40

Deriving Variable Polling Frequency Policies for Pro-active Management in Networks and Distributed Systems

P. Dini, R. Boutaba
Computer Science Research Institute of Montreal
1801, McGill College street, # 800, Montreal, Qc, H3A 2N4, Canada
e-mails: {dini, rboutaba}@crim.ca

Abstract

Management activities are based on the state of distributed system components, relations of these components, and their behaviour. Since management policies are applied across an abstraction of distributed systems, the quality of decisions is dependent on the representation fidelity of the system real state. Obviously, the data collection process updating the abstract representation of real distributed systems has a major significance to carry out relevant management decisions. Existing approaches promote a fixed polling frequency for all components. We present shortcomings of these issues and propose adapted polling formulae considering the behavioural parameters of managed components. Models for the flexible polling frequency and for the frequency adaptor are presented. Based on these models, we propose adapted formulae considering both the availability features and dynamics features in a linear and exponential approach. Policies for applying these formulae are proposed with respect to the component grouping and frequency classification. Implementation aspects are discussed.

Keywords

Variable polling frequency, pro-active management, distributed systems

1 INTRODUCTION

Two paradigms are currently used for the system management namely, *the platform-centred management* (commonly used by proprietary architectures) and *the distributed management* (recommended by OSI and ODP standards). Distribution is viewed either as *heterogeneous management entities* localized on specific sites and cooperating for a common problem, or *homogeneous management entity units*, distributed on many sites (so called SMAEs in OSI

This work was partially funded by the Ministry of Industry, Commerce, Science and Technology, Quebec, under IGLOO project organized by the Computer Science Research Institute of Montreal, and by a grant from the Canadian Institute for Telecommunication Research (CITR) under the Networks of Centres of Excellence Program of the Canadian Government.

A. A. Lazar et al. (eds.), *Integrated Network Management V*

standards). Management activities are based on the state of DS (Distributed System) components, relations of these components, and their behaviour. Since *management policies* are applied across an abstraction of distributed systems, the quality of decisions is dependent on the representation fidelity of the system real state. Obviously, the data *collection process* updating the abstract representation of real DSs has a major significance to carry out relevant *management decisions.*

1.1 Architectural issues

Updated data are collected into two phases namely, at *physical level* (from real resources to standardized records) and at *management level* (from management agents to their correspondent managers). As shown in Figure 1, these data are collected from the real resources either by the source initiative as change events (*physical notifications, management notifications*), or by specialized software agents (*physical actions, management actions*). At the *management level,* received data are recorded within managed object MIBs (Management Information Bases) each time an update action or notification occurs.

In the *platform-centred management* approach, a manager receives from or interrogates its own agents with respect to the state of MIB components, via *management notifications (event approach),* and respectively *management actions (commands approach). Centralized management* policies are consequently applied on MIBs across appropriate agents. A similar aspect raises within a *distributed management* platform. However, in this approach, decision policies are derived by knowledge exchanges between managers.

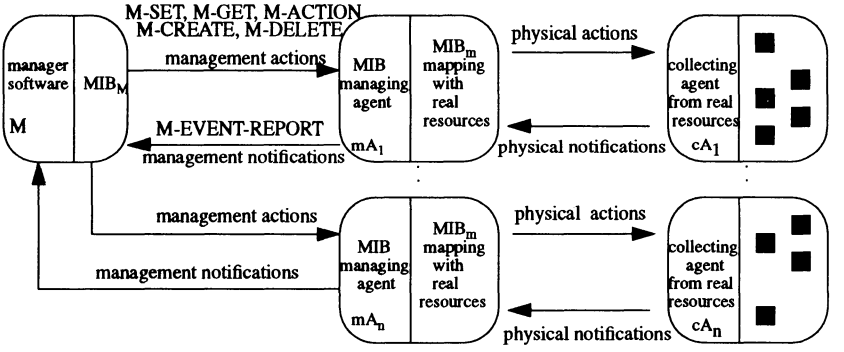

Figure 1 Overall Architecture of the OSI Management System.

As previously presented, managers must accurately know the real state of managed components. Within the *event approach,* a real DS component automatically sends notifications (at the physical level) as state change events, which are conveyed by specialized agents to MIBs. The MIB agent sends in turn this notification (at the management level) to its own manager. The more the number of changes is high, the more the amount of traffic increases, and the system performance implicitly decreases. Even further, many times, the notification data may not be relevant for the system management.

