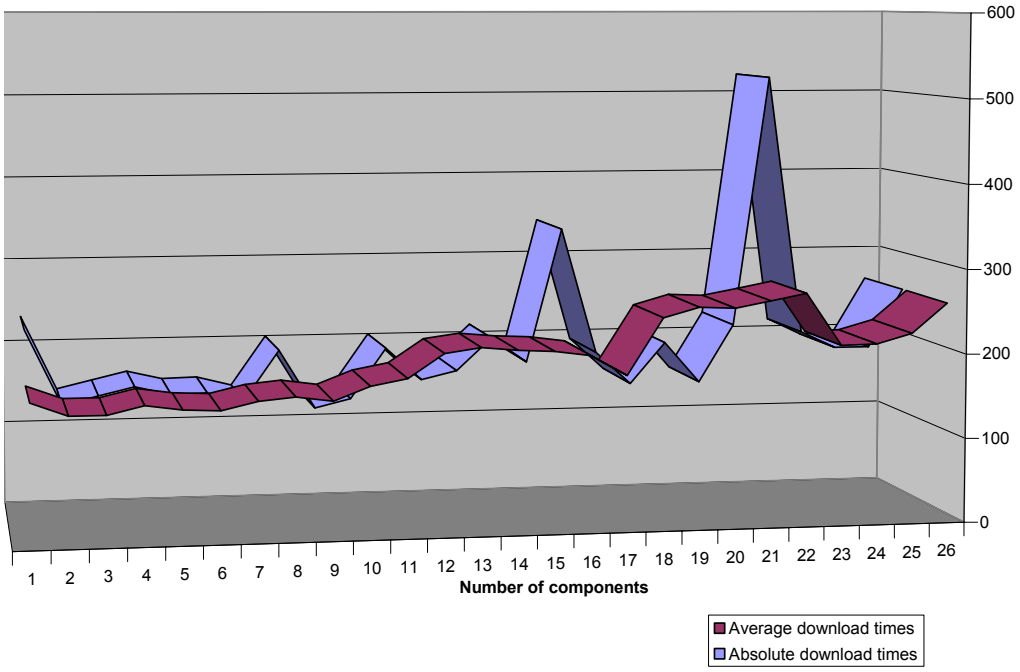

Figure 6 - 3. Download time statistics for the telnet test

In the figure we observe how the average download time starts growing slowly, however it remains almost stable.

Figure 6 - 4 shows the polling time statistics for the telnet test.

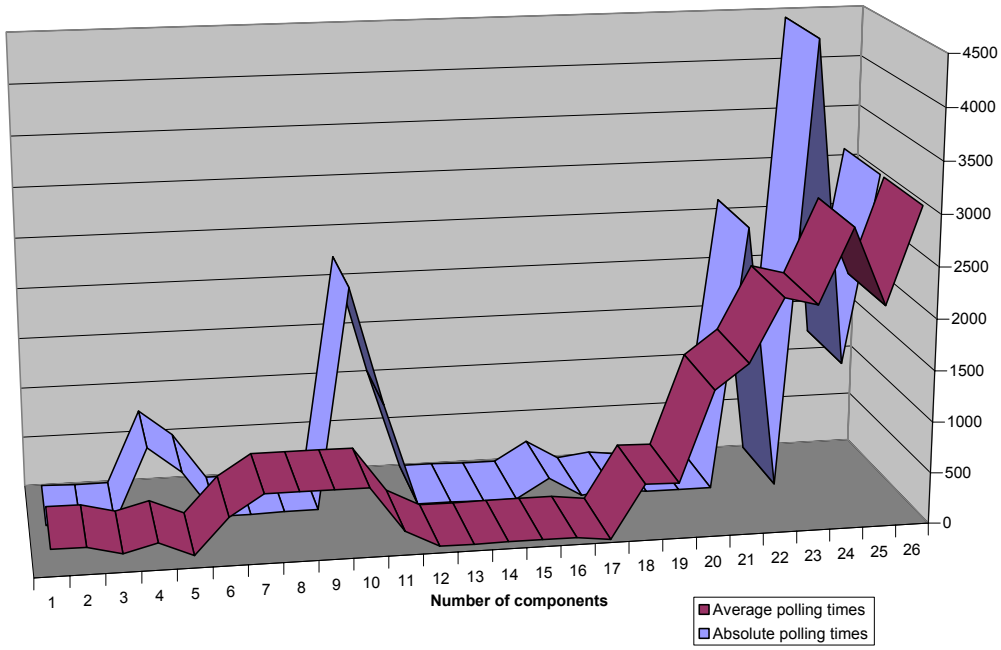

Figure 6 - 4. Polling time statistics

The absolute polling time have a great variance over time, although it is clearly observable that it starts growing from around the 20 components.

The figure below shows the statistics for the enforcement of the policy over the CISCO router.

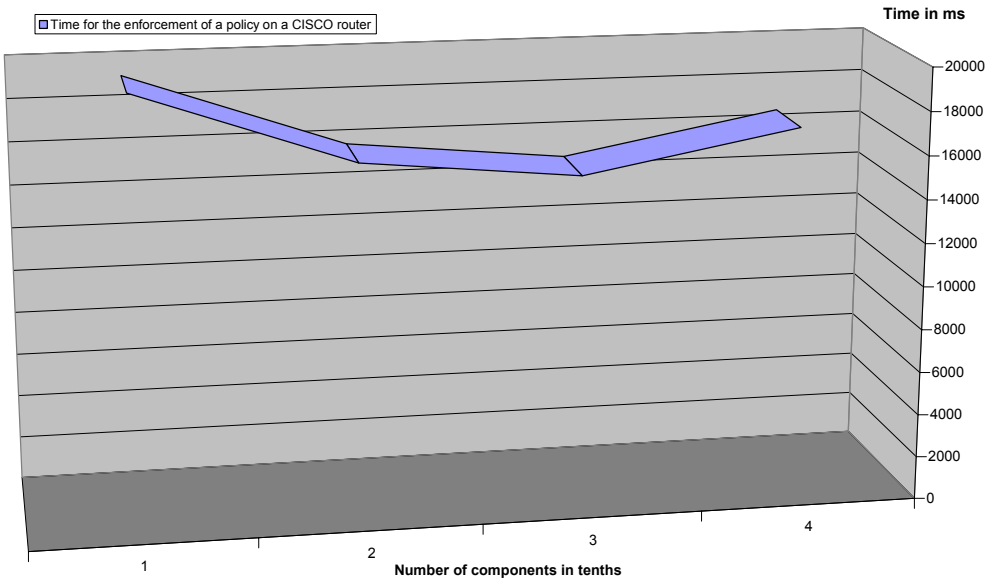

Figure 6 - 5. Policy over CISCO time statistics

In the figure we can see that, in contrast with the downloading times, and more clearly with polling times, the time for enforcing the policy over the CISCO router remains stable. This leads us to the conclusion that the system is not heavily loaded with 30 components.

The two figures below show respectively the size of used heap and the management traffic generated by the management station.

