A Management Information Tree Architecture supporting Efficient Managed Object Selection

Dongjin Han, Wen-Zhe Cui, Youngeun Park, Geonung Kim and Sunshin An
Computer Network Lab., Dept. of Electronic Eng., Korea University,
1-ka anamdong, sungbukgu, Seoul 136-701, Republic Of Korea
{han,cmc, panzee, sunshin}@dsys.korea.ac.kr

Abstract

This paper deals with the architecture, design and implementation of a Management Information Tree(MIT) supporting efficient Managed Object(MO) selection. We present the architecture and mechanism of the MIT in detail and also show how the proposed architecture is implemented in a Telecommunication Management Network(TMN) platform. In addition, we present a new mechanism decreasing MO selection delay, named as Class Level Filtering(CLF). The main idea of CLF is to use the class information of MOs to exclude the scoped instances of improper MO classes from the conventional filtering. Analysis and performance tests in various cases are presented. Results show a superior performance of the MIT and CLF.

Keywords

Class Level Filtering, Management Information Tree, Scoping, Filtering, Network Management Platform, Telecommunication Network Management, OSI Management, Management Information Base, Managed Object

1. Introduction

As communication networks become massive and distributed, they require the use of open standards-based management systems. Telecommunication Management Network (TMN) provides a globally accepted ultimate framework for the unified management of all types of telecommunication and its underlying services in the future. TMN systems use object-oriented information modeling techniques and agent/manager concepts that underlie the Open Systems Interconnection(OSI) systems management[1].

A Managed Object(MO) is an OSI abstract view of a logical or physical system resource to be managed. Although MO class instances have many relationships, containment is treated as a primary relationship to yield unique names. Instances of MO classes can be thought as logically containing other instances. In a TMN, the full set of MO instances available across a management interface are organized in an Management Information Tree(MIT), which is also referred to as the

containment hierarchy. An MIT requires that an attribute of each instance serve as a naming attribute. The attribute and its value form a Relative Distinguished Name (RDN), which should be unique for all object instances at the first level below. The Distinguished Name (DN) is the concatenation of all the RDN after the root managed object. Managed Objects in an MIT may be accessed either individually or collectively through an object-oriented query facility. Managed object selection involves two phases, namely scoping and filtering. Scoping entails the identification of the managed object(s) to which a filter is to be applied. The scoping is specified with reference to a specific managed object instance, referred to as the *base managed object.* The base managed object is defined as the root of the subtree of the MIT from which the search is to commence. Four specifications of scoping level are defined[2]:

- the base object alone;
- the *n*th level subordinates of the base object;
- the base object and all of its subordinates down to and including the *n*th level;
- the base object and all of its subordinates(whole subtree).

A filter is a set of one or more assertions about the presence or values of attributes in a scoped managed object. If the filter involves more than one assertion, the assertions are grouped together using logical operators. If the filter test succeeds for a given managed object, then that managed object is selected for the performance of the operation. We refer the filter test in an MO to ***MO Level Filtering***, a conventional filtering, compared to ***Class Level Filtering(CLF)***, which we present in the paper as a new concept.

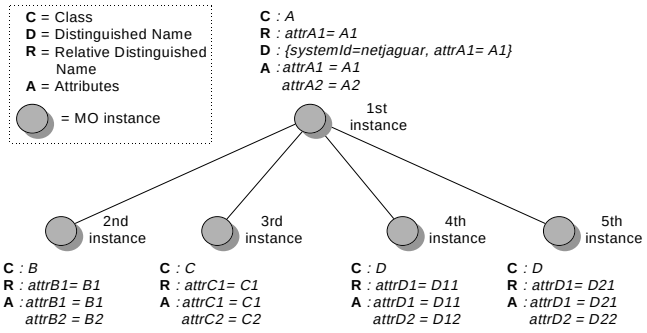

Figure 1. An example of a simple MIT.