In the *commands approach,* supervisors (agents for the real resources, or managers for agents) send actions to collect data (eventually updated). Collecting commands could be issued periodically with a variable or fixed frequency. The process of collecting information at regular intervals is known as *polling.* The result of a polling operation is either new data (as *information message*), or no new changes (*control message* with no reports).

There are three *important features* of this process namely, the *polling interval* defined as the

amount of time between two consecutive polling operations within which the polled component has not transmitted new data, the *walk time* representing the amount of polling interval consumed by polling messages, and the *response time* referring to the amount of time between a command and the appropriate response. Evaluation of these features depends on several management aspects.

Among the two most polling relevant aspects are the *standardized management communication and recorded information*, and the *adaptation of the polling frequency* with respect to significant evolving system parameters. For the former, the problem is solved only at the *management level*, where the polled resource must accept the management messages as used by its callee, for example SNMP (Snmp 1990), or CMIP (Cmip 1991) protocols (Figure 1). Data on real resources are recorded in special management data bases called MIB (Mib 1990) as managed objects, and accessed by management agents. At the *physical level* there are only private solutions. Several attempts for a generic agent (Claudé 1990) must be developed. Adaptation of the polling frequency could conform with either general conditions (traffic, specific topology), or component behavioural aspects (state, notifications).

In DSs, two types of *polling mechanisms* are currently used: the *roll-call polling* and the *hub polling* (Stalling 1993). The former implies that each subordinate resource is in turn interrogated, while its manager is waiting to the *response message*. The latter supposes that interrogated subordinates are tightly coupled within a loop. The manager sends a *polling operation* to a loop head-component which is propagated up to the loop tail-component. Head- and tail-components receive and respectively send the start and the stop polling cycle flags from/to their callee. Further, all loop components, except the loop tail, pass the polling operation request to its neighbour. All called components send a message to the callee (*information* or *control*). Roll-call polling mechanism is used by SNMP across Internet (Stalling 1993). Hub polling is frequently combined with the roll-call polling and used within intelligent hardware resources at the CAN (Chip Area Network) or BAN (Board Area Network) level, since the hub polling is strongly dependent on the topology of resources (known as *daisy chaining* mechanism (Liu and Gibson 1986).

1.2 Shortcomings of existing approaches

Managing by the *event approach* is expensive in time costs. Even in the *command approach* the *polling frequency* must continuously be adapted in order to avoid *control messages* as responses. Several existing approaches propose a statistical computation of the polling interval according to the network traffic (Schwartz 1977), buffer occupancy (Konheim and Meister 1974), or particular parameters of network topologies (Ahuja 1982). This computation is rather static because mainly the average values are considered. Callison claims that values of the real-system parameters must be *re-evaluated* at a fixed temporal intervals (based on *newer statistics* on the previous parameters) in order to ensure the data validity (Callison 1994). Then, the polling frequency is globally established considering statistical combinations of many network parameters. Its update is valid for all polled components. In the following we presents several undesired results due to these approaches.

Let us take a simple example based on Figure 1, where a manager M polls n agents called mA_i (i = 1..n). Conforming to the existing approaches all agents are equally treated within a polling cycle, that is, M sends a message to mA_1 and then, it waits for the mA_1's response. mA_1 will be revisited after the polling of all mA_i (i = 2...n). If all these n-1 pollings return no new changes, the network has been unnecessary *overloaded*. Further, during this time, mA_1 could have many changes to transmit, that will be *delayed* in the next polling cycle. In related works, the polling frequency is the same for all components, determining either a high management traffic which diminishes system performances in the case of no new changes, or a loss of possible relevant changes, for those components which are less stable. We consider that the polled components must be classified upon several behavioural or managerial criteria. For each group, a distinct polling frequency could be used.

In *de facto* systems, the roll-call polling mechanism is used. A distinct polling frequency is useful even if the components are polled in parallel. Here, the problem is how to classify polled components in groups having the same polling frequency. Polling frequency can be updated with respect to either many *global parameters* (as in previous works), or *specific parameters*, such as *component performances*. Component performances refer to *qualitative parameters* of component's services. Our thesis is that the polling frequency must be adapted with respect to the *component's behaviour*. Several current behavioural parameters concerning the *availability features* (Dini, Bochmann, Boutaba 1996) and *dynamics features* (Dini 1996) have been identified. The proposed approach in this paper takes a *basic polling frequency*, computed by one of previous statistical methods, and *adapts it* to be *more accurate* with respect to the component behaviour.