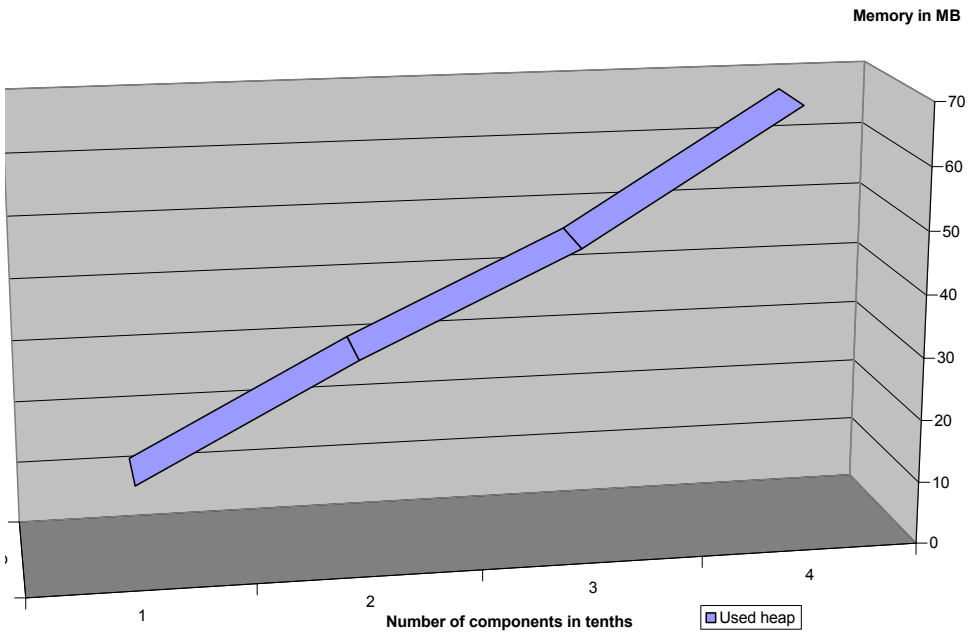

Figure 6 - 6. Size of the used heap statistics

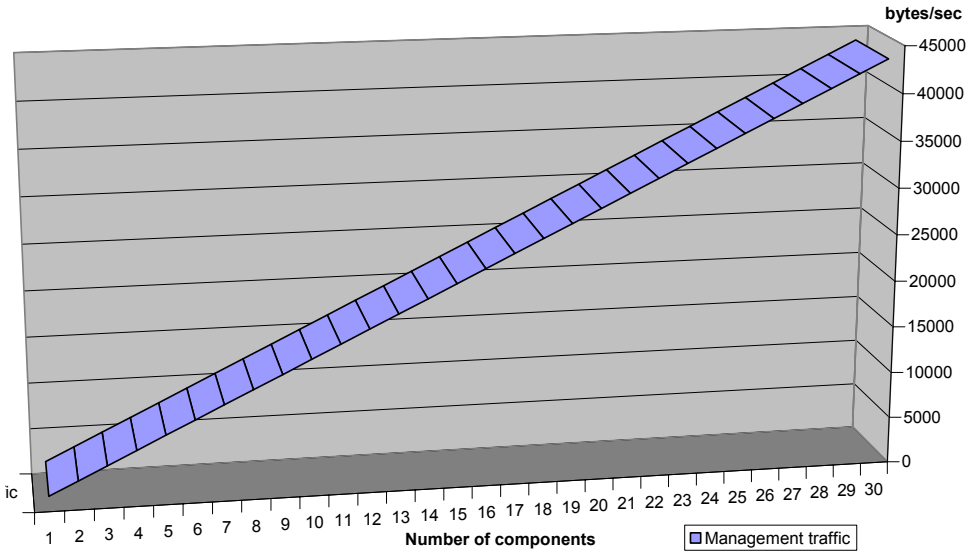

Figure 6 - 7. Generated management traffic statistics

Both the heap and the management traffic grow at a constant rate. The reason for this behaviour in the management traffic is due to the fact that each component generates a fix bandwidth of 1453 bytes/sec. In what relates

to the heap, the reason is that each component uses approximately the same heap size, around 1.7MB.

The figures that follow pertain to the test with the simulated monitoring components. The first figures show the behaviour of the framework when having a network-level only management infrastructure.

The first three figures shown are the component download time, polling time and policy enforcement on a CISCO router time statistics respectively.