When a management operation is performed on multiple objects, a series of linked results is passed back to the managing application in a “back-to-back” fashion. The scoping and filtering has advantages in both expressive power and minimization of management traffic. However, they require unnecessary processing delays and consumption of system resources. This defect is illustrated in detail

using an example of a simple MIT depicted in Figure 1.

In this section, we assumed the base managed object to be *{systemId=netjaguar, attrA1=A1}* and the scope as *wholesubtree*(the base object and all of its subordinates) for all service requests. In the case of the filter not being specified, all instances shown in the example are selected and the MO level filtering is not performed on any of them. If a filter is specified as *(attrD2=D12)*, the MO level filtering are performed on all of the scoped instances and *4*th instance is ultimately selected for the management operation. In this process, we notice that the *1*st, *2*nd and *3*rd instances have no reason to test the MO level filtering. Because the instances don't include the specified attribute(*attrD2*), the result of filter test is obviously *FALSE*. If improper MO instances could somehow be excluded before performing the MO level filtering, the selection overhead can be decreased. The exclusion may be achieved by comparing the class information of scoped MO instances to specified attributes. We name this feature as ***Class Level Filtering***. If CLF is adapted in upper case, MO level filtering are performed only two times rather than five. However, additional overhead occurs owing to the CLF. Because the class level filtering require little processing time than the MO level filtering, the overall performance increases. MO level filtering and CLF have the following differences and relationships :

- MO level filtering is executed by testing the value or existence of specified attributes *in an MO instance*.
- CLF is executed by testing only the existence of specified attributes *based on class information*, which is constructed separately from MO instances in managed application initializing time.
- MO level filtering requires more processing time than CLF. Performance comparison are described in section 4.
- CLF precedes MO level filtering. Managed object instance are selected by performing scoping, CLF and MO level filtering sequentially.

Network management systems which administer actual network resources are developed based on a software system called a network management platform[3]. Network management platforms provide major functions defined in a TMN and interfaces for network management system developers. The functions and performance of a network management platform greatly affects those of a network management system built on the platform. We design and implement a network management platform suitable to the TMN environment of today and tomorrow. The CLF feature and the enhanced MIT are included in the platform.

The remainder of this paper is organized as follows. Section 2 describes the architecture and mechanism of the MIT, which is specially designed to support the class level filtering feature and perform the basic roles efficiently. In section 3, we provide the mechanism of the class level filtering in detail. In section 4, we evaluate the efficiency of our MIT based on case studies. The paper is concluded in section 5.

2. Architecture of Management Information Tree

Instances of MO classes have a containment relationship and can be created in accordance with the naming rule of MO classes. This containment relationship can be thought as logically containing other instances and can be treated as a primary relationship between instances of MO classes to yield unique names. To represent the containment relationship, a managed system administers the MIT internally. The MIT is a starting point for management operations and a key component in a management platform for managing instances of MO classes. By means of the containment relationship, all instances of MO classes are administered and accessed. Therefore, the performance of the MIT highly affects that of a network management system.

A managed system performs three tasks on an MIT to select managed objects. The first task is searching for a base managed object, which is the starting point for the selection of one or more MO instances. The second task is scoping. With the exception of the M_CREATE service, all other management operations permit the managing system to specify not just an object instance alone, but also some containing object instances in the MIT. The final task is filtering. This is a sort of restricting conditions in that it decides whether or not management operations are to be executed on the scoped objects.

2.1 Management of Class Information

In our implementation, all MO instances are classified by MO class information for performing the CLF. Based on OSI system management, an MO class is defined in terms of packages, attributes, actions, notifications, etc. Several MO classes can share the same packages, attributes, actions, or notifications. In other words, packages, attributes, etc. can be used in several MO classes repeatedly. In order to manage the class information efficiently, the management mechanism is designed considering the reusability of the class elements and the searching time. The structure for the class information management is depicted in Figure 2.

