Distributed Intelligent Monitoring and Reporting Facilities

George Pavlou[1], George Mykoniatis[2], Jorge-A. Sanchez-P.[2]

Abstract: Distributed intelligent monitoring and reporting facilities are of paramount importance in both service and network management as they provide the capability to monitor quality of service and utilisation parameters and notify degradation so that corrective action can be taken. By intelligent, we refer to the capability of performing the monitoring tasks in a way that has the smallest possible impact on the managed network, facilitates the observation and summarisation of information according to a number of criteria and in its most advanced form, permits the specification of these criteria dynamically to suit the particular policy in hand. In addition, intelligent monitoring facilities should minimise the design and implementation effort involved in such activities. The ISO/ITU Metric, Summarisation and Performance management functions provide models that only partially satisfy the above requirements. This paper describes our extensions to the proposed models to support further capabilities, with the intention to eventually lead to fully dynamically defined monitoring policies. The concept of distributing intelligence is also discussed, including the consideration of security issues and the applicability of the model in ODP-based distributed processing environments.

Keywords: Management, Monitoring, Reporting, OSI, TMN, ODP

1. Introduction

Managing Quality of Service (QoS) is of paramount importance in both current and future service architectures operating over underlying broadband technologies such as Synchronous Digital Hierarchy (SDH), Asynchronous Transfer Mode (ATM) etc. Engineering considerations such as minimising the impact of management systems on the managed network and reacting in a timely fashion to performance degradation need to be supported by relevant computational and information models. In the Telecommunications Management Network (TMN) framework [1], Open Systems Interconnection (OSI) Management principles [2] are used to structure and access management information in an object-oriented fashion. The need for generic information models supporting distributed intelligent monitoring activities was recognised long ago, resulting in the Metric Monitoring [7] and Summarisation [8] function specifications.

These functions distribute essential intelligence close to the managed resources, reducing the amount of management traffic and providing support for a sophisticated event-driven operation paradigm. Though of undoubted usefulness, it soon becomes apparent that the intelligent monitoring policies they support are predefined. A slightly different policy than the ones supported needs to be specified in object-oriented terms, realised and then fielded in systems supporting such facilities as pre-compiled logic. Ideally, fully flexible generic facilities are necessary, where new policies can be realised dynamically, without the need to alter anything at the managed end of the spectrum.

Having realised these limitations, in the context of the RACE-II Integrated Communications Management (ICM), TOMQAT and ESPRIT-III IDSM projects *(the last two acronyms will be expanded)* we have carried out research leading in the provision of such advanced facilities. This research culminated in the specification of the Intelligent Monitoring Function (IMF) [11] and its object-oriented realisation as part of the OSI Management Information Service (OSIMIS) TMN platform [13]. In this paper, we explain the relevant issues and operational models and present examples to show the broad applicability of these concepts. We also discuss issues related to dynamic intelligence, security and potential mappings on platforms based on the Common Object Request Broker Architecture (CORBA) [10]. We finally refer to the state of the art in the distribution of management intelligence and position our approach in that paradigm.

The rest of the paper has the following structure. We first look briefly at the OSI management model,

1 University College London, Comp. Science Dept., Gower Str., London WC1E 6BT, United Kingdom
2 National Technical University of Athens, Elec. and Comp. Engineering Dept., 9 Heroon Polytechniou St., Zographou, Athens, Greece.

which provides the underlying framework on which our approach is based. We then discuss the current state of the art in intelligent monitoring in the context of OSI management and present our own intelligent monitoring model, which is an extension of the former. A number of examples showing the applicability of our model are then presented, explaining the associated benefits. We finally position our model in the "management by delegation" paradigm, consider security issues and examine potential mappings on CORBA.

2. The OSI Management Model

OSI Management [2] projects a fully object-oriented model, with applications in agent roles "exporting" Managed Objects (MOs) that encapsulate managed resources at various levels of abstraction. Applications in manager roles access these objects in order to implement management policies. Managed objects are formally specified using the Guidelines for the Definition of Managed Objects (GDMO) [6], which is an object-oriented information specification language with management orientation. The associated access service, the Common Management Information Services (CMIS) [5], has essentially "remote method call" semantics. The OSI manager-agent model is shown on Fig,. 1.

Managed objects may exist in managed network elements, end-systems and distributed applications. They may also exist in management applications that act in agent roles. The distinction between manager and agent roles for a management application serves only the purpose of the model and is not strong in engineering terms: a management application may be in both roles. This is in fact the norm in a hierarchical layered architecture such as the one projected by the TMN [1].

A key difference between the OSI management and Open Distributed Processing (ODP) [4] object-oriented frameworks is that OSI managed objects always come in "clusters" or "ensembles". Each managed element or management application that acts in agent role exports one such cluster; this is also referred to as the Management Information Base (MIB). Managed objects in those clusters have typically many relationships but containment is treated as a primary one that yields unique hierarchical names. A management application may contain other computational entities in addition to the managed objects e.g. managing objects. These are not visible externally and are considered an internal design and implementation issue; this is why managing objects are depicted with dotted-line boundaries in Fig. 1. In the manager-agent model projected by OSI management, only managed objects are specified as these provide the means of open communication between management applications and managed entities.

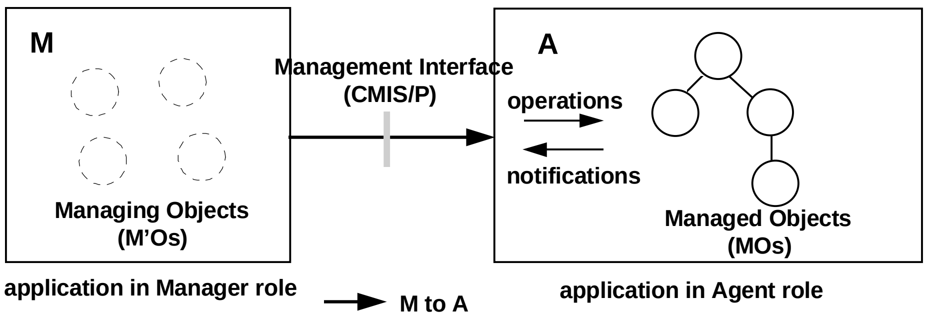

M
Managing Objects (M'Os)
Management Interface (CMIS/P)
A
operations
notifications
Managed Objects (MOs)
application in Manager role
M to A
application in Agent role

Fig. 1. The OSI Manager - Agent Model

OSI management addresses the open communication between applications in manager and agent roles. It can be extended though with distribution aspects through the OSI Directory [3]. Unlike ODP, where every object may be distributed, in the extended OSI management model the unit of distribution is the managed object cluster i.e. the management application. The OSI Directory provides a hierarchical distributed object-oriented database which is updated by management applications that register their location and capabilities so that various transparencies may be realised. In this extended model, managed objects may be addressed through global names. A global name consists of a global part, which identifies the logical “cluster” or application they belong to; and of a local part, which identifies the managed object within that cluster. Global names may be used to provide access to managed objects in a location transparent fashion.