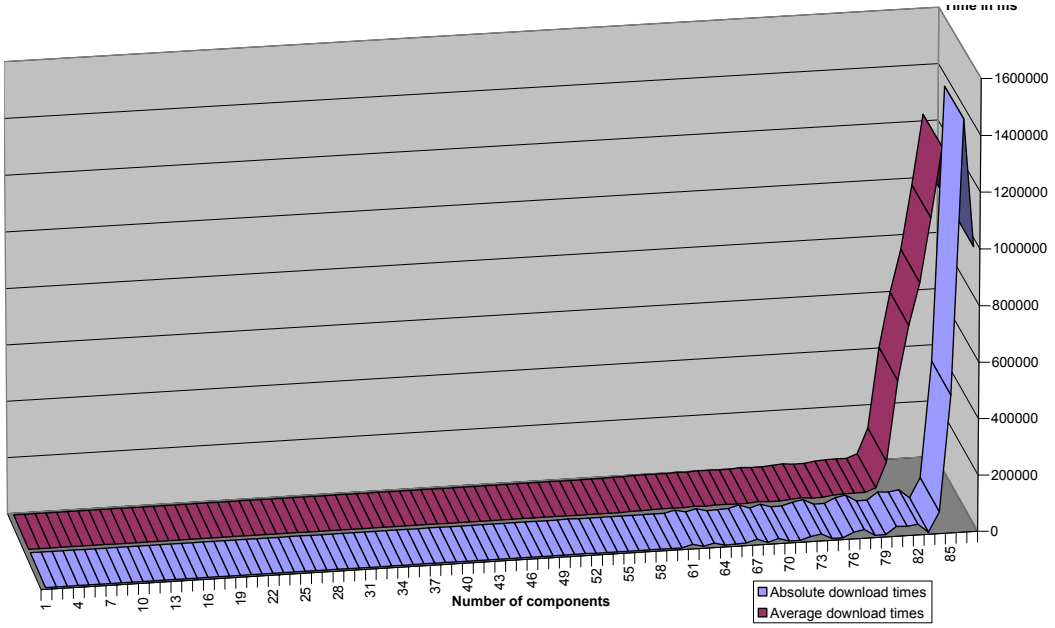

Figure 6 - 8. Download time statistics

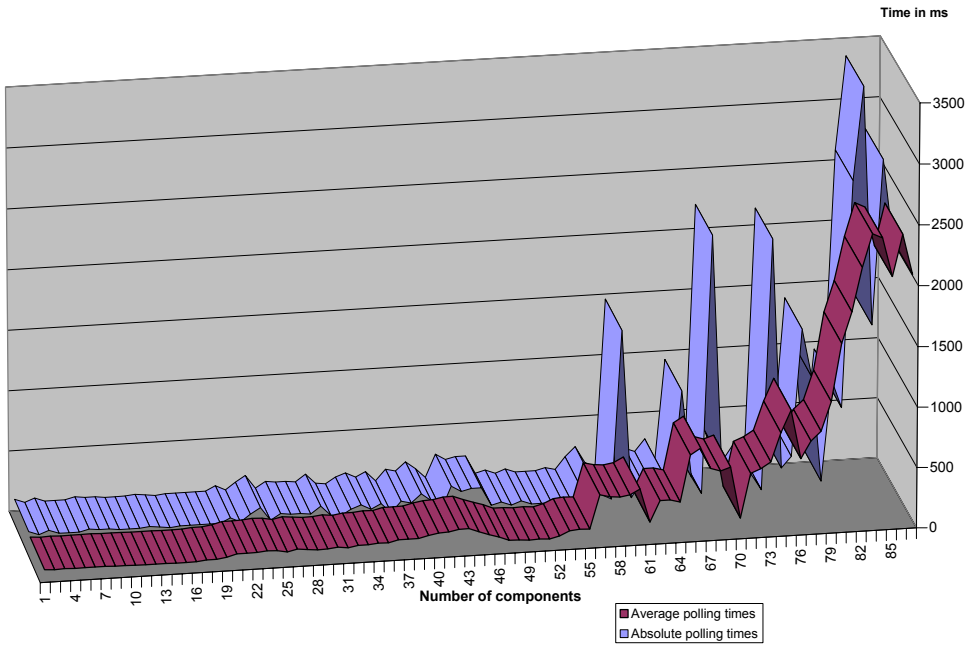

Figure 6 - 9. Polling time statistics

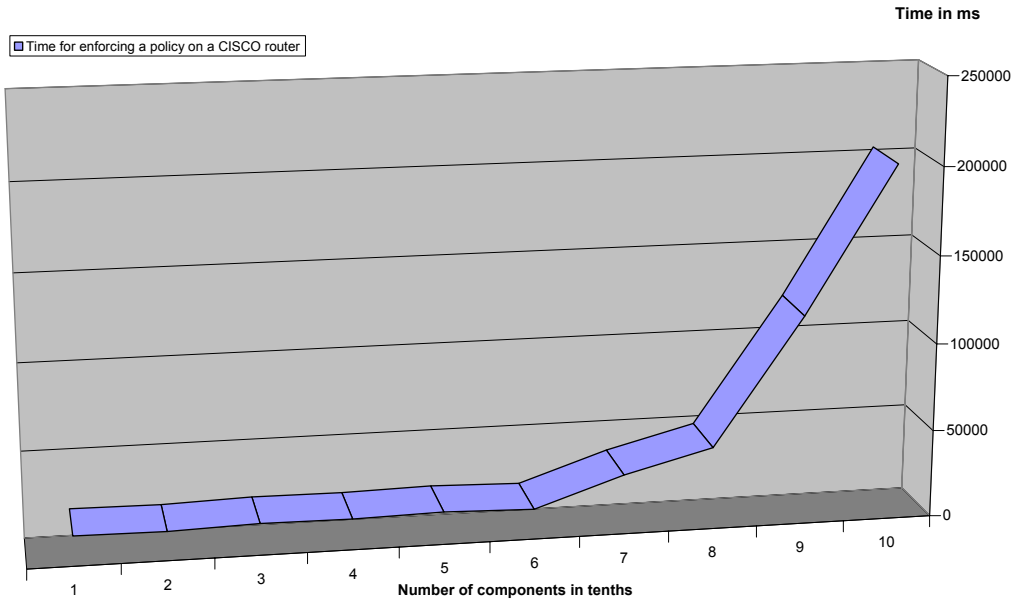

Figure 6 - 10. Policy enforcement time on a CISCO router statistics

In the three figures we can see the same behaviour. When reaching 80 components all times grow significantly. In the test we could not load more than 91 components because the system was not able to load component number 92. After half an hour of waiting this component to be loaded we stopped the test. At 90 components all delays have grown more than 20 times the initial time.

The figure below shows the statistics for the size of the heap used.

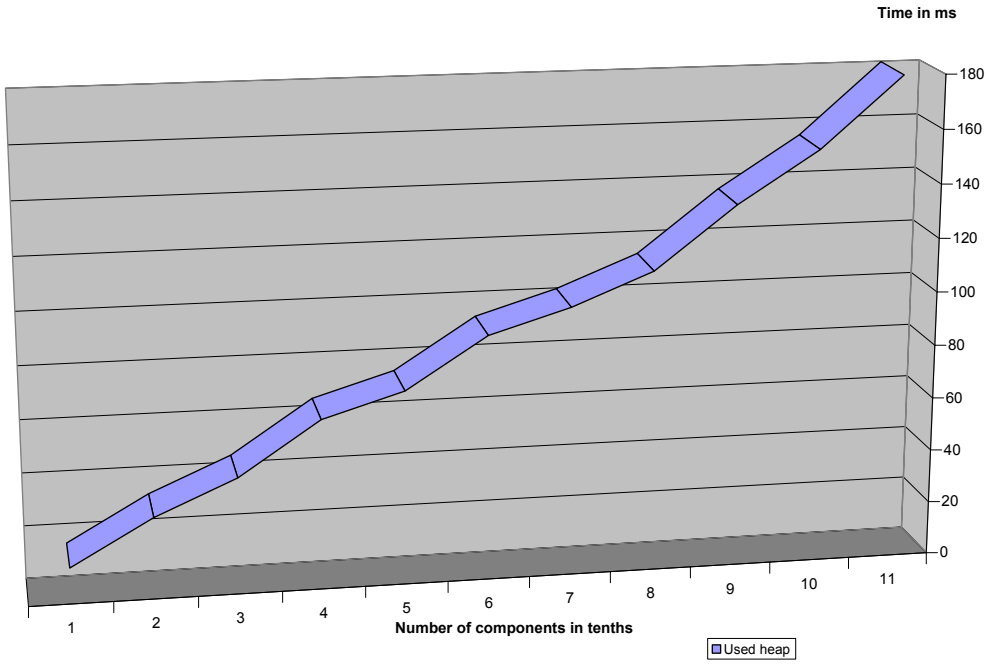

Figure 6 - 11. Used heap statistics

As in the telnet test, the load of the heap grows at a constant rate with the number of components. When reaching the 92 components the heap size is slightly higher than 180 MB.

In the second part of the scalability test with the simulated components we evaluate how the system behaves with a network-level over element-level management infrastructure. With such a management infrastructure the monitoring is realised at the element-level. The network-level functionality is limited to translating the network-level policy to an element-level one. As the policy information is independent of the underlying managed device we only need one component to translate all policies for all routers. Therefore, the network-level station only loads one single component in the whole test. For this reason the heap size remains constants. The second consequence of doing the monitoring at the element level is that the task at the network-level

is limited to the translation of the policy, which is done only once. Hence, the CPU is not being used periodically as with monitoring components doing polling tasks. In consequence, the CPU load is low during the entire test and the times for enforcing the policy on the CISCO router remain constant. The polling times have also been measured at the element-level but since each element-level manager receives only one policy, each EL manager loads only one monitoring component. Another data to be taken into account is the management traffic generated between the network-level station and the element-level station. In the best case, when the monitoring component is already installed in the EL station, the forwarding of the policy generates a total of 13799 bytes. Hence, for the 100 policies, this makes a total of 1.3Mbytes of management traffic distributed on the total time needed for processing the 100 policies. Just ten seconds of monitoring of the 100 monitoring components at the network-level station in the network-level-only infrastructure generate an equivalent traffic.

In the worst case, the monitoring component would need to be downloaded to the EL station. This would generate, together with the forwarding of the policy, a total of 27206 bytes of management traffic. Hence, the processing of the 100 policies, all needing to download the monitoring component, would generate 2.7Mb of management traffic, equivalent to 20 seconds of monitoring of the 100 components in the network-level-only case.

The figure below shows the evolution of the times needed for the enforcement of the policy on the CISCO router as the policies are processed.