To reduce searching time, AA tree is used as the data structure for storing management information. The information of MO classes, packages, attributes, notifications and others are stored in respective AA trees. As shown in Figure 2, an MO class stores the pointers of all packages included in that MO class. In addition, the pointers of all the information related to the packages are stored in the same package. A node in the MIT maintains the pointer of an *MO class info*. Using this information, the CLF can be performed.

2.2 Components

In this paper, the structure of the MIT consists of three components: *MO node , MO Class node*, and *MO Info node*. In addition, this structure uses a doubly linked list along with a balanced binary tree as shown in figure 3. Each MO node maintains a balanced binary tree consisting of MO info nodes and a pointer of a subordinate MO Class node. This tree is used in searching a base managed object. Each MO

Info node maintains a pointer of one node among subordinate MO nodes. Doubly linked list is used in scoping. In the MIT, all MO instances are classified by MO class information which was described above .

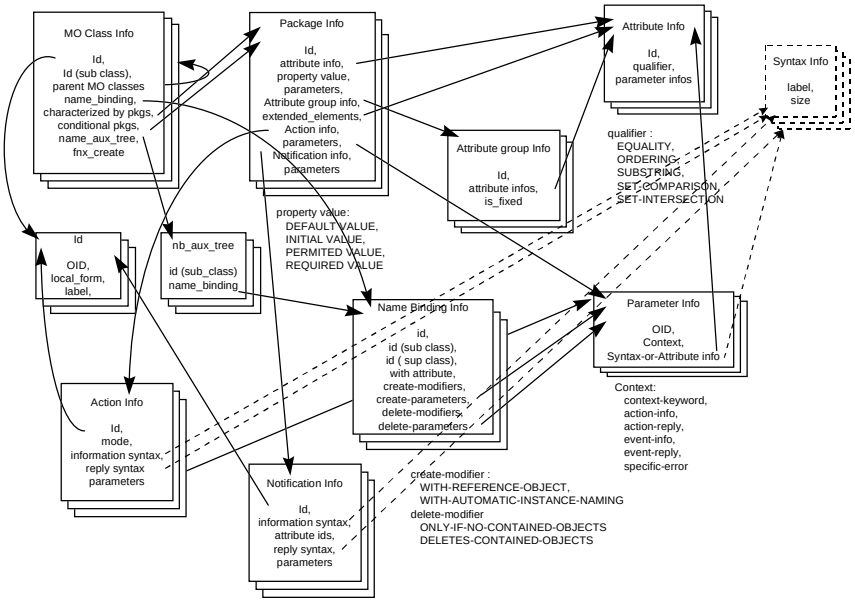

Figure 2. Structure for the class information management.

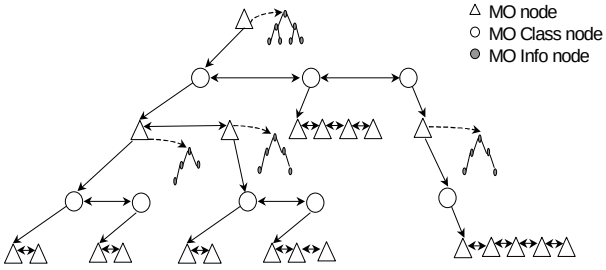

Figure 3. Example of the Management Information Tree.

The MO node stands for an MO instance and has an interface with MO instances which are in the memory, DB, file system, etc. Therefore, management operations sent by a managing system should be executed through the MO node. The elements in the MO node are the pointer of MO instances, DN, MO Class node list, and the balanced binary tree consisting of MO Info nodes. Each MO Info node contains a Relative Distinguished Name (RDN) used as a key value in searching for

a base managed object, and a pointer to a subordinate MO node with the same RDN. An MO Class node is an intermediate node between a parent and several child MO nodes. Each MO Class node maintains the pointer to an *MO class info* which was described earlier.