This paper focus on adapted formulae of polling frequency in DSs, based on behavioural component parameters which are currently measured or computed. In Section 2 we present several related works treating *statistical polling formula* and *behavioural parameters* with respect to the *component availability* and *dynamics features*. Section 3 focuses on our proposals, reporting on polling model, frequency adaptor model, and adapting formulae. Two kinds of corrections are proposed namely, linear and exponential. In Section 4 we present results of the implementation of these proposals. Conclusion and future work presented in Section 5 conclude the paper.

2 RELATED WORK

An analysis of the *polling interval* concerning the *roll-call mechanism* is presented in (Agbaw 1994). Agbaw discusses the *polling interval* of the *roll-call mechanism* used by SNMP and presents factors which determine the elapsed time between successive polling operations of the same agent. The calculus implies a number of N agents subordinated to a manager, the average time (Δ) required to perform a single poll, and the desired polling interval for a manager engaged full-time in polling.

Other statistical analyses of the performance in order to establish a polling interval are presented in the literature. Schwartz identifies the *polling interval* and the *response time* as two significant network performance facets (Schwartz 1987). However, the response time is considered more directly related to user needs. All equations presented in by Schwartz are related to the relationship between the *walk time* (amount of polling interval consumed by polling messages), *polling interval*, and *response time*. In this approach, the polling interval is a random variable, depending on the *network traffic* in both polling directions. Schwartz's model is *traffic-based*. The polling interval is not updated with respect to the behaviour of the polled component.

Konheim's and Meister's (1974) statistics model is *buffer occupancy-based*. Computing formulas focus on the communication invariants at each component (packet-arrival rate, frame-length, average of the walk time). The calculus is oriented to get the polling process time, rather than to capture the influence of the system component behaviour. Consequently, this is a complementary approach to our proposal.

Ahuja's model is based on a particular case, where polled components respond only by *control messages* (no information). In this particular approach, the time distance between adjacent components is equal and invariant. A computing formula for the polling interval is obtained (Ahuja 1982). The model could approximate the minimum polling interval for a single level of a hierarchy manager-agents.

The foreseen works prescribe formulas to compute the polling interval by only taking into account the propagation time issues referring to either the component topology, traffic parameters, or communication aspects related to the polling process itself. Callison claims that values of the real-system parameters must be *re-evaluated* at a fixed temporal intervals (based on *newer statistics* on the previous parameters) in order to ensure the data validity (Callison 1994). Otherwise, the correctness of the collected information does not necessarily imply cor-

rect management decisions, since real-time systems involve active components (Shin and Parameswaran 1994). However, we consider that a flexible correlation of the polling frequency with *behavioural parameters* of a DS components can substantially reduce the *management traffic*. These behavioural parameters are presented in details by Dini, Bochmann and Boutaba (1996) and Dini (1996), and briefly described in Dini, Bochmann, Koch, and Krämer (1997).

3 PROPOSAL

Our proposal refers to three following aspects namely, the *model* for a variable polling frequency, *adapted polling frequency formula*, and *criteria for component classification* in groups having the same polling frequency. We are considering here only the polling at the *management level*.

3.1 Models for flexible polling frequency

In Figure 2, we present the current model versus our proposed model. In the *existing model*, the same polling frequency is initially established at a fixed value for all polled components. This value is adapted with respect to the network traffic, the network topology, and other global network parameters. We propose a new polling model, where polled components are classified in groups. A distinct polling frequency, e.g. f_{01} or f_{02} in Figure 2, is used for each group, which represents the basic frequency computed for the existing system component, but corrected with a *component type depending factor* (ζ).

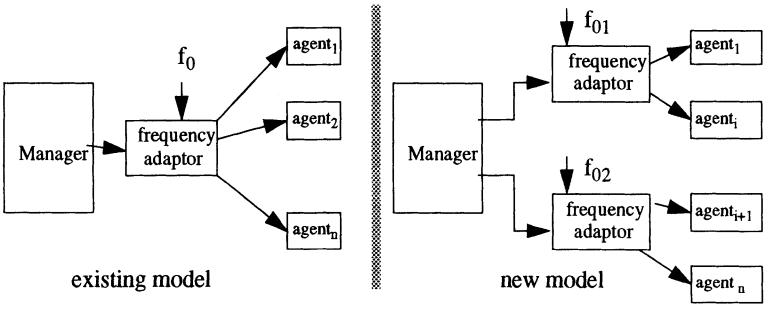

Figure 2 Polling models using a fixed or a variable polling frequency.