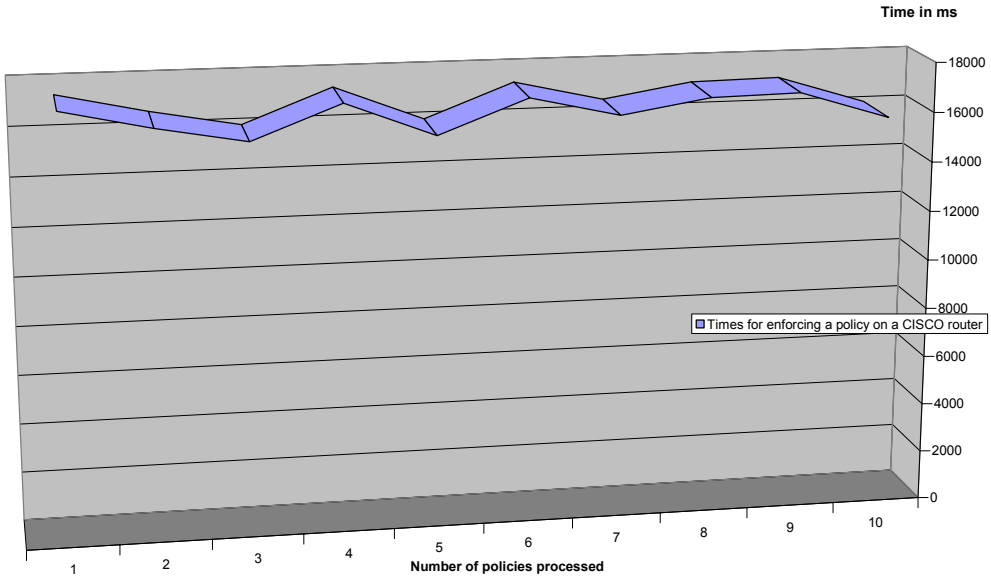

Figure 6 - 12. Policy enforcement on a CISCO router statistics

The rest of the data obtained is summarised in the table below:

	NL station	*EL station*
Used heap	Stable at 16MB	14MB
Polling time	Not applicable	91ms
Component-download time	151ms	748ms

Table 6 - 7. Summarised data for the scalability test with the NL over EL management infrastructure

Summarising, the scalability test has shown that a network-level over element-level management infrastructure solves to a great extent the scalability problems that might appear because of the managed network size. When running a network-level over element-level management infrastructure the system behaved perfectly for 100 components. While, for the same number of components the network-level-only management infrastructure blocked.

c Scalability in terms of number of users

In this sub-section we are going to extend the arguments and considerations pointed out at the introduction of the section. There are mainly two arguments that justify our assertion that the number of users would not create a scalability problem on the MANBoP system. The first one is the likely reduced number of users of the management system: in the order of tenths (most likely even less than ten) [FAIN01]. The second one is the rate of policies (i.e. management requests) that these users might generate. In order to have an approximate idea let's analyse the first scenario. In this scenario a service provider requests a Virtual Active Network for his own, which requires four policies, and creates and configures a webTV service for his customers, which requires again four more policies. This makes a total of eight policies for getting a VAN, creating a service and configuring it. It is not adventurous expecting much less policies for service maintenance and reconfiguration. In the scenario there is only one policy for this task. Hence, taking into account that, first, resource reservation and service creation are tasks realised only once in the service lifetime (usually a lifetime of days or even months or years), and second, that the number of users is expected to be small, we can state that there are few chances of facing scalability problems in terms of number of users.

Nevertheless, let's figure out a bad scenario in terms of number of users. Let's imagine a peak of 100 policies at the same time to analyse how would the framework behave. As it is implemented now, based on the statistics obtained in the first evaluation scenario, the system spends an average of 1000ms in processing the policy (omitting enforcement time), which can be developed in a different machine as we have already seen. Let's now assume that this number cannot be reduced (although it can surely be). With these assumptions, in the worst implementation case (a policy must wait until the previous one is completely processed) the last policy would be processed after 100 seconds, which is clearly not acceptable. However, with a good implementation policies would be processed in parallel and only in certain

processing points they might need to wait. After the statistics obtained in the evaluation these points will probably be the validation of the policy (60 ms in average[35]) and the component download (300 ms in average). The second time, the component download will only be realised for a relatively small number of the 100 policies since most of the times the component will already be downloaded. Furthermore, both the component download and the policy validation times can probably be reduced by downloading components in parallel and by having multiple validator instances running. Hence, with an optimised implementation (always keeping JAVA as programming language even if it performs worse), it is realistic estimating a policy processing rate of one policy every 50ms (20 policies/sec). With such a rate the last policy of the 100 received would be processed after five seconds, which for the management plane is an acceptable number. A rate of 20 policies/second allows having 20 service providers sending one policy per second in average, which is certainly not expectable.

Even though being highly unlikely, let's now imagine what solutions the network operator might have for higher rates of policies per second received. As we have extensively seen, the flexibility of the MANBoP framework allows the network operator to create the management infrastructure that best fits its needs. With a distributed management infrastructure (e.g. network-level over element-level) the scalability of the framework in terms of number of users is enhanced since many of the policies can be directly introduced at the element-level managers as element-level policies, thereby reducing the policy rate at the management station. From the nine policies in the evaluation scenario, six (the four service policies and the policies for creating and activating the VEs for the service provider) could have been introduced directly at the corresponding element-level managers. Furthermore, if the network operator still faces scalability problems, it might even create a management infrastructure with several network-level managers managing each of them a part of the network and communicating via specialised Monitoring Meter and Policy Consumer components.

Along the sub-section we have justified why we do not expect the management framework to experiment scalability problems in terms of number of users, and hence, why we have not realised any evaluation in that sense. We have also reflected about how will the system behave in front of a high load of policy requests and what solutions offers the MANBoP framework to network operators to solve this kind of scalability problems. We have seen how the flexibility properties of MANBoP are, once more, useful to solve many management problems, and particularly, scalability problems.

---

[35] We are considering here only the time where the XML validation is made and not other processes also related with the validation, which have been considered in the delegation evaluation, but not considered here because they can be realised in parallel.

d Security

The security aspects of the system have been never considered for the MANBoP framework as they are seen as out of the scope of this project thesis. The reason for this is that the management framework we wanted to design does not offer any challenge in terms of security. There are many existing security tools that could be used to secure the framework. The inclusion of these tools within the framework requires a substantial effort, and on the other hand, it would not provide any added value in terms of research interest neither on the framework's functionality. In consequence, I decided to skip the security issues and focus the effort on other framework functionality.

In the conclusions and future work chapter of the thesis we will elaborate extensively around the basic security mechanism that can be added to the framework.

Nevertheless, some minor, effortless, security mechanisms have been included in the framework's implementation. These security mechanisms are mainly two: authentication and authorisation mechanisms.

The authentication system implemented is based on a simple login and password strings. These strings are included in the method used to introduce the XML policy in the system. The system checks that there is a registered user in the system with that login and password. Such login and password strings represent the user who is introducing the policy and against whose access rights the policy must be validated.

The XML policy includes another couple of login and password. These ones represent the user to whom the policy affects, either modifying his assigned resources or delegating to him new management functionality.

In any of the two cases the login and password strings are encrypted and hence they offer, as they are implemented now, weak security.

The second security mechanism implemented is an authorisation mechanism that comes from the implementation of the delegation functionality in the system. The delegation of functionality, as already described, is based on the creation of restricted XML Schemas to which validate user's policies. This functionality represents a simple authorisation mechanism based on the XML validation properties. The user management request (represented as an XML policy) is validated against the user access rights (represented as a restricted XML Schema). The restricted XML Schemas assigned to each user are included within Information Model Objects (IMOs), called Schema objects, in the Database.

Again, this authorisation mechanism is relatively weak because there are no integrity mechanisms implemented for the XML Schemas, neither for the Schema IMOs.

The performance data obtained in the evaluation for these security mechanisms implemented have been already given in part in the Delegation sub-section. However, in brief, the authentication procedure spends an average of 10ms to be completed, while the authorisation mechanism spends an average of 130ms.