2.3 Mechanism for Searching and Scoping

In a general case where a managing system requests a management operation with scoping and filtering conditions, a managed system must find a base node at which the scoping process begins, and select nodes in accordance with the scope information. The balanced binary tree maintained in the MO node is used to search for a base managed object. Sibling MO nodes at the same level possess unique names, i.e. RDN, which is used as a key value in searching the balanced binary tree. Searching mechanism using the balanced binary tree is depicted in Figure 4.
Time complexity of searching base MO nodes can be expressed by the following equation.

$$\text{Time complexity} = \sum_{j=1}^{k} \log(N_j) \ ,$$

k = number of RDNs of base node DN
N_j = number of child nodes at level j

After searching for the base MO node, the scoping is performed in accordance with the scope type and scope level specified by a managing system. The scoping is executed recursively in the MIT, resulting in the selection of one or more MO nodes, with the exception of scope type being *BaseObject*. In this paper, a scoping process is implemented as two *while* loops. The outer loop is used for inviting MO Class nodes, while the inner loop for inviting MO nodes. As a matter of course, the scope type determines the behavior of inviting nodes. Figure 5 illustrates a scoping example in which the scope type is *baseToNthLevel* and the scope level is 2.
An MO Class node maintains child MO nodes created from same class. When a child MO node with a new class is created, both the MO class node and MO node are created simultaneously. In the case where all MO nodes with the same class and parent are deleted, the MO Class node managing that MO nodes will also be deleted.

The MO class nodes in the MIT seems to be redundant in scoping. However, the MO class node is required in performing the class level filtering. The decrease in performance owing to the existence of a MO class node in performing scoping need not to be taken into consideration. The reason for this is that the additional pointer passing time can be negligible relative to the time which could be shortened as a result of the class level filtering. Generally, the attribute value comparison which occurs several times in the MO level filtering requires more time than the pointer passing. We make clear the concept of the class level filtering in the following section.

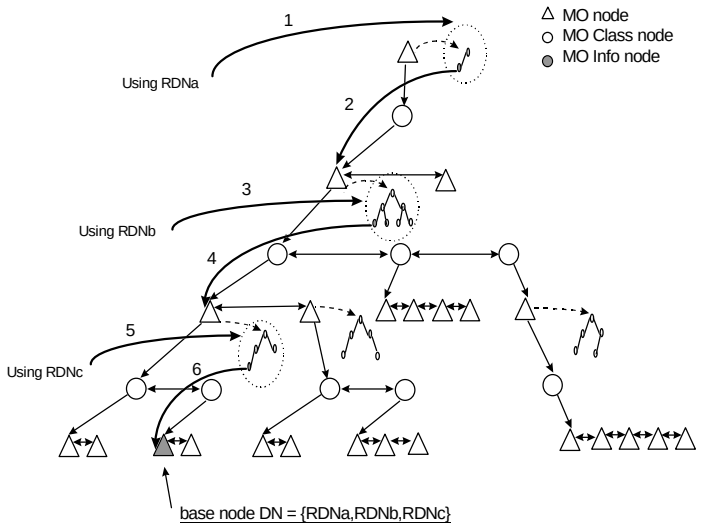

Figure 4. Searching mechanism in the MIT.