Managed objects in a cluster may be accessed either individually or collectively, through an object-oriented database query facility. Many objects may be selected by traversing containment relationships, the facility known as *scoping*. The selection may be further refined by specifying a boolean expression to be evaluated, containing assertions on attribute values, which is known as *filtering*. Scoping and filtering may be thought as a limited version of the ODP trading service. The key difference though is they are tightly coupled with a managed object cluster and this minimises the traffic that would be otherwise required to access a separate trader and then the managed objects one by one. An example using this query facility could be “retrieve all the attributes of the routing table entries that point to the next hop X”. This may be expressed by scoping all the objects contained by a route table object and using a filter that asserts the next hop address attribute to be X. One CMIS request is sent and possibly a number of replies are returned. Scoping and filtering are very powerful and are typically exercised across a management interface. In addition, managed object attributes may have scope and filter syntax and semantics. In this case, their evaluation depends on the managed object behaviour. This capability is used by the Intelligent Monitoring Function.

OSI Management is a communications concept and, as such, it is object-oriented only in terms of information specification and access. Managed objects are accessible across management interfaces but the internal structure of the communicating applications is not dictated and may not be object-oriented. Platform infrastructures such as OSIMIS [13] have shown how such an object-oriented information specification may be mapped onto a fully object-oriented engineering framework, providing high-level Application Programming Interfaces (APIs) and various transparencies [12]. The Network Management Forum (NMF) is currently looking to standardise such object-oriented APIs.

3. Performance Monitoring OSI Systems Management Functions

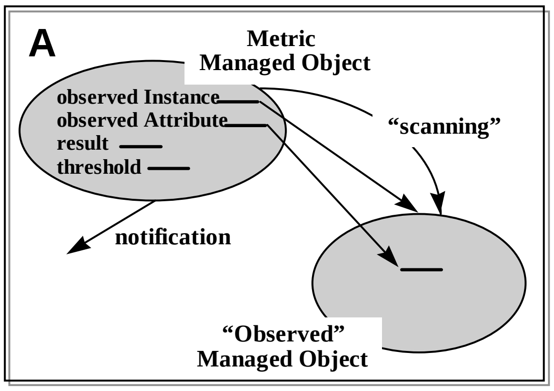
While the manager-agent model does not in itself restrict the amount of intelligence that may be specified and realised by managed objects, most standard specifications concentrate on providing a rich enough set of attributes and actions which model information and possible interactions with the underlying resource. Notifications are also provided to report significant exceptions but they are usually generic ones e.g. object creation / deletion and attribute value change.

The OSI Structure of Management Information (SMI) specifies generic attribute types such as counter, gauge, threshold and tide-mark. Gauges model entities with associated semantics, e.g. number of calls, users, etc. or the rate of change of associated counters e.g. bytes per second etc. Thresholds and tide-marks may be applied to gauges and generate QoS alarms and also attribute value changes, indicating change of the high or low “water mark”. Such activities are of a managing nature, in fact in the paradigm projected by the Simple Network Management Protocol (SNMP) [19] they are solely performed by managing systems.

Although thresholding functions could be made part of managed objects modelling real resources (in fact, there have been such early managed object specifications), it does not take long to recognise their genericity. As such, they should better be separate generic functions so that they become reusable. The relevant International Standards Organisation (ISO) / International Telecommunications Union (ITU) group standardising OSI management recognised the importance of generic monitoring facilities and standardised the Metric Monitoring [7], Summarisation [8] Systems Management Functions and the TMN Interface Performance Management Function [16]. By making such functions generic, it is possible to implement them once and make them part of the associated platform infrastructure.

The whole idea behind the Metric Monitoring Function [7] and the objects realising its functionality (metric objects) is to provide thresholding facilities in a generic fashion. Monitor metric objects may be instantiated in an application in agent role and be configured to monitor, at periodic intervals, an attribute of another managed object. The managed object may be the abstraction of a physical or logical aspect of a network or service element. The observed attribute should be a counter or gauge and the metric object either observes it as is or converts the observed value to a rate over time. Statistical smoothing of the observed value is also possible if desired.

The main importance of this facility is the attachment of gauge thresholds and tide-marks to the resulting derived gauge which may generate QoS alarms and indicate the high and/or low "water mark". In fact, the metric objects essentially enhance the "raw" information model of the observed object. The metric monitor functionality is depicted in Fig. 1. and can be summarised as:



A: Application in agent role ("exporting" a management interfac
——: Attribute

Fig. 1. The metric object model

- ***data capture****: through observation or "scanning" of a managed object attribute,*
- ***data conversion****: potential conversion of a counter or gauge to a derived gauge,*
- ***data enhancement****: potential statistical smoothing of the derived result,*
- ***notification generation****: QoS alarm and attribute value change notifications.*

Gauge thresholds are always specified in pairs of values: a triggering and a cancelling threshold. The former will generate a notification when crossed only if the latter has been previously crossed in the opposite direction. This prevents the continuous generation of notifications when the measured value oscillates around the triggering threshold and is known as the hysteresis mechanism. Fig. 2 shows both high (e.g. "over-utilisation") and low thresholds (e.g. "under-utilisation") applied to the observed value. Typically, a normalised value of around 5% is used as the hysteresis interval.