Summarising, the security of the MANBoP framework is yet on a foetal state and several security mechanisms should be added to guarantee the safety of the system. In the conclusions and future work chapter we will elaborate a bit more on which could be these security mechanisms.

E Interworking

The interworking of a system depends mainly on three aspects: the level of standardisation of the technologies used, the level of standardisation of the interfaces offered and the level of standardisation of the adopted Information Model.

In what concerns to the technologies used in MANBoP they are all well standardised and offer good interworking capabilities. The programming language, JAVA, is widely known and standardised. Nevertheless, the programming language is not so important since we are working over a CORBA platform [OpenORB] that provides programming language transparency, as well as location and technology transparency.

The fact of using CORBA for the implementation of the MANBoP framework is probably the most important aspect in terms of interworking because it simplifies a lot the interworking process.

The last important technology, in relation with interworking, chosen in MANBoP is XML. The XML language is used for expressing policies. The XML language is becoming a de-facto standard for the expression of different types of documents, including policies.

The use of XML language for expressing MANBoP policies eases the interworking with the MANBoP interface. Furthermore, CORBA technology can be used to discover the interface dynamically.

MANBoP offers three external interfaces to request management actions. These are, a GUI offered by the Policy Editor (not implemented in the proof-of-concepts), the upper interface to receive policies from higher-level applications and the lower interface to receive signalling request (not implemented neither).

The GUI is a guided, easy-to-use interface oriented to human intervention, hence it requires no interface standardisation and is not interesting in terms of interworking.

The upper interface is a simple interface, easily discoverable with CORBA that introduces just the XML policy and the user's credentials. The IETF Policy Working Group [IETFPol] , which is in charge of standardising policy-

based management, has not standardised any interface for a Policy-based management yet. However, this drawback can be easily overcome by introducing one of the de-facto standard ways of transmitting XML policies (e.g. SOAP [W3C03] and XML-RPC [XMLRPC]) in the interface. In this way XML policies could be easily introduced in the system enhancing a bit more the interworking capabilities of the system.

The lower interface has been designed to support any type of signalling protocol through a dynamic extensibility mechanism (see chapter 4 for more information). Therefore, the interworking capabilities through the lower interface are optimal.

The last criterion to assess the interworking capabilities of the system is the Information Model. This is probably the weakest aspect, in relation with interworking, of the MANBoP framework. In practice, the only Information Model part affecting the interworking is the Information Model used for defining the policies. The IETF has standardised a Policy Core Information Model (PCIM) [IETF]. Nonetheless, in MANBoP we have decided not to use this model and define our own, although based on the PCIM. There are two reasons for this. The first one is that the PCIM model, although it could be adapted to cope with active and programmable network requirements, is not oriented to this goal. The second and main reason, is that the PCIM model is too complex needing several classes to define a policy action or policy condition. We have simplified the PCIM model to reduce its size substantially. Hence, the size of policies is considerably reduced (at least five times smaller than following the PCIM model) and their processing is faster. Furthermore, the use of the IETF's PCIM model is still small and its success is uncertain.

Overall, we can state that the MANBoP framework has good interworking properties basically due to the technologies used in the implementation. The use of the standard Policy Core Information Model would probably enhance the interworking capabilities of the system although it would certainly lower down the overall system performance.

F Portability

The portability properties of a system are mostly based on the technologies used, although a small part has to do also with the way the implementation is made. In this sub-section we are going to analyse the portability criterion of the MANBoP framework.

The technologies used in the implementation of the system (XML, JAVA and CORBA) assure a complete portability on every machine. The only requirements for the machine are some minimum computational requirements, as CPU and memory. The framework has run on a station with a Pentium 166Mhz processor, although this might be the minimum for an acceptable performance.

The second factor that might influence the system portability is how the implementation is made. Particularly, how are implemented the interactions with the computer resources (e.g. disk files, etc.). To have full portability of the system, this must be done in an OS-independent way. In MANBoP we have taken particular care in this sense.

As conclusion of the portability analyses we can highlight that the MANBoP framework has run with success in different Windows and Linux computers without any adaptation needed. Hence, the system has good portability properties as long as the minimum computational requirements are available.

2nd Statistical Criteria

In this section we recompile and present all performance evaluation data obtained during the evaluation of the framework. Part of this information has been already presented when assessing some functional criteria.

In those parts where there is performance information publicly available from similar projects (i.e. FAIN) we will compare both and try to justify the differences.

The information obtained when running the scalability scenario has already been all presented in the scalability evaluation sub-section. For this reason, we will not include it in this section.

A Delay and CPU

The two tables below contain a summary of the performance evaluation data related with processing times and CPU consumption obtained when running the scenarios.

The first table provides the main processing times, as well as the total time, for the bootstrap and for all policies processed in the first scenario. These times are given for the network-level station when working both alone and over element-level managers and for the element-level stations.

	Policy	Group processing time	Validation time	Conflict checking	Extensibility time[36]	Enforcement [/Monitoring]	Total policy-processing time
Network-level-only management infrastructure	bootstrap	-	-	-	-	-	2935ms
	1st VAN policy	160ms	140ms	230ms	1062ms	3977ms	6179ms
	2nd VAN policy	90ms	211ms	140ms	0	201ms	1923ms
	3rd VAN policy	110ms	130ms	120ms	0	150ms	611ms
	4th VAN policy	90ms	120ms	130ms	0	18146ms	18737ms
	1st Service policy	90ms	140ms	121ms	400ms	4477ms	5388ms
	2nd Service policy	110ms	91ms	110ms	0	6469ms	7090ms
	3rd Service policy	70ms	80ms	100ms	0	5458ms	5818ms
	4th Service policy	70ms	80ms	110ms	0	10205ms	10615ms
	CISCO policy	-	180ms	150ms	591ms	3575 / 27ms	4647ms
Network-level over element-level	bootstrap	-	-	-	-	-	2744ms
	1st VAN policy	90ms	140ms	140ms	876ms	110ms	1782ms
	2nd VAN policy	100ms	241ms	200ms	401ms	410ms	1612ms
	3rd VAN policy	101ms	150ms	80ms	0	361ms	812ms
	4th VAN policy	150ms	171ms	90ms	0	150ms	671ms
	1st Service policy	60ms	110ms	80ms	150ms	51ms	581ms
	2nd Service policy	80ms	100ms	90ms	0	111ms	431ms
	3rd Service policy	70ms	70ms	80ms	0	220ms	530ms
	4th Service policy	80ms	90ms	80ms	0	70ms	421ms
	CISCO policy	-	70ms	231ms	291ms	140ms	2274ms
Element-level	bootstrap	-	-	-	-	-	
	1st VAN policy	-	271ms	293ms	537ms	1927ms	5465ms
	2nd VAN policy	-	84ms	120ms	140ms	47ms	383ms
	3rd VAN policy	-	152ms	90ms	0	123ms	425ms
	4th VAN policy	-	112ms	110ms	0	5537ms	5789ms
	1st Service policy	-	110ms	100ms	231	4418ms	4893ms
	3rd Service policy	-	70ms	121ms	0	5283ms	5501ms

Table 6 - 8. Policy processing times summary

The second table contains those MANBoP packages that require, in average, more computational power. This information is given as an average global number for all possible positions within the management infrastructure.