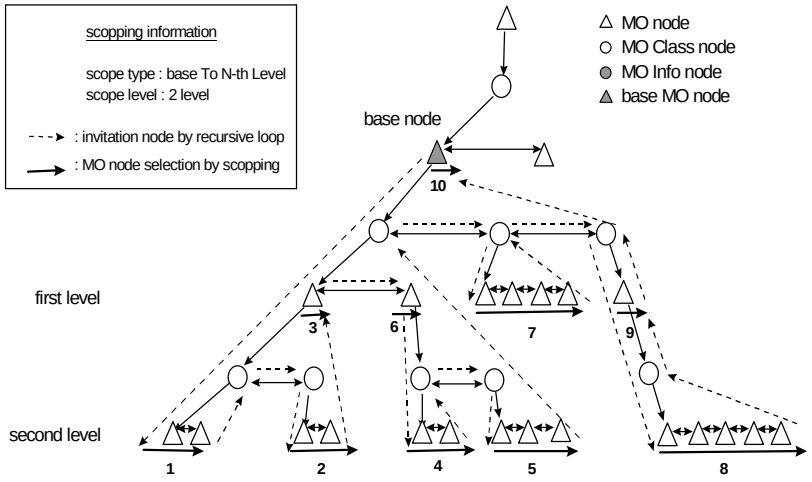

Figure 5. Scoping example in MIT.

3. Class Level Filtering

The filtering is a process which checks whether or not a management operation should be performed in an scoped MO instance. If the result is false, the management operation is not performed and vice versa. If the number of instances selected from the scoping is large and/or the filter is complex, the operation time

for the filtering largely affects the total time required to perform the management operation.

To overcome this shortcoming, we develop the concept of CLF. The MIT in our study is comprised of not only instances information, but also MO class information for the instances. The instances in the MIT are classified by MO class information. Therefore, it is possible to exclude the instances of improper class from being filtered using this class information.

Figure 6 shows the example of the CLF. Given the filter information as *(a=1 OR b>10) AND id=panzee*, if the *'a'* attribute is not contained in an MO class, we are able to cut down the filter applied to the instances of that MO class. That is to say, *(a=1 OR b>10) AND id=panzee* is reduced to *b>10 AND id=panzee*. If the *'id'* attribute is not contained in the MO class, the total filter expression becomes *FALSE*. Accordingly, the management operation need not be performed on the instances of that MO class. Consequently, unnecessary overhead for performing the filtering on improper MO instances are reduced. In section 4, we evaluate the CLF feature in various case.

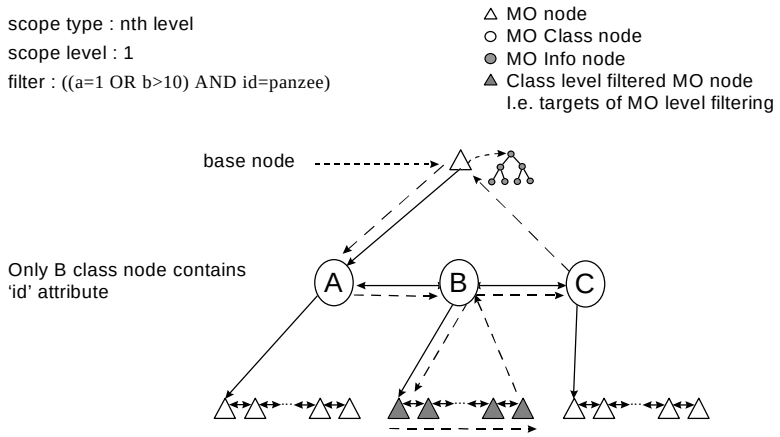

Figure 6. Example of Class Level Filtering.

In the worst case that a managing system frequently specifies the filter of which items, i.e. attributes, are included in most MO instances, in other words, all the results of most class level filtering are *TRUE*, the class level filtering is not preferable. This case is very rare in real world. In addition, this drawback can be overcome by the toggling class level filtering. We adopt a special MO instance to which a special action belongs. This action toggles the class level filtering in the runtime. It may be invoked by the M_ACTION management operation. Therefore, the choice of whether the class level filtering is to be performed or not is given to the manager. Consequently, the manager can toggle this feature by the special action, case by case.

If there are more than one manager connected to a managed system, toggling the CLF feature in runtime can be granted to each individual manager. To support this capability, the managed system manages the toggle information for each manager using association descriptors, which are obtained by ACSE stack in association establishing time. Receiving a management operation from a certain manager, the managed system decides whether or not the class level filtering should be performed.