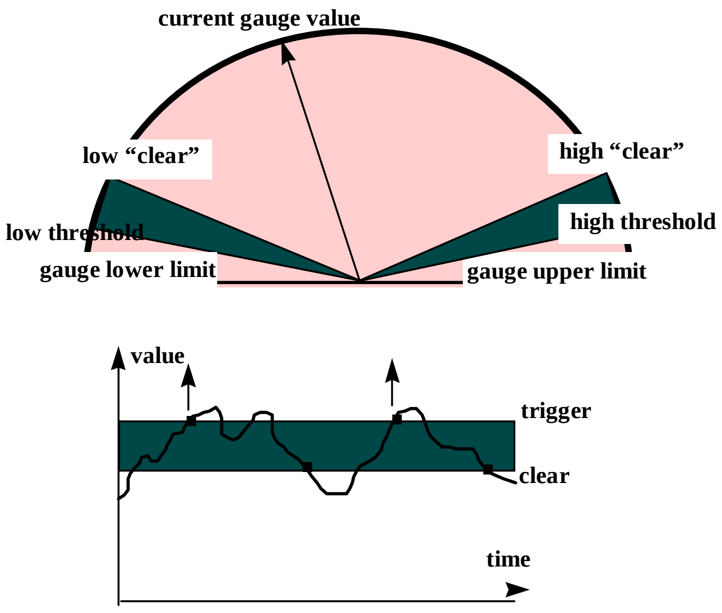


Fig. 2. Thresholding with hysteresis

The Metric objects offer the OSI management power through event reporting and logging, even if the "raw" observed management information model does not support such notifications. More importantly, they obviate the use of rates, thresholds and tide-marks in a way tied to specific managed objects but they allow the same flexibility and power dynamically, whenever a managing system needs it. Such a monitoring facility reduces the management traffic between applications and their impact on the managed network by supporting an event-based operation paradigm. It should be emphasised that these facilities are of "managing" nature but offered within a managed object cluster across a management interface (see Fig. 1). Finally it should be mentioned that similar facilities have been adopted in the SNMP framework as part of the Remote Monitoring (RMON) and Manager-to-Manager (M2M) MIB specifications. These are partial translations of the OSI Metric Objects [7].

The Summarisation Function [8] and the objects realising its functionality (summarisation objects) extend the idea of monitoring a single attribute to monitoring many attributes across a number of selected managed objects. They offer similar but complementary facilities to metric objects. In this case, there are no comparisons / thresholding but only the potential statistical smoothing and simple algorithmic results of the observed values (min., max., mean and variance). These are reported periodically to the interested managing systems through emitted notifications. The observed managed objects and attributes can be specified either by supplying explicitly their names or through CMIS scoping and filtering. The observed values may be raw ones, modeling an underlying resource, or enhanced values as observed by metric objects. They can be reported either at every observation period or after a number of observation periods (buffered scanning).

The major importance of this facility is that a number of values a managing system needs can be specified once and then reported as mentioned, without the managing system having to send complex CMIS queries periodically or having to perform the statistical smoothing. Intelligence in this context has to do with the periodic scanning and reporting of diverse information automatically, after the criteria for its assembly have been specified once. Such criteria may be expressed through CMIS scoping and filtering, providing a lot of flexibility in summarising information of dynamic nature (e.g. traffic across certain ATM virtual path or virtual channel connections, etc.) Any other complex computations on the summarised information are left to be performed by the managing system.

4. Advanced Intelligent Monitoring Facilities

4.1. Rationale

While the Metric Monitoring and Summarisation facilities can be combined to provide a lot of flexibility, they still leave the task of complex calculations and comparisons to managing systems. In particular, in most cases it is not one attribute value that needs to be monitored and compared to thresholds but the combination of more than one attribute, possibly across different managed objects. The derived value is thus the result of the application of a mathematical operation on the observed values. Such an operation could be: sum (a1+...+an), division (a/b), difference (a-b), etc. The combination of more than one operation may be used to construct arbitrary expressions, albeit at the cost of multiple monitoring objects.

In short, a facility is needed that combines the properties of the metric (thresholding on a derived result) and summarisation objects (observation of arbitrary objects and attributes). This combination is effected through a mathematical operation that is performed on the observed values to yield the derived result. We have recognised the need for such a facility early on and have specified and implemented a number of GDMO classes that provide this functionality, termed collectively Intelligent Monitors (IMs) [11]. Since then, the relevant ISO/ITU working groups have also recognised this need and provided amendments to the Metric and Summarisation Systems Management Functions and produced the TMN Interface Performance Management Recommendation [16], offering similar functionality.

In most cases, simple mathematical calculations are enough to express real-world problems. For example, if connection acceptance and rejection counts are kept, the connection rejection ratio is given by the simple formula below:

$$connection\_rejection\_ratio = \frac{\sum connections\_rejected}{\sum connections\_accepted + \sum connections\_rejected}$$

In a similar fashion, if PDU rejected and sent counts are kept, the error rate across that transmission path is given by their ratio as below:

$$transmission\_error\_rate = \frac{\sum PDUs\_rejected}{\sum PDUs\_sent}$$

Such arithmetic operations can be either specified statically, through a relevant attribute of the summarisation object, or specified dynamically by combining the observed attribute values with operands. In the latter case, ultimate flexibility is provided in applying arbitrary mathematical operations.

4.2. Model

The underlying concepts for the Intelligent Monitoring Function are based on the combination of the Metric Monitoring and Summarisation Functions. The IMF provides for:

- *the ability to monitor information provided by single or multiple attribute types,*
- *the selection of the observed attribute values either explicitly, through the names of the containing objects, or through scoping and filtering,*
- *the identification of an operation (e.g. sum) to be applied on the observed attribute values,*
- *the identification of an algorithm (e.g. uniform weighted moving average) used to smooth the derived result,*
- *the ability to emit notifications when the derived result crosses a threshold or "pushes" a tide-mark or when any of its attributes change.*

The model for the operation of the IM is similar to that of summarisation and is shown in Fig. 3. Information is obtained by observing attributes of other objects, including managed objects representing a management view of an underlying network or service resource, metric objects, or other IMs. The attributes of those "observable" objects are accessed at the object boundary, triggering associated behaviour in the same fashion as across a management interface. The selection of the observed objects and their attributes, the identification of the mathematical operation and the identification of the statistical smoothing algorithm to be applied are effected through attributes of the IM that are set at creation or at any other time. In the latter case, the operation of the IM object should be suspended temporarily for its operational parameters to be changed. Summarising, in engineering terms the pre-compiled logic of the intelligent monitor is parametrised through its attributes (operational parameters) which can be manipulated across a management interface.