Component name	Average time
Policy Consumer (PC)	3006ms
Code Installer Application (CIA)	542ms
Policy Consumer Manager (PCM)	202ms
Decision-making Monitoring system (DmMs)	191ms
Database (DB)	176ms
Monitoring Meter (MM)	140ms

Table 6 - 9. MANBoP components with higher processing times in average

[36] With extensibility time we refer to the total time needed to download all components that have been installed to process the policy.

a Comparison against FAIN

The only performance data publicly available from the FAIN project are some processing times published in the deliverable D40 [FAIN03a] and other figures [FAIN03b]. Nevertheless, it is very hard to take any conclusion from the comparison between both proposals since the scenario, the testbed or the times taken are not exactly the same in any case.

VAN creation time	VAN activation time	Deploy functional domain	
		QoSPDP	DlgPDP
38000ms	53000ms	1000ms	250ms

Table 6 - 10. Processing times in the FAIN NMS

The processing times shown in the table above correspond to the FAIN NMS in a scenario very similar to the first evaluation scenario presented. However, the machine used as NMS in that testbed was an Intel 166 Pentium. This might explain the different results when compared with MANBoP. In MANBoP the VAN creation time is 6179ms and the VAN activation time is 18737ms. Moreover, since these times are so generic that it is hard to find out the cause of these differences.

In relation to the times for deploying new functional domains, these times have a lot to do with the size of the component being downloaded. In MANBoP these times are around 800ms for the QoSPC component and around 400ms for the DelegationPC.

FAIN provides some more times obtained at the FAIN EMS. In this case the machine is an Intel 1,5GHz Pentium IV machine with 500 MB of RAM. The table below shows this information.

Deploy Policy		Deploy Functional Domain		
QoS	Dlg	QoS PDP	Service PEP	Dlg PDP
19000ms	10500ms	6000ms	4500ms	160ms

Table 6 - 11. Processing times in the FAIN EMS

The interest of the EMS data, in terms of comparison with MANBoP, is in the processing times for the deployment of the QoS and delegation policies. In MANBoP these times are around 5500ms for a QoS (VAN1 or VAN4) policy and around 400ms for a delegation policy (VAN2 or VAN3). It is hard to find an explanation to the differences between the times, however in the case of the delegation policies it might be perfectly caused by the different delegation mechanism implemented in FAIN and MANBoP.

In FAIN the delegation mechanism implemented by default is the creation of a Management Instance for the service provider. A Management Instance is an environment within the management station where the service provider obtains the delegated management functionality. Within this environment the

service provider is free to do anything. It is a concept similar to that of a sandbox.

The process of creating this Management Instance is much more resource-consuming that the delegation process in MANBoP where only a restricted XML Schema is created and assigned to the service provider. This might explain the big difference in the processing times of the delegation policies between FAIN and MANBoP.

As commented before the times for the deployment of components are closely related with the size of the component, hence no direct conclusion can be extracted from the published times.

Finally, the last performance data for the FAIN management system has been obtained from [FAIN03b]. The information published in this paper is shown in the table below:

VE time	*Instantiation of components times*	
	QoSPDP	*DelegationPDP*
10637ms	212ms	176ms

Table 6 - 12. Processing times in the FAIN EMS for a VE and delegation

The times published in the paper were those that where necessary to first, create and activate a VE for a SP and second the times needed for instantiating the QoSPDP and DelegationPDP components in an EMS station. The machine where these measures were taken was an Intel 1,5GHz Pentium IV computer with 500 MB of RAM. In this case, the scenario was completely different since the testbed consisted of just one FAIN ANN and one EMS.

The VE time shown in the table is the time for creating a VE for a service provider, creating a MI to this service provider and activating the VE for that service provider. These processes are similar to those done when enforcing the policy group for creating the VAN in the MANBoP EL manager. The times for enforcing these four policies in MANBoP make a total of 12062ms, which is a similar number. Again, it is very hard to find out the differences in the comparison, although it might probably have to do with the scenario, particularly if we take into account the difference between the numbers in the FAIN D40 deliverable and the FAIN paper. Also, another possible explanation for the differences is that in the scenario published in the paper the dynamic components are not downloaded but just instantiated when they are needed.

B Memory

In this sub-section we will provide graphics with the size of the heap used by the MANBoP framework at any time of the management process when running the first scenario. Additionally, we also provide the size of the heap being used at any time by the CIA component.

At the end of the sub-section we will also provide some numbers on the disk size occupied by the Database.

The figures below show the evolution of the used heap for the MANBoP components as well as for the CIA component. There is one figure for the network-level station when working alone, another for the network-level station when working over element-level managers and a last one for the element-level station.