4. Performance Evaluation

4.1 Analysis

Consider one MO Class node connected with *n* MO nodes. We think of p_c, t_c and t_{mi} as the probability of success at the CLF, the CLF time of the MO Class node and the MO level filtering time of *i*th MO node respectively. In the case of a filter succeeding in the CLF, the total filter time is:

$$t_f = t_c + \sum_{i=1}^{n} t_{mi}$$

In false, the total filter time is:

$$t_f = t_c$$

The average filtering time is:

$$t_f = p_c\left(t_c + \sum_{i=1}^{n} t_{mi}\right) + (1 - p_c)t_c = t_c + p_c \sum_{i=1}^{n} t_{mi}$$

Assumes that p_c and t_{mi} are independent. Accordingly the expectation of t_f is:

$$E[t_f] = E\left[t_c + p_c \sum_{i=1}^{n} t_{mi}\right] = E[t_c] + E[p_c] * E\left[\sum_{i=1}^{n} t_{mi}\right]$$
$$= E[t_c] + E[p_c] * nE[t_m] = E[t_c] + nE[p_c]E[t_m]$$

The filtering time $t_f^{/}$ of the case without CLF is:

$$t_f^{/} = \sum_{i=1}^{n} t_{mi}$$
$$E[t_f^{/}] = E\left[\sum_{i=1}^{n} t_{mi}\right] = nE[t_m]$$

We can compare the filtering time of the cases with and without CLF as follow:

$$\frac{E[t_f]}{E[t_f']} = \frac{E[t_c] + nE[p_c]E[t_m]}{nE[t_m]} = \frac{E[t_c]}{nE[t_m]} + E[p_c] \qquad \text{Equation (1)}$$

From Equation (1), it can be found that the ratio of $E[t_f]/E[t_f']$ consists of two parts: $E[t_c]/nE[t_m]$, which is caused by the CLF, and $E[p_c]$, which reflects the probability to perform the MO level filtering. As the number of MO nodes, i.e. n, increases, $E[t_c]/nE[t_m]$ decreases and consequently $E[t_f]/E[t_f']$ decreases. As $E[p_c]$ decreases, $E[t_f]/E[t_f']$ decreases. When the number of MO nodes is large enough, $E[t_c]/nE[t_m]$ tends to reach 0 and ratio $E[t_f]/E[t_f']$ tends to reach $E[p_c]$. For $E[p_c]$ <=1, we can conclude $E[t_f]/E[t_f'] <= 1$, $E[t_f]<=E[t_f']$.

The above discussion can be extended to the case that there are K Class nodes in whole scope area. When the CLF is used, the total filtering time t_f is:

$$t_f = \sum_{j=1}^{K}\left(t_{cj} + p_{cj}\sum_{i=1}^{n_j} t_{mji}\right)$$

Assumes that the p_{cj}, t_{mii} are independent. The expectation of t_f is:

$$E(t_f) = E\left(\sum_{J=1}^{k}\left(t_{cj} + p_{cj}\sum_{i=1}^{n_j} t_{mji}\right)\right) = KE[t_c] + KE[n]E[p_c]E[t_m]$$

The total filtering time of the case without the CLF is as follow:

$$t_f' = \sum_{j=1}^{K}\sum_{i=1}^{n_j} t_{mji}$$

$$E(t_f') = E\left(\sum_{J=1}^{k}\sum_{i=1}^{n_j} t_{mji}\right) = KE[n]E[t_m]$$

We can also compare the filtering time of the cases with and without the CLF through the ratio of $E[t_f]$ and $E[t_f']$:

$$\frac{E[t_f]}{E[t_f']} = \frac{KE[t_c] + KE[n]E[p_c]E[t_m]}{KE[n]E[t_m]} = \frac{E[t_c]}{E[n]E[t_m]} + E[p_c] \qquad \text{Equation (2)}$$

In Equation (2), we can find that $E[t_c]/E[n]E[t_m]$ is caused by the CLF. In the CLF, only the Object Identifiers(OIDs) of attributes are checked for deciding whether the attributes belong to the class or not. On the other hand, both the OID and the value of an attribute are checked in the MO level filtering. The time of an attribute value comparison is greater than that of OID comparison. So it can be considered that $E[t_c]$ is less than $E[t_m]$.