A particularly useful type of IM is the Probability Estimator (see Section 5.4 for an example application). This is a buffered homogeneous object able to scan one observed attribute and calculate the probability that its value is greater than a pre-set threshold:

$$\Pr[A > T] = \frac{number\_of\_appearances\_of\_the\_condition[A > T]}{number\_of\_observations}$$

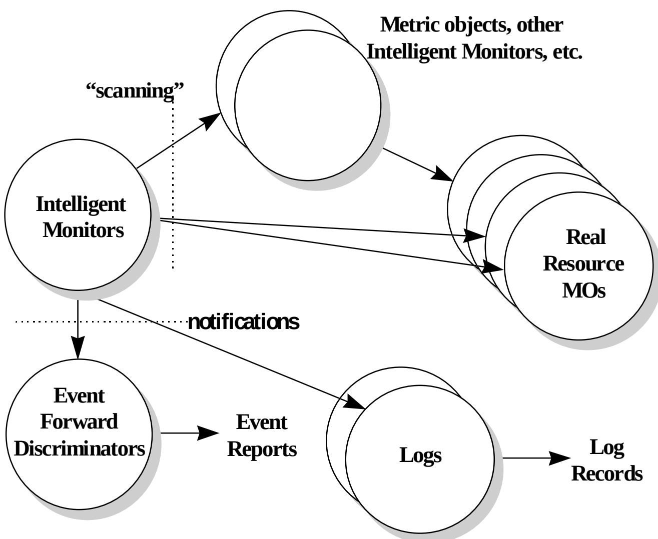


Fig. 3. Intelligent Monitors model

While there is a lot of flexibility in specifying attributes and objects to be observed, either explicitly or through scoping and filtering, the full set of available mathematical operations is statically specified and realised when the IMs are implemented using relevant platform infrastructure. The introduction of a new unforeseen mathematical operation implies new specification and subsequent realisation. An additional drawback is that a complex mathematical operation needs to be decomposed to simple generic ones that are supported, necessitating the existence of more than one IMs. This can pose a scalability problem regarding activities at entities with 10's of thousands managed objects (e.g. a mediation function for an access network).

The above observation has led us to research the possibility for specifying dynamically the operation to be applied on the observed attributes. This is possible by merging the relevant operations and operands with the observed attributes in a single attribute with a complex, recursive, tree-like syntax (see Appendix). ASN.1 is very powerful and particularly suitable for such complex representations. The notion of "global attributes" is also introduced as a tuple that consists of a global object name and an attribute type. Through this type of object, it is possible to specify dynamically any arbitrary mathematical operation as dictated by the management policy, without the need for any additional specification or implementation. The drawback is that the observed "global attributes" need to be explicitly specified since they are referenced in the mathematical operation i.e. the flexibility of using scoping and filtering to identify the observed attributes dynamically is lost. This can be partly compensated by using these in conjunction with the previous types of IM objects.

Since managed object names can be global, IM objects can also be used for the intelligent summarisation of information which exists logically or physically at different clusters and locations. In this case though, the advantage of monitoring within the same node and thus reducing management traffic is lost. A new possibility that emerges is the inclusion of further intelligence so that a top-level complex operation is broken down to sub-parts based on observed values within the same node or domain. Each such sub-part will be delegated to a subordinate IM object which will be created in that domain. All the buffered scanning and smoothing will take place locally while resulting values will be reported back to the "master" IM to produce the final value, perform comparisons and trigger notifications. The result of using such a facility is effectively "automatic decomposition and downloading" of management intelligence as close to the managed elements as possible, reducing the overhead of management traffic and increasing the observed information timeliness by reducing network latency. The mapping of managed object names to nodes or domains could be through the Directory [3]. We have not yet addressed this type of IMs but we consider this a future possibility for further research.

4.3. Realisation

Implementing OSI management based information specifications can be a daunting proposition without suitable supporting infrastructure. In our case we have implemented standard Metric, Summarisation and IM objects using the OSIMIS object-oriented TMN platform [13]. In OSIMIS, GDMO information objects become C++ engineering objects. A GDMO/ASN.1 compiler was used to produce stub managed object infrastructure, which hides completely all access details, leaving only behavioural aspects to be implemented.

An important platform aspect for realising such objects is the ability to evaluate scoping and filtering locally, to address managed objects by name and to access their attributes at the object boundary. OSIMIS was designed with flexibility in mind and these functions are supported through easy to use APIs. Finally, accessing objects can be through the same API regardless of location, offering useful transparency services. The resulting IMF implementation has become an integral part of the OSIMIS platform. Metric and intelligent monitor objects exist as pre-compiled knowledge at every application in agent role, ready to be instantiated by managing systems.

Metric and intelligent monitors follow the compiled model, being implemented in C++. Their logic and behaviour is parametrised through their attributes, which can be manipulated across the management interface. Very complex mathematical expressions can be achieved through the attribute syntax described in the Appendix. The fact that the compiled model is followed is in contrast with the full management by delegation paradigm discussed in section 6, which typically requires interpreted languages such as TCL, Scheme etc. The advantage of the compiled model is increased performance, while the interpreted approach offers ultimate flexibility.

4.4. Management Policies

The OSI Metric Monitoring, Summarisation and Event Reporting Functions together with the IMF provide the means to distribute management services and to realise intelligent agents in remote networks instead of performing such activities in centralised managing systems, acquiring the required values by polling.

Direct usage of the Systems Management Functions requires the cumbersome and error-prone creation and initialisation of a set of managed object instances. A Monitoring Policy [17], [18] is necessary to allow the network operator to describe a monitoring task in terms of a monitoring formula. An associated Enforcement Service is then able to parse the monitoring formula and create the necessary instances.

If thresholds are defined within a Monitoring Policy, an appropriate Event Reporting Policy [17], [18] could be set up. Event reporting policies describe the event flow between applications in agent and manager roles (which notification from which managed object instances has to be reported to which manager). Finally an Event Reporting Policy Enforcement service could create and configure the respective Event Forwarding Discriminator objects that will realise the desired reporting strategy.

5. Applicability Examples

We present here three applicability examples, showing the benefits of the proposed Intelligent Monitoring model in terms of reduced management traffic and timeliness of observed information. The first two are monitoring and network utilisation examples for an Fiber Distributed Data Interface (FDDI) Metropolitan Area Network (MAN) and an Asynchronous Transfer Mode (ATM) Wide Area Network (WAN). The third example addresses a performance verification service in the context of a TMN system performing Virtual Path Connection and Routing Management (VPCM) of an ATM WAN. The first and third examples have been designed and implemented in the context of the RACE-II ICM project while the second one in the context of the RACE-II TOMQAT project.

5.1. FDDI Network Monitoring

As a first example, we consider a performance monitoring case study for a FDDI network. The management policy in this case is to continuously monitor the utilisation of the network and generate both early warnings and “red light” QoS alarms, indicating levels of congestion. In addition, we would like to keep the history of network utilisation over periods of time and, in particular, the highest congestion levels reached over those observation periods.

A FDDI network has a ring topology and packets are injected in the ring at a particular source node, to be subsequently forwarded across the ring by other nodes until they reach the specified destination node. Usually FDDI rings are used as backbones, carrying traffic of Local Area Networks (LANs) that are attached to them. Most FDDI rings come with SNMP-based management interfaces at every node. The latter support attributes related to the activity of every physical interface of that node; e.g. the number of octets received and sent through the two FDDI interfaces of each node. This can be considered as "raw" information available to managing systems for data conversion, enhancement, thresholding, history, etc.