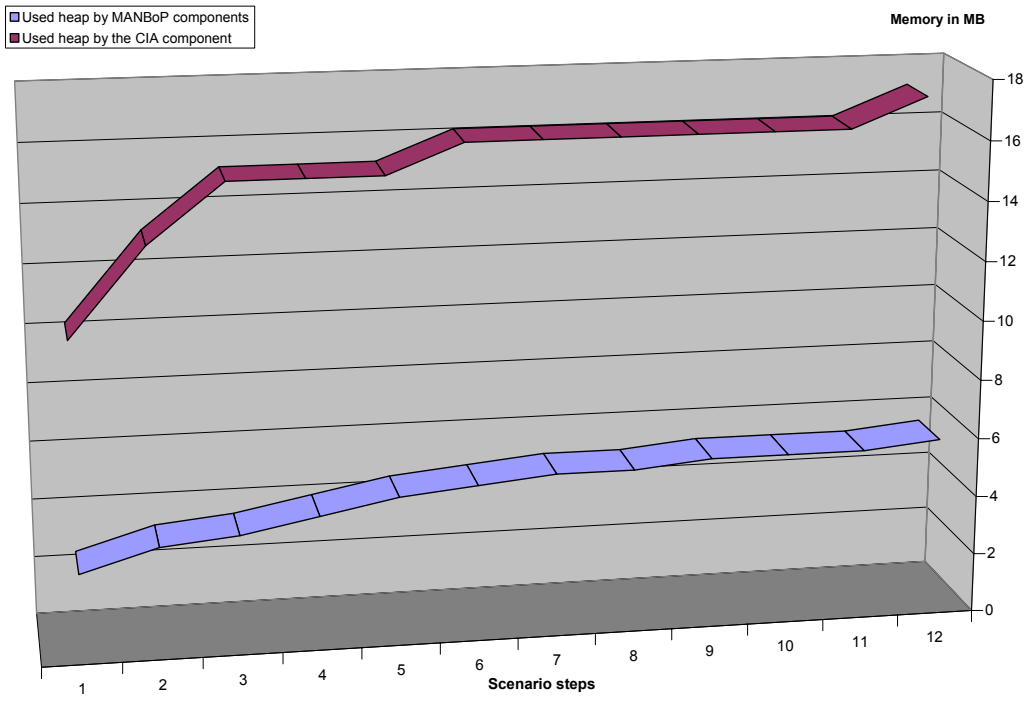

Figure 6 - 13. Network-level only Used Heap

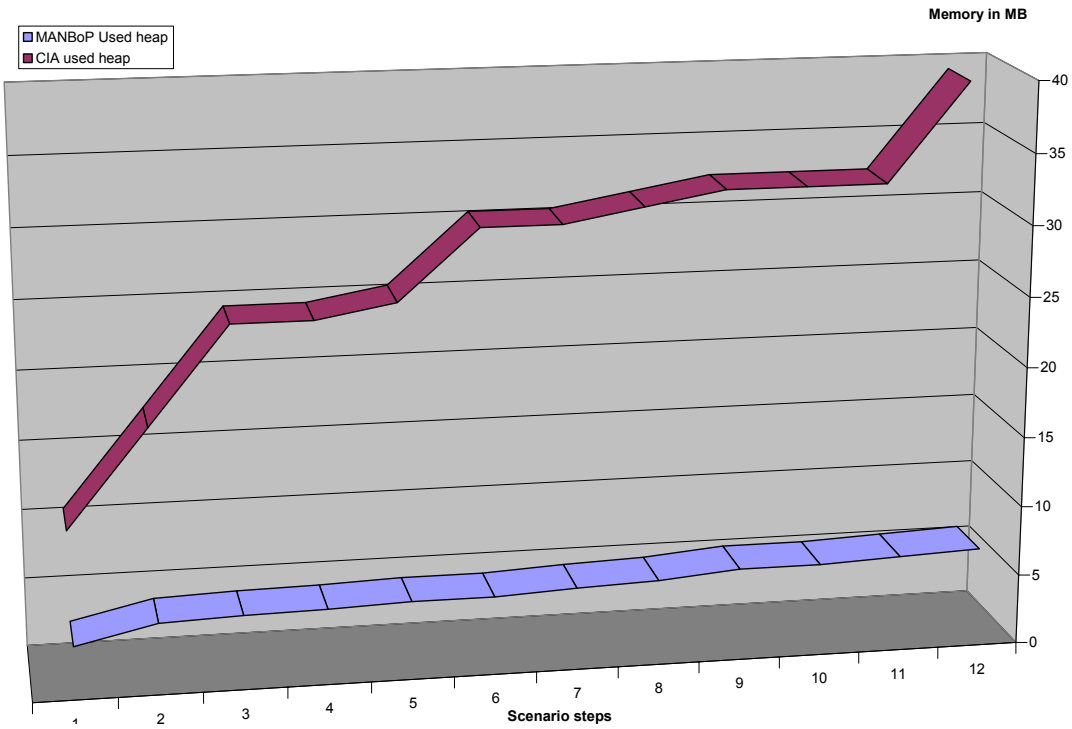

Figure 6 - 14. Network-level over Element-level Used Heap

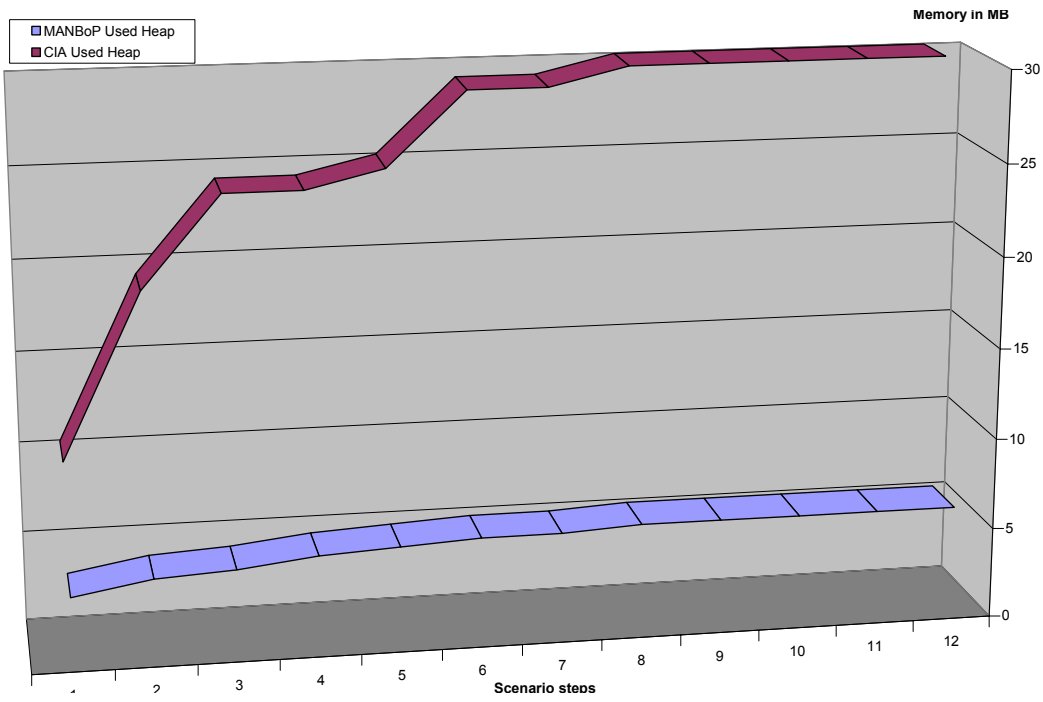

Figure 6 - 15. Element-level Used Heap

There are few remarkable comments to make about these figures. In relation to the heap used by MANBoP, it is interesting to note that it grows with the number of policies processed, not with components loaded as the CIA does, and how it tends to stabilise around five megabytes.

The CIA heap depends a lot on the components loaded into the system and how much memory do they require as these components run inside the CIA ORB. On the other hand, policy processing does not almost affect the heap used by the component.

In the table below we give some data about the disk size occupied in the Database and its different parts.

	User Info	User Policies[37]	Manager info	Managed Topology Info	Total
Disk size in KB	48	11,4	1,8	7,6	68,8
Size per unit / unit type	24KB/User	1,2KB/Policy	0,3KB/Dynamically installed component info	1,5KB/Managed device	68,8KB/Manager

Figure 6 - 16. Database size occupied when running the first scenario

C Bandwidth

In the table below we show the statistics obtained for the management traffic generated in the scenario for each of the management infrastructures. We provide the information for both the network-level management station and for the element-level management station.

	NL-only	NL over EL (NL station)	NL over EL (EL station)
Total number of packets	690	873	325
Management traffic packets	418	256	99
Code downloading packets	171	410	118
Naming service packets	101	207	108
Average packet size (bytes)	195	318	300
Average management packet size (bytes)	97	186	188
Average code downloading packet size (bytes)	471	463	501
Average Naming Service packet size (bytes)	135	196	185
Total bytes of traffic	134650	277943	97792
Bytes of management traffic	40486	47775	18613
Bytes of code downloading traffic	80556	189653	59163
Bytes of Naming Service traffic	13608	40515	20016

Figure 6 - 17. Traffic generation statistics

[37] User's policies are indeed part of the user info. Nevertheless, because of their significance we have included the amount of disk they occupy in a different column.

From the table above the most interesting information is comparing the management traffic generated by the network-level station when working alone against the management traffic generated when working over network-level managers.

The total traffic when working over element-level managers is more than two times the traffic when working alone, although most of the difference is due the traffic generated for downloading the components.

We can also see that the naming service related traffic is much higher when working over the element-level managers because a big part of it is generated when preparing the download of the components. Hence, in a situation where the components needed to process a policy are already downloaded the management traffic generated is almost the same.

The reasons why it is still higher when working over element-level managers is that most of the policies in the scenario, indeed all except the enforcement of the last one, is done over active routers (i.e. FAIN and ABLE routers). Therefore, the components that monitor and enforce these policies, and hence generate more management traffic, are located in the machines themselves, and the management traffic with the management station is simply a CORBA request and reply session.

On the other hand, when working over element-level managers the management traffic generated for processing these policies (those over the active routers) is due mainly to two reasons. The first one is that the policy-processing request to the element-level managers generates more traffic, as the XML policy is included in this request. The second one is that in this case, the reply is not sent as result of the method called but afterwards, hence at least a second CORBA request and reply session is needed.

If we look only at the traffic generated for the policy enforcement in the passive router the management traffic generated by the network-level manager, in the network-level only management infrastructure this traffic is 16363 bytes in 272 packets, while for the network-level over element-level management infrastructure is 4483 bytes in 30 packets.