The filtering time of the MIT with the proposed mechanism should include both the CLF time and the MO level filtering time. It seems to be that extra load is caused by the CLF in the proposed mechanism. But, a portion of MO level filter is avoided owing to the CLF. All the MO level filtering under a MO Class node are avoided when the result of the CLF of the MO Class node is *FALSE*.

4.2 Performance Measurement

To evaluate the performance of the proposed MIT, at first, the execution time of the CLF in an MO Class node and the MO level filtering in an MO instance are measured receptively And then two MITs with and without the proposed mechanism are constructed using the same MIB. The scoping and filtering time of the two MITs are measured to evaluate the CLF algorithm.
The measurements of the CLF and MO level filtering are shown in table 1:

Table 1. The MO selection time comparison between CLF and MLF

Attribute Type	INTEGER	REAL	STRING	OID
Class Level Filtering (μs)	25	29	27	26
MO level Filtering (μs)	48	69	50	93

From the above result, it can be found that the CLF time is shorter than the MO level filtering time. In most cases, the CLF time is less than 1/2 of the MO level filtering time. The reason casing this result was described before. From the Table 1, we can also find that the MO level filtering time depends on the complexity of the syntax type of the attribute. For example, the MO level filtering time of OID is clearly longer than INTEGER.

The test MIB used in the measurement consists of 9 MO classes(including *system* MO class). The inheritance tree and naming rule of these MO classes are shown in Figure 7. The conditions for a measurement are as follow:

- The number of the instances of one MO class is one, five, ten, fifteen and twenty in each case.
- A base managed Object is set as *system* which positions highest in the MIT.
- A scope type is set as *wholesubtree.*

Following four filters are used receptively

- No filter is specified;
- Filter 1: *"NameBinding=System-top"*;
- Filter 2: "*testObjectName=test1"*;

- Filter 3: "*(testObjectName=pan) OR (testReal=4)*".

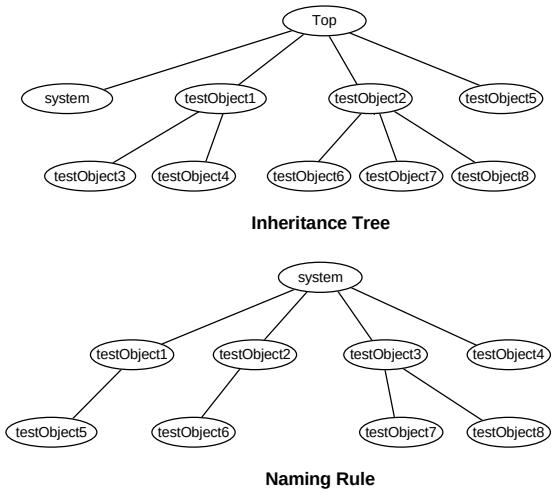

Figure 7. The inheritance tree and naming rule of test MIB.