In SNMP-based management systems, all these functions take place in centralised management stations which retrieve this information periodically across the network. This mode of operation is in accordance with the SNMP fundamental axiom of physically separating managing and managed functionality in order to keep network elements simple. Given the fact that SNMP agents do not support sophisticated intelligent monitoring facilities, there is not much that can be done to "push" intelligence to the network element level. This intelligence has to be deployed instead at the next level of the management hierarchy, which has to be kept as close to the managed elements as possible. In our case study, this is an adaptation / mediation layer which is conceptually part of the element management layer of a TMN [1] hierarchy. We thus operate a generic CMIS/P to SNMP application gateway or Q-Adaptor Function (QAF) [14], which provides automatically an OSI view of SNMP managed objects. In engineering terms, one such device will adapt for all the FDDI nodes, being both a TMN Q-Adaptor (QA) and a Mediation Device (MD). This should operate at one of the FDDI nodes, keeping all the monitoring traffic in the FDDI domain.

Averaging the throughput across the FDDI ring is a function that can easily be provided by the IMs. An instance of such an object is created within the CMIS/P-SNMP QA/MD application with the task to monitor locally the FDDI interface objects, calculate the mean, convert it to a throughput gauge, smooth it statistically and compare it to threshold values specifying early warning and congestion conditions. In addition, the highest and lowest throughput levels will be recorded through relevant tide-mark attributes. Note that monitoring requests within the QA/MD will be translated and forwarded to the FDDI ring across the SNMP interface of the nodes. Despite this, the polling traffic will be limited in the FDDI domain while the result of the IM will be presented to higher-level functions in an event-driven fashion, through the emission of QoS alarm and attribute value change notifications. Event forwarding discriminator and possibly log objects will have to be created to disseminate notifications or log them locally. Table 1 shows the attributes of the monitoring object set by the managing object or application to support this function.

Note that the interface objects containing the attributes to be monitored are specified using the scoping and filtering facilities. That way, exact knowledge of the FDDI ring topology in terms of its number of nodes is not necessary. An alternative, less flexible way to specify those objects could be through a managed object instance / attribute list. Note also that the observed value is statistically smoothed after the calculation in order to reduce problems with rapid fluctuations. Finally, the threshold values are shown normalised: 85% and 95% are the early warning and red-light values respectively, with 5% hysteresis each. In reality, these would have to be specified based on the units of the measured traffic (octets) and the maximum speed of the network (100 Mbits / sec).

Attribute	*Value*
scannerId	fddi-thput
baseManagedObject	{} (the top MIT object)
scope	3rdLevel (interface entries are there)
filter	(objClass=interfaceEntry & ifType=fddi)
attributeId	ifOutOctets
granularityPeriod	10 (secs)
operation	mean (of observed values)
algorithmType	EWMA (exponentially weighted moving avg)
estimateOfMeanThld	{ Low=0.8 Switch=Off High=0.85 Switch=On, Low=0.9 Switch=Off High=0.95 Switch=On }

Table 1

The class of monitoring object used above supported a number of operations to be applied on the observed values, from which the mean was used. As we already described, the limitation of this approach lies in the fact that the number of supported operations are finite and pre-defined. Our case study was served well by the existing operations, but it would be nice to be able to supply a coefficient to be applied to the mean so that threshold values can be normalised.

Bearing in mind we are observing bytes / sec, the coefficient for the 10 Mbit/sec FDDI should be $(8/100*10^6)$. For example, if our FDDI has four nodes and c is the above coefficient, the observed value is given by the expression in Fig. 4.

In this case, the IM specifies through one attribute both the observed object instance and the operation to be applied to the observed values. Note that the absence of scoping and filtering implies knowledge of the managed object name modelling the FDDI interface at each node. This can be discovered through a get operation with scope and filter before the monitoring object is created.

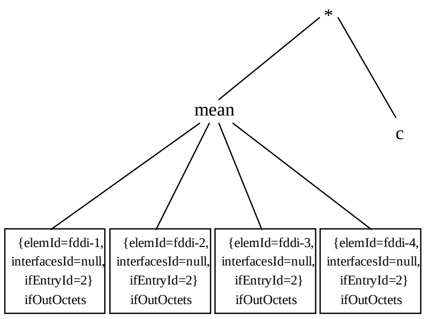

*
mean
c
{elemId=fddi-1, interfacesId=null, ifEntryId=2} ifOutOctets
{elemId=fddi-2, interfacesId=null, ifEntryId=2} ifOutOctets
{elemId=fddi-3, interfacesId=null, ifEntryId=2} ifOutOctets
{elemId=fddi-4, interfacesId=null, ifEntryId=2} ifOutOctets

Fig. 4. Expression with coefficient

Summarising, the functionality described for the FDDI case study is a matter of deciding on the management policy and instantiating the type of intelligent monitoring object suitable for the policy in hand. Semantic-free (generic) applications such as browsers, event monitors, loggers etc. may be used to receive the QoS alarms or to retrieve and plot history data and trends. In short, it may be a matter of minutes (or hours at most) to implement sophisticated monitoring policies, as opposed to the weeks (or months) that would be required otherwise for the full development cycle.