In our solution, we consider that the basic frequency f_0 is deduced from formulae of existing approaches which apply various computing criteria, as presented further. If one considers only the number of polled components (n) and the average time required to perform a single poll (Δ), the polling frequency should guarantee that $f \leq 1/n\Delta$ (Stalling 1993). Consequently, in this case, $f'_0 = 1/n\Delta$.

In the traffic-based approach, the average scan time ($\bar{t}_c$) is calculated assuming a fixed walk time for all stations, considering the arrival rate of packets at each station, and including the total traffic intensity on the common channel (Schwartz 1987). For those DS components polled upon Schwarz's criteria, the basic polling frequency is $f''_0 = 1/\bar{t}_c$. In the topology-based approach, the poll scan time T is computed following Ahuja's criteria, which assumes that the distances between adjacent stations are equal. The basic polling frequency for these systems is $f'''_0 = 1/T$. Since all these frequencies ignores behavioral aspects related to DS components, the basic frequency represents the minimum polling frequency established with respect to global conditions. This basic frequency will be modified by using the correction coefficient, whose

computing formula will be proposed later. Consequently, the new polling frequency must dynamically be adapted.

3.2 Frequency adaptor model

Since different subsystems could correspond to different previous criteria, the proposed frequency adaptor model allows to compute the variable polling frequency using f'_0, f''_0, f'''_0, or any other basic frequency. For each component or group of components, the adaptor must compute the appropriate coefficient (ζ) based on behavioural parameters. The proposed model has four distinct blocks, as shown in Figure 3.

- *the component health parameters block* represents the current availability and dynamics features, as discussed in Section 2.2 (usually, a specific MIB-like database);
- *the coefficient computing block* receives these current values (6) and computes the appropriate ζ coefficient (5). We will propose ζ computing formula in Section 3.3.
- *the frequency adaptor block* receives several input data having distinct meanings. From the discriminator block, there is the *activate* (4) command which starts the frequency adaptor activity. The entry (1) ensures different basic frequencies (it can always be modified), whereas the entry (2) is reserved for an imposed basic frequency due to management goals. The variable frequency f_{0i} is the new polling frequency for the component *i*.
- *the discriminator block* has three functions. First, it automatically performs the correction based on entries called (1), (2), and (4), giving a variable polling frequency noticed f_{0i}. Second, based on the entry (5), it captures certain abnormal values, and commands the highest polling frequency.

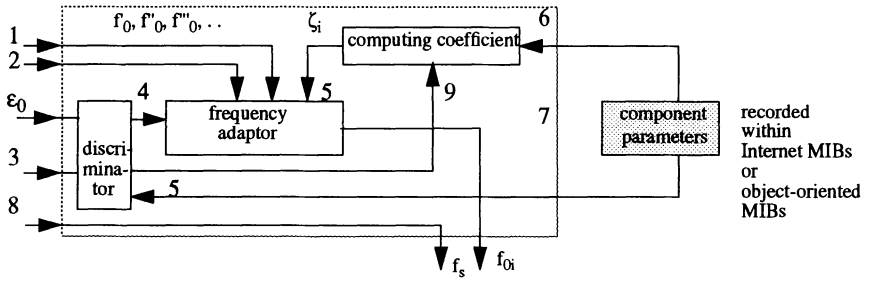

Figure 3 Model of the adaptor for a variable frequency.

Concrete conditions will be discussed later. Third, according to the manager policy, the discriminator imposes the computing formula to the coefficient computing block (9). This issue refers to either components classified in the same group, or to a single component which has changed its own behaviour. The entry (8) ignores all previous variants and imposes a polling frequency as requested by the manager (f_s). The activity performed within this model has three options: either the manager requests the polling with a well-defined frequency (8), the manager imposes a basic frequency f_0 (1 and 2) which will be adapted by the coefficient ζ, or the discriminator detects an anomaly of h(t) value ranges. Since the first case is manager-dependent, we are concerned in the following with the computing polling frequency of the last two cases.

3.3 Adapting formulae

Various adapted formulae have been proposed to be used by the coefficient computing block.