The attributes *testObjectName* and *testReal* are defined in the MO class *testObject1*. The MO classes *testObject3*, *testObject4* are derived from *testObject1*. The measurement results of MO selection time with and without CLF are given as Figure 8, where micro second is used as the unit of the time.
For all cases, as the number of instance per MO class increases in both the cases with and without CLF, the measured time increases linearly When no filter is specified(Figure 8 [a]), in both cases with and without the CLF, the CLF and MO level filtering are not performed and only the scoping time is taken. Therefore the MO selection time is almost same in both cases. When filter 1 is specified(Figure 8 [b]), the MO selection time with CLF is little longer than the MO selection time without the CLF. Because *NameBinding* is defined in Top, for all MO classes the CLF result will be *TRUE* and the MO level filtering will be performed for all MO instances. This is the worst and very rare case. But, it can be found that the extra overhead caused by the CLF is small enough comparing with the MO selection time as shown in Figure 8 [b]. When filter 2, "*testObjectName=test1"*and filter 3, "*(testObjectName=pan) OR (testReal=4)*" are specified(Figure 8 [c], [d]), the MO selection time with the CLF is clearly shorter than the MO selection time without the CLF. In these cases, the CLF results will be *TRUE* for only *testObject1*, *testObject3* and *testObject4* MO classes, and the MO level filtering will be performed for only the MO instances of *testObject1*, *testObject3* and *testObject4*. Since the MO level filtering is avoided for all of the instances of other MO classes, the MO selection time is distinctly reduced as depicted in Figure 8 [b].

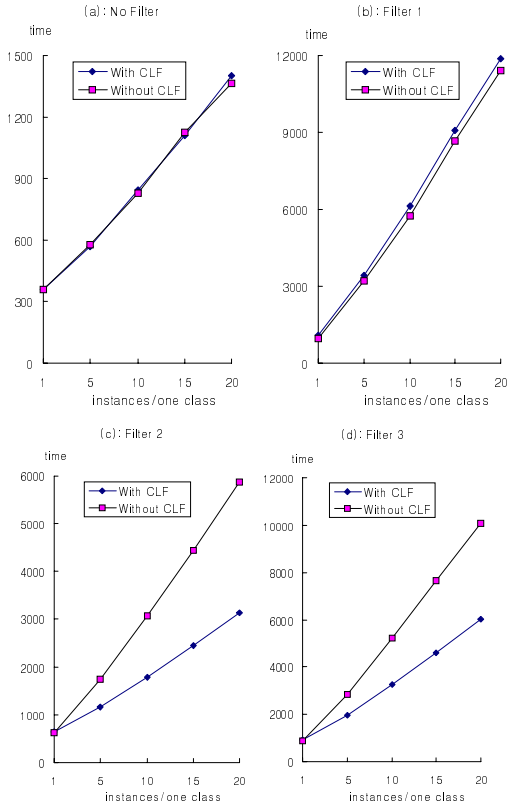

Figure 8. The MO selection time comparison.

5. Conclusion

In this paper, we have proposed a Management Information Tree(MIT) architecture supporting efficient MO selection. In addition, we have developed a new mechanism for MO selection in our MIT, named as CLF. For various cases, we discussed how it could be used to improve the performance of MO selection in an MIT. Through extensive analysis and performance test in this paper, we have shown that our MIT and CLF can significantly reduce the MO selection delay.

References

[1]. CCITT, M.3000 series recommendations, CCITT, 1992
[2]. CCITT, X.700 series recommendations, CCITT, 1992
[3]. Iosif G. Ghetie, “Network and System Management Platform analysis and

Evaluation", Kluwer Academic Publishers, 1997
[4]. Salah Aidarous, Thomas Plevyak, "Telecommunications Network Management Technologies and Implementations", IEEE Series on Network Management, 1997
[5]. Morris Sloman, et al., "Network and Distributed Systems Management", Addison-Wesley Publishing Company, 1994
[6]. George Pavlou, "The OSIMIS Platform:Making OSI Management Simple", Proceedings of the fourth international symposium on integrated network management, 1995
[7]. M. Feridun, L. Heusler and R. Nielsen, "Implementing OSI Agent/Managers for TMN", IEEE Communications Magazine, September, p. 62-67, 1996
[8]. Toshio Tonouchi, Takashi Fukushima, Asuka Manki, Shin Nakajima, "An Implementation of OSI Management Q3 Agent Platform for Subscriber Networks", Proceeedings of the 1997 IEEE International Conference on Communications - Volume 2/3, p. 889-893, 1997
[9]. Grady Booch, "Object-Oriented Analysis and Design", Addison-Wesley Publishing, 1996