5.2. ATM Network Monitoring

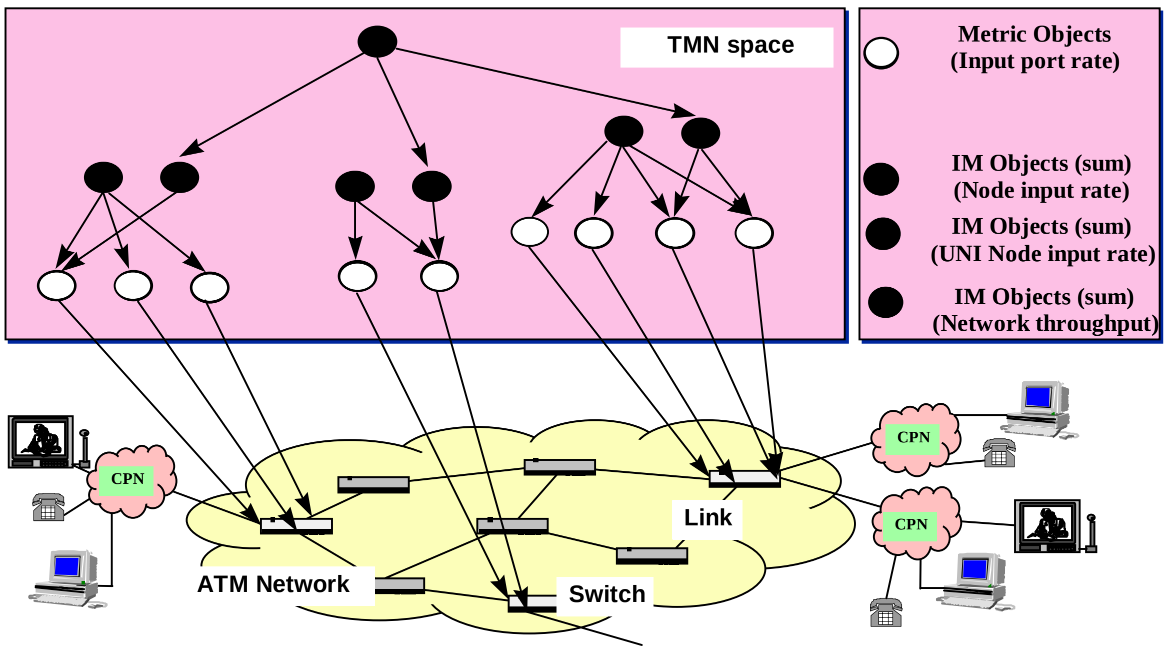

TMN space
Metric Objects (Input port rate)
IM Objects (sum) (Node input rate)
IM Objects (sum) (UNI Node input rate)
IM Objects (sum) (Network throughput)
CPN
CPN
CPN
Link
ATM Network
Switch

Fig. 5. ATM intelligent monitoring

In this example, we consider the monitoring activity of an ATM-based Broadband Integrated Services Digital Network. Specifically, only one "raw" attribute (e.g. numberOfCellsReceived in an ATM port), available at switches with management interfaces, is used to compute the following:

- *link / port throughput (rate) and utilisation,*
- *node input rates, including the User-to-Network (UNI) and Network-to-Network (NNI) input rate,*
- *network throughput.*

As a first step, metric objects should be instantiated at each switch agent for the computation of each link throughput (numberOfCells over time). There should be one such metric object for each link at each node. The overall input rate for each node can be calculated by another IM, observing all the link input rates and adding them up (sum operation). At each User to Network Interface (UNI) node, another intelligent monitor can calculate the UNI node input rate by observing and adding the rates of the input links. Finally, the overall network throughput can be calculated by a top-level IM, observing all the IMs that calculate User to Network Interface (UNI) node input rates and adding them up. The relevant hierarchy of monitoring objects is shown in Fig. 6. Note that the first two levels of metric and intelligent monitors are instantiated in the switch agents, so the effect of their monitoring activity has no impact to the managed network. The top level intelligent monitor calculating the network throughput is instantiated in a network-wide performance management application e.g. a TMN performance management Operation System (OS).

Instantiating all these intelligent monitoring objects can be done in a way that does not need pre-defined knowledge of the network topology but "discovers" it from management information in a topology related management application. Instantiating the intelligent monitoring objects can be done, for example, using an interpreted access API such as the Tcl-CMIS offered by the OSIMIS platform or by using the policies described in section 5.2. A small script e.g. 50-100 lines of interpreted object-oriented logic is enough to start-up the above monitoring function, supported subsequently by generic tools such as browsers, event monitors, etc.

5.3. Performance Verification in Broadband Networks

In the last example, a management service for network Performance Verification (PV) is considered [20]. The managed environment is assumed to be a public ATM network supporting a wide range of services made up of network bearer service classes characterized by the relevant Class of Service (CoS). Evaluation and verification of network performance is done on the basis of measurements concerning the performance of the CoSs the network supports; e.g. rejection ratio (blocking probability), jitter, cell delay, cell loss ratio, etc. For example, the rejection ratio parameter is calculated from the following formula:

$$rejection\_ratio = \frac{rejection\_rate}{rejection\_rate + accept\_rate}$$

The rejection and acceptance rates are computed from the relative counters of the rejected and the accepted connections, which are available at the network elements. The probability of the rejection ratio to be greater than a certain value can be measured by the following expression:

$$\Pr[rejection\_ratio > Threshold] = \frac{number\_of\_appearances\_of\_the\_condition\left[rejection\_ratio > Threshold\right]}{number\_of\_total\_observations}$$

In order to measure the above probability, the following Metric and IM objects need to be instantiated:

- a Metric object for the calculation of the Acceptance Rate,
- a Metric object for the calculation of the total Rejection Rate,
- an IM to compute the rejection ratio from the Acceptance and Rejection Rate values,

- A probability estimator IM that will scan the observed attribute (i.e. the rejection ratio) at periodic intervals and store the values in a buffer. At every reporting period an operation will be applied to the values that have already been stored at the buffer.

Based on the measurements, the PV service evaluates network performance and, by comparing it with the maximum allowable values (performance targets), verifies whether it is within acceptable levels or not. The PV service is realised by a relevant Operation Systems Function (OSF) in a TMN system. The interactions of the PV OSF with other OSFs is shown on Fig. 7.

The PV OSF is located at the Network Management Layer of the TMN layered hierarchy. When it requests a particular performance measure, a monitoring activity is created in the Current Load Model (CLM) OSF, also located at Network Management Layer, which is the coordinator for the network monitoring activities. The latter delegates element level monitoring activities to the Network Elements Functions (NEF), where the raw data is located, or to Mediation (MF) and Q-Adaptor Functions (QAF) depending on implementation constraints i.e. in the case that the elements do not support fully capable TMN interfaces.

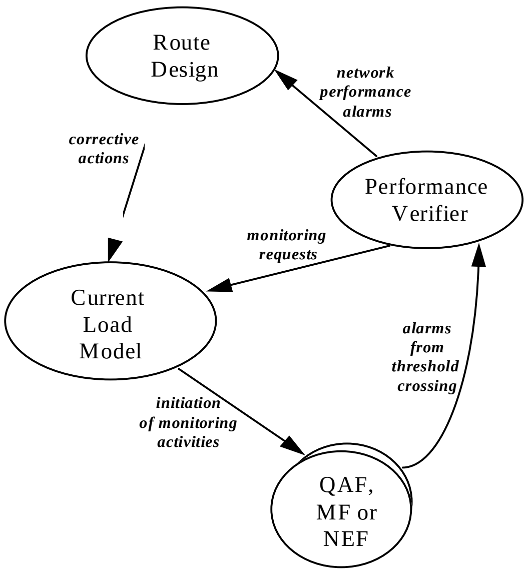


Fig. 6. Performance Verification Interactions

These monitoring activities are responsible for the creation of the appropriate Metrics and IM objects. The interaction between the PV service and the CLM OSF is based on the concept of asynchronous event reporting. By setting the thresholds at the Metrics and IMs and by using the asynchronous event and alarm reporting mechanism, rather than synchronous polling, both the communication and the processing overhead are reduced. Also the interaction between the CLM and the underlying NEFs, MFs or QAFs is achieved by the same mechanisms. Specifically, when a threshold crossing is detected at the lowest level, a QoS alarm is triggered. The PV OSF has already created an event forwarding discriminator in the CLM OSF and the CLM has created event forwarding discriminators in the NEFs, MFs or QAFs. So the alarm is forwarded to the PV through the CLM OSF. The reason for this indirect forwarding is that not all the alarms are forwarded to the PV OSF i.e. the CLM OSF implements a first level filtering and significance assessment function. Finally, after receiving such notifications, the PV Service is responsible for raising network-wide performance alarms. The latter are directed to the Route Design OSF which should implement corrective actions through the current load model i.e. route and bandwidth allocation changes. The latter are implemented by changing configuration information in network elements.