Conforming to the existing approaches, we assume that a polling frequency f_0 is previously computed with respect to network global parameters. Two approaches are possible to adapt the polling frequency:

- *using behavioral parameters:* Behavioural parameters summarized in Dini, Bochmann, Koch and Krämer (1997) may be used independently, or in combination to compute new polling frequency values. These parameters are related to operational aspects of managed objects. From a generic formula $[f_\zeta, (1 + \zeta)f_0$, where ζ is a correction coefficient, four different adapted polling frequency formulae have been deduced. An agent-based management using these formulae is presented in the paper cited above at this conference.
- *using new state change models:* In order to predict the behavior of a managed object and apply pro-active management policies, a refinement of state change model have been proposed in Dini, Bochmann, and Boutaba (1996). We consider aspects related to the usage state, current availability, and costs. This paper presents several policies related to this approach.

In both cases, the proposed formulae try to adapt the polling frequency to current changes of behavioural parameters or states. If the discriminator receives (input 5) behavioural parameter values below several thresholds guaranteed by the component, it disallows the calculus of the coefficient computing block and allows the manager to apply other appropriate policies, i.e. a well specified polling frequency f_s. If the current availability tendency is stable, the discriminator stops the coefficient computing and f_0 is applied.

3.4 Polling algorithms

The following scenarios could be used by a manager to apply a self-adapting managing policy based on the state of managed objects.

Simple examples

Let us consider a simple system of a server with many clients. Various situations can be identified, as presented in the following examples.

- *example 1.* The manager uses a polling frequency policy to *get state* values of managed objects with a fixed polling frequency f_0. The managed object representing the server can send traps or notifications concerning its usage state as follows: if the number of clients exceeds 10, the server send *active1* event, if the number exceeds 15, the server send *active2* event, whereas the *active3* event is sent when the number of clients is 19. 20 represents the internal event of the server managed object causing the state change notification of type *usageState* from *active* to *busy.*
- *example 2.* The manager uses a polling frequency policy to get state values of managed objects with a fixed polling frequency f_0. The health of the managed object representing the server varies according to variations of its operational state (Dini, Bochmann, Boutaba 1996). The managed object representing the server can send *alarm1*, *alarm2*, and *alarm3* traps or notifications appearing at well-defined current availability thresholds. For example, alarm1 corresponds to *currentAvailability = 0.9*, alarm2 to *currentAvailability = 0.85*, and alarm3 to *currentAvailability = 0.75.*
- *example 3.* The manager uses a polling frequency policy to *get state* values of managed objects with a fixed polling frequency f_0. If within a large period of time, there are no notifications from the managed object representing the server, or the *get state* does not capture a state change, the traffic has been usefulness charged. A function adapting the polling frequency with respect to costs incurred by polling without changes versus changes occurring and not captured in time has been proposed by Agbaw (1994). This solution is valid only in polling-driven systems, such as Internet.

Flexible polling frequency algorithms

Since CMIP-based systems allow the polling approach by the primitive M-GET, while the SNMP-based systems acts as event-driven across the primitive Trap, different implementation solutions are specific to event-driven (CMIP-based) or polling-driven systems (SNMP-based). Consequently, regardless of which approach is considered, an harmonization of these two ways must be performed by an adequate *flexible polling frequency policy.* Since the current availability value represents the confidence that the managing system has on the real resource availability, it seems naturally to accommodate the polling frequency with the current availability. A higher polling frequency is used to poll managed objects displaying a low current availability, following an adaptation function. Even in the event-driven approach, the polling frequency policy could detect the de-activation of notification services. However, if a managed object has a stable usage state or operational state, the management traffic is usefulness increased for a class of managed objects, while a uniform polling frequency could lead to loss useful information regarding other managed objects. According to these cases, we develop three flexible polling frequency policies concerning either i) the usage state, ii) the current availability, or iii) the costs incurred by usefulness polling versus changes lost. Hereafter, E represents the component type and F represents the component identifier.

i) flexiblePollingFrequency1 policy. This policy adapts the basic polling frequency f_0 to $f_{adapted} = (1 + th_i) \times f_0$, where th_i corresponds to the notification $active_i$. These th_i are specific to different component types. For example, a CPU decreases its QoS according to the number of clients. When this number increases, the probability to cause failures because of lack of capacity becomes greater.

```
.....
flexiblePollingFrequency1
.....
functional behavior
 input:   E:type, F: name, active_i, f_0, th_1, th_2, th_3
 output: f_adapted
.....
 begin
   compute f_adapted = (1 + th_i) x f_0
 end begin
.....
```