Summarising the behaviour in terms of bandwidth performance will only be better for a network-level-only management infrastructure when the percentage of management actions over active routers is much higher than the percentage of management actions over passive routers.

Section VI.4 – Conclusions

In this chapter we have evaluated the MANBoP framework based on a number of criteria that have also been described. The evaluation criteria have been extracted from analysing the Thesis objectives. These criteria have been grouped in two types: functional criteria and statistical criteria.

The functional criteria, which comprises flexibility, extensibility, delegation, scalability, security, interworking and portability, is oriented towards assessing the functional aspects of the MANBoP framework.

On the other hand, the statistical criteria, comprising processing time and CPU, memory and bandwidth, is targeted towards recompiling, analysing and assessing the performance figures of MANBoP.

After the description of the criteria followed to evaluate the framework we have presented and analysed the evaluation results for each of the above-mentioned criteria.

A general output from the evaluation of the framework is that, probably, the design decision with a higher impact on the performance of the management system was that of using a policy-based management system. When compared with other, non policy-based, alternatives the main drawback of a policy-based system is basically its worse computational performance since policies must be parsed and interpreted. Nevertheless, on the management plane, where decisions are usually long-term decisions, this is not an issue as crucial as in the data or even control planes.

On the other hand, the election of a policy-based management system has many advantages in terms of flexibility and even management station load.

The first advantage of a policy-based management system is that policies allow a more autonomous management. With a policy-based management system the operator can specify the rules that govern the network behaviour.

Another advantage of a policy-based management system, indeed of being an interpreted system, is that it eases the process of dynamically extending the management functionality by downloading new components capable of interpreting new policies. This allows the dynamic addition of new functionality to cope with new requirements or even with new managed resources.

Finally, the fact of having independent components dynamically installable and removable from the system is helpful to reduce the overall load of the management station by removing those components that are not being used.

All these advantages together with the natural fit of policies for specifying delegation of functionality and access rights, one of the key requirements for managing active and programmable networks, have justified our election of a policy based management system.

Looking into the evaluation of the flexibility criteria, the main conclusion that can be extracted is that the obtained results present the framework as a useful tool for network operators to manage heterogeneous networks in terms of flexibility. The results have shown that the framework permits the management of active, programmable and passive routers (the times for managing different types of routers are on the same scale) without, for this reason, loosing any of the advantages that can be obtained when using the

capabilities offered by these technologies. This is shown by the way we manage FAIN and ABLE routers in the scenarios. In particular, we have demonstrated how the use of ABLE facilities for its management significantly reduces the management traffic required. In brief, the flexibility of the MANBoP framework has appeared as an interesting capability for solving many possible problems that might appear when managing a network.

In relation to the creation of different management infrastructures based on the MANBoP framework we have demonstrated in the scenario the simplicity of the process. Indeed, it is only necessary to start the same piece of code in the different stations with the appropriate configuration files. The remaining processes for achieving the entire functionality are automatic. Moreover, the scalability scenario shows how the possibility of easily create different management infrastructures can be very helpful for solving different problems, particularly scalability problems, in a cost-effective way.

The last aspect analysed within the flexibility criteria is the policy group processing functionality. The results show that the performance penalty paid for supporting this functionality (an average of 6% over the total policy processing time) is small when compared with the flexibility that this mechanism adds to the management framework. The added flexibility derived from including policy group processing functionality is huge since it allows even a higher automatism of management tasks. Without policy group processing functionality it would be the user of the management system (network operator or service provider), the responsible of introducing policies one by one and removing them if an interrelated policy was not enforced successfully.

In what refers to the results obtained from the extensibility evaluation, we can state that it fulfils the objectives introduced at design time. Moreover, the performance implications of the extensibility mechanisms are not significant on the overall system behaviour. The processing times expended on the extensibility mechanism when a new component is installed are around a 13% of the total processing time. On the other hand, the gains obtained from it are enormous; specially taking into account that extensibility support is a must on a management system for active and programmable networks.

The evaluation of the delegation criteria has also given satisfactory results. The delegation mechanism designed and implemented in MANBoP appears as a good alternative for delegation of functionality in a policy-based management system using XML as language for expressing policies. Its main advantages are the wide syntactical and semantical flexibility that it offers, its simplicity and the fact that the most cost-expensive task, the authorisation of policies, is done by specialised code. Indeed, the processing time expended in the authorisation process is around a 10% of the total policy processing time. Another significant data obtained from the evaluation is that the Authorisation Check Component, that carries out the authorisation process,

is not among the six components consuming more computational resources of the framework.

The results obtained from evaluating the scalability criteria indicate that a network-level over element-level management infrastructure solves to a great extent the scalability problems that might appear because of the managed network size. When running a network-level over element-level management infrastructure the system behaved perfectly for 100 components. While, the network-level-only management infrastructure performance was severely damaged when reaching 75 components and at 100 components the system was completely blocked.

When evaluating the scalability criteria we have also justified why we do not expect the management framework to experiment scalability problems in terms of number of users, and hence, why we have not realised any evaluation in that sense. We have also reflected about how will the system behave in front of a high load of policy requests and what solutions offers the MANBoP framework to network operators to solve this kind of scalability problems.

The evaluation of the security criteria has been developed as a way to describe the scarce security mechanisms actually implemented in MANBoP as security is considered as out of the scope of this Thesis. Summarising, MANBoP security is yet on a foetal state and several security mechanisms should be added to guarantee the safety of the system. In Chapter Seven we will elaborate a bit more on which could be these security mechanisms.

The results obtained from evaluating the interworking criteria show that the framework has acceptable interworking properties. These properties are mainly due to the technologies used in the implementation. The use of the standard Policy Core Information Model would probably enhance the interworking capabilities of the system although it would certainly lower down the overall system performance.

The last functional criteria evaluated were the portability criteria. The results have demonstrated that the MANBoP framework has run with success in different Windows and Linux computers without any adaptation needed. Hence, we can state that the portability of the system is good as long as the minimum computational requirements are available.

The evaluation of the statistical criteria provides a recompilation of the performance figures of the framework. In terms of processing times, it is interesting reviewing the six components that spend more time realising their tasks. The one expending more processing time is the Policy Consumer component followed by the Code Installing Application, Policy Consumer Manager, Decision-making Monitoring system, Database and Monitoring Meter components. These results are not surprising as, in the one hand the time needed for enforcing a policy depends a lot on the managed device and, on the other hand, when the Code Installing Application component is called

it must find, download and install the dynamically installable component required.

In those occasions when it has been possible, we have compared the data obtained with that available from the FAIN project. Although the observed figures were different the comparison was difficult to make due to different evaluation environments or very high-level information coming from the FAIN project.

The results obtained from evaluating the memory consumed by the framework indicate that the heap consumed by all MANBoP components except the CIA remains almost stable around 5 Mb. On the other side, the heap consumed by the CIA component depends on the number and size of the dynamically installed components that are loaded in the system. We have also measured the disk size occupied by the Database, which for two users, five network elements and ten policies is almost 70 Kbs.

Finally, in terms of bandwidth performance the results show that the framework performs better for a network-level-only management infrastructure when the percentage of management actions over active routers is much higher than the percentage of management actions over passive routers. Otherwise, a more distributed management infrastructure will perform better.

Summarising, along this section we have assessed the MANBoP framework following different functional and statistical criteria. Overall, the results obtained from the evaluation have been satisfactory meeting the initial objectives of the Thesis.

The following chapter presents the final conclusions of the thesis. We will highlight the most relevant points of the work done and elaborate a bit around future work that might be realised to enhance the system functionality.