By distributing the functions required over the Network Management and Network Element Management Layers and by utilizing the powerful framework of OSI management, the architecture of the TMN hierarchical system can be designed so as to avoid the management communications overhead inherent in centralized systems. Decentralisation is achieved by moving management intelligence “downwards” to network elements and keeping costly polling functions there, alleviating the traffic on the managed network and allowing higher-level management functions to concentrate on problem solving.

6. Discussion

The distributed intelligent monitoring and reporting facilities presented, try in essence to move intelligence to the agent part of the manager-agent model, reducing network traffic and allowing managing entities to focus on the realisation of management policies, being supported by a rich event-based paradigm. Their generic nature makes it possible to implement them once and make them part of relevant platform infrastructure.

Their use can enhance "raw" information models and pays real dividends when used in conjunction with simple information models resulting from SNMP to GDMO translation [14]. Arbitrary expressions combining attribute values are possible, while the use of scoping and filtering provides greater flexibility in specifying information with dynamic properties to be observed.

6.1. Management by Delegation

All the above facilities, from the Metric objects to the fully dynamically distributable IMs, try in essence to break away from the static, compiled knowledge of monitoring intelligence. In fact, they are steps towards fully dynamic intelligence that could be downloaded to the managed resources and operate autonomously as much as possible, reporting back to a master managing object which will have a global view of the domain in which it operates the policy. Such intelligence may include monitoring aspects as well as intrusive (control) in order to react to pre-defined conditions that express the management policy in hand.

That paradigm of operation is often referred to as "*management by delegation*" [21]. Relevant research has taken place and is still ongoing at Columbia University [21][22], INRS-Telecommunications Canada [23][24] and University College London [15]. One key aspect of delegation is the definition of a scripting language that is essentially a low-level policy language for a particular management paradigm (SNMP, OSI Management, CORBA or other). Scripts could be downloaded and executed in a controllable fashion close to the managed resources. The general delegation model is presented in [21] while [22] points to possibilities of realisation in the SNMP framework. In the OSI Management context, delegation can be achieved through Active Objects [15] which are defined in GDMO while the relevant interpreted scripting language is an extension of TCL. The delegated script is downloaded through an attribute while its execution state, exceptions and results are controlled through other attributes and notifications. [23] and [24] present a delegation/worm paradigm with its own object-oriented model based on the CAML language and a lightweight process environment.

While the full delegation approach is promising, there are still a number of problems to be solved. The main limitations are the low efficiency of the interpreted approach and the verification of arbitrary scripts before execution: a misbehaved script may consume arbitrary computing resources, leading potentially the managed system to starvation. The intelligent monitoring approach presented here avoids these problems by following the compiled model, parametrising the behavioural of IMs through their attributes. The flexibility of this approach can go quite far through the scoping and filtering facilities, the specification of arbitrary mathematical operations as explained and the use of the rich OSI event reporting facilities. Ultimately though, a full delegation paradigm following the interpreted model will be able to support the same functionality in a more flexible fashion. What remains to be seen is if the other related problems will be overcome, most notably the low performance e.g. a uniformly weighted moving average algorithm realised in a language like TCL is orders of magnitude slower than the same logic in C/C++.

6.2. Security Issues

Another important aspect of any distributed management environment is security. In the OSI management model [2], security services comprise authentication, integrity, confidentiality and access control. Authentication is checked when an association is established while integrity and confidentiality relate to information in transit; these are obviously orthogonal to managed objects. Access control enables authorised initiators, i.e. applications in managing roles or human users driving such applications, to have different levels of access to the "exported" management objects by the target application. The enforced access control could be at the object class / instance level or at the individual attribute, action and notification level [9].

Access control is enforced by special support managed objects which identify authorised initiators, the protected targets (i.e. objects to be accessed) and the rules that express the access control policy. Protected managed objects (targets) are “unaware” of the access control policy they are subject to. This means they can be implemented without being concerned with access control at all. The access control policy is defined and realised through a set of relevant access control objects which are instantiated in an object cluster. The generic part of an agent application authorises or refuses particular requests to managed objects based on information in those access control objects.

As already discussed, Metrics and IMs act essentially as managing objects within an application in agent role. According to the standard access control model, unauthorised initiators may be not allowed to create such objects. If, though, creation rights are granted, every time such an object is created or modified by an initiator, it should be also checked that the initiator has *read* access to the observed or summarised information. This is because the initiator may indirectly “learn” about the values of other objects through a Metric or IM. The problem with enforcing this type of access control is that it cannot be done generically, as the Access Enforcement Function (AEF) does not know the semantics of the Metrics or IMs. In general, the AEF belongs to the generic part of an agent application and knows only the semantics of the access control related objects.

The solution to the problem is to implement the part of the AEF that needs to check read access rights to the monitored information in the Metrics and IMs themselves. This check is performed when these objects are created or whenever the monitored information changes e.g. through a set operation. The drawback of this approach is that the access control infrastructure becomes non-transparent i.e. the implementation of the Metrics and IMs is exposed to details related to access control and becomes, as such, more complex. In addition, any changes to the access control model will have an impact to the implementation of the Metrics and IMs. This is unfortunate but it is considered a small price to pay given the power and flexibility these objects provide. It should be also added that in the full delegation approach i.e. if an interpreted script is evaluated by such objects, access rights to other objects should be checked *every* time, the reason being that references to other objects may be produced dynamically by the executing script. This exacerbates the general performance problem of the interpreted approach.

6.3. Mappings on ODP / CORBA

All the work on IMF and IM objects has been based on the OSI management framework [2]. Given the advent of ODP-based approaches [4], an interesting consideration is to investigate the mapping of those concepts onto OMG CORBA [10], regarding the latter as the pragmatic counterpart of ODP. The IM objects are specified in GDMO/ASN.1 [6], from an information perspective and realised as C++ engineering objects, using suitable flexible infrastructure provided by the OSIMIS platform [13]. It should be possible to map them onto CORBA IDL objects from a computational perspective and realise them as engineering objects on a CORBA platform.