ii) flexiblePollingFrequency2 policy. This policy adapts the basic polling frequency f_0 to $f_{adapted} = f_0 / h_i$, where h_i thresholds correspond to the notifications $alarm_i$. These h_i are specific to different component types. Since the current availability value represents the confidence the managing system has on the real resource availability, it seems naturally to accommodate the polling frequency with the current availability. A higher polling frequency is used to poll managed objects displaying a low current availability.

```
.....
flexiblePollingFrequency2
.....
functional behavior
 input_1: E:type, F: name, alarm_i, f_0, h_1, h_2, h_3
 input_2: E:type, F: name, f_0, h_1, h_2, h_3
 output: f_adapted
.....
 begin1
   compute f_adapted = (1 + h_i) x f_0
 end begin1
.....
 begin2
   invoke currentAvailabilityFunction (Dini 1996)
   if currentAvailability > h_1 then f_adapted = f_0
   if currentAvailability > h_2 then f_adapted = f_0/h_1
```

```
    if currentAvailability > h3 then f_adapted = f0/h2
    else f_adapted = f0/h3
  end begin2
.....
```

ii) flexiblePollingFrequency3 policy. This policy is based on costs of the usefulness polling C_p versus costs incurred by changes lost C_l. According to these costs, Agbaw proposed a function to adapt the polling frequency (Agbaw 1994). Let us suppose that we have these costs for each type of system component.

```
.....
flexiblePollingFrequency3
.....
functional behavior
 input: E:type, F: name, Cp, Cl, f0
 output: f_adapted
.....
 begin
   invoke f_Agba (Agbaw 1994)
   f_adapted = f_Agba

 end begin
.....
```

Combined polling policy

Polling policy combines these three *flexible polling policies* with the get state management action to self-adapt the polling to current state changes, threshold notifications, or costs.

```
.....
polling policy
.....
f_polling = f0
functional behavior
 input: E:type, F: name, th1, th2, th3, h1, h2, h3
        [alarm1 | alarm2 | alarm3] | [active1 | active2 |
         active3], Cp, Cl

 output: E:type, F: name, f_polling
.....
begin
   if alarm_i then flexiblePollingFrequency2(alarm_i)
                   and
                   f_polling = f_adapted
   fi
   if active_i then flexiblePollingFrequency1(active_i)
                   and
                   f_polling = f_adapted
   fi
   else get state (E, F)
       if currentAvailability < h1
                   then flexiblePollingFrequency2(h)
                   and
                   f_polling = f_adapted
       else flexiblePollingFrequency3
                   and
                   f_polling = f_adapted
       fi
   fi
end begin
.....
```

According to the behavior of the *combined polling policy*, the frequency of polling is adapted to various constraints, coming from the unexpected behavior of system components.

4 IMPLEMENTATIONS ISSUES

Several aspects arise applying these formulae. Commonly, even for a single component, the correction coefficient continuously varies. It is a heavy task to adapt the value of the polling frequency each time a change occurs. Then, a clustering of these values is useful. Second, having a distinct polling frequency for each component makes heavy the management activity. Frequently, the managers want to cover a area of components under the same umbrella. Then, a clustering of components must be adopted. Finally, the increasing complexity of computing formulae needs a manager policy to correctly select and link together the formulae's expressions.

4.1 Clustering polling frequency values

For one component or for a group of components, there is a single basic formula given by f_0. Usually, when a polling request is issued, returned data are received by the same manager component. If the manager has *n* distinct possibilities to process polled data, we propose a clustering of the largest interval $[f_0, (1 + \zeta)f_0]$ in intervals having the length equal to the $(\zeta \times f_0) / n$. For the exponential adaptation (Dini, Bochmann, Koch and Krämer 1997), the interval is $[f_0, (1 + e^{\zeta})f_0]$ and the ratio is $e^{\zeta} \times f_0$.

4.2 Component classification policies

Components behave a large spectrum of current parameter values and states. However, the polling frequency can not follow each variation. Consequently, the domain range of parameter values must be divided into many subranges. For each subrange, a unique polling frequency value is established. It will be useful to group components having the same subranges into classes in order to compute a single frequency. Sometimes, other grouping criteria are used (geographical, administrative, etc.). In this case, the manager must decide a policy to choose the reference values to represent a group and select the group's ζ.

Current availability-based classification policy

The policy we propose to classify parameter values considers several distinct cases:

- C_1 represents special components having a fixed polling frequency, regardless their current parameter values. This class has two opposite subclasses namely,
  - C_{11} denotes those components which are not critical, and then, their polling