Though this is theoretically possible, first considerations reveal a number of limitations. First, there is no notion of clusters of objects in CORBA as in OSI agents or access to them with database-like facilities such as scoping and filtering. As such, management traffic will be generated, defeating partly the purpose. Another more important limitation is that the notion of unique global names across domains does not (yet) exist in CORBA. Also, it is not easy to map onto CORBA IDL recursive ASN.1 definitions like the one used by intelligent monitors to specify arbitrary operations on observed attributes.

These considerations relate to the current state of CORBA. Trading services may be used in the future to emulate scoping and filtering while the issue of global naming is going to be addressed through naming servers and federation. The problem of mapping complex recursive ASN.1 syntaxes to equivalent IDL ones remains but there exist workarounds through non-generic mappings. We will report our findings in the future.

7. Conclusions

We explained in this paper the issues behind distributed intelligent monitoring and reporting, and the use of the OSI management access and information models to provide flexible generic support object specifications, the Intelligent Monitors. We also showed the applicability of those objects in the TMN context through some simple but representative case studies. Given the future integration of service execution and management infrastructures in a unifying framework based on ODP principles, it is of paramount importance that support for such powerful concepts is maintained.

Acknowledgements

This paper describes work undertaken in the context of the RACE II ICM, TOMQAT and the ESPRIT-III IDSM projects, which are partially funded by the CEU.

References

[1] CCITT Rec. M.3010, *Principles for a Telecommunications Management Network (TMN),* Study Group IV, Report 28, 1991.

[2] CCITT Rec. X.701, *Information Technology - Open Systems Interconnection - Systems Management Overview*, 1991.

[3] ITU-T Rec. X.500, *Information Technology - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Service*, 1988.

[4] ITU-T Rec. X.900, *Information Technology - Open Distributed Processing - Basic Reference Model of ODP - Part 1: Overview and guide to use*, 1994.

[5] CCITT Rec. X.710, *Information Technology - Open Systems Interconnection - Common Management Information Service Definition, Version 2*, 1991.

[6] CCITT Rec. X.722, *Information Technology - Open Systems Interconnection - Structure of Management Information: Guidelines for the Definition of Managed Objects*, 1992.

[7] ITU-T Rec. X.739, *Information Technology - Open Systems Interconnection - Systems Management - Part 11: Metric Objects and Attributes*, 1993.

[8] ITU-T Rec. X.738, *Information Technology - Open Systems Interconnection - Systems Management - Part 13: Summarisation Function*, 1993.

[9] ITU-T Rec. X.741, *Information Technology - Open Systems Interconnection - Systems Management - Part 9: Objects and Attributes for Access Control*, 1995.

[10] Object Management Group, *The Common Object Request Broker Architecture and Specification (CORBA)*, 1991.

[11] J.Sanchez, G.Mykoniatis, G.Pavlou, J.Reilly, K.McCarthy, *Intelligent Monitoring Functions in OSIMIS*, ICM/WP3/NTUA/0097, 1993.

[12] G.Pavlou, T.Tin, A.Carr, *High-level APIs in the OSIMIS TMN Platform*: *Harnessing and Hiding,* in Towards a Pan-European Telecommunication Service Infrastructure - IS&N '94, Springer-Verlag, 1994.

[13] G.Pavlou, K.McCarthy, S.Bhatti, G.Knight, *The OSIMIS Platform: Making OSI Management Simple*, Integrated Network Management IV, Chapman & Hall, 1995.

[14] K.McCarthy, G.Pavlou, S.Bhatti, N.DeSouza, *Exploiting the Power of OSI Management in the Control of SNMP-capable resources,* Integrated Network Management IV, Chapman & Hall, 1995.

[15] A.Vassila, G.Knight, *Introducing Active Managed Objects for Effective and Autonomous Management in the TMN*, Bringing Telecommunications Services to People - IS&N'95, Springer-Verlag, 1995.

[16] ITU-T Rec. Q.822, *Stage1, Stage 2 and Stage3 Description for the Q3 Interface - Performance Management*, 1993.

[17] J.Kippe, *Monitoring and Reporting Policies*, International Workshop on Services for Managing

Distributed Systems ’95, Karlsruhe, 1995.

[18] J.Kippe, *Implementation Specification for Additional Policies - Monitoring Policy*, IDSM Deliverable 23, 1995.

[19] J.Case, M.Fedor, M.Schoffstall, J.Davin, *A Simple Network Management Protocol*, RFC 1157, 1990.

[20] P.Georgatsos, D.Griffin, *Management Services for Performance Verification in Broadband Multi-Service Networks*, Bringing Telecommunications Services to People - IS&N’95, Springer-Verlag, 1995.

[21] Y.Yemini, G.Goldszmidt, S.Yemini, *Network Management by Delegation*, Integrated Network Management II, North Holland, 1991.

[22] G.Goldszmidt, Y.Yemini, *Evaluating Management Decisions via Delegation*, Integrated Network Management IV, Chapman & Hall, 1995.

[23] J.-Ch.Gregoire, *Management Using Delegation*, Advanced Information Processing Techniques for LAN and MAN Management, North Holland, 1993.

[24] J.-Ch.Gregoire, *Models and Support Mechanisms for Distributed Management*, Integrated Network Management IV, Chapman & Hall, 1995.

Appendix: ASN.1 syntax to express arbitrary monitoring algorithms

```
InfoFunctionSyntax DEFINITIONS ::=
BEGIN
IMPORTS          AttributeId FROM CMIP
                 DistinguishedName FROM IF ;

    InfoFunction          ::= SEQUENCE {   CHOICE {operation Operation,
                                                   operands Operands },
                                           CHOICE {operands Operands,
                                                   infoFunctions SEQUENCE OF InfoFunction} }

    Operands              ::= SEQUENCE OF Operand

    GlobalAttributeId     ::= SEQUENCE{    globalMOName DistinguishedName,
                                           attributeId AttributeId}

    Operand               ::= CHOICE {     globalAttrId GlobalAttributeId,
                                           constant Const }

    Const                 ::= CHOICE {     integer INTEGER,
                                           real REAL }

    Operation             ::= ENUMERATED {         add (1), -- >= 2 operands
                                                   subtract (2), -- ..
                                                   multiply (3), -- ..
                                                   divide (4), -- ..
                                                   arithmeticalMean (5), -- ..
                                                    -- other operations may be added}

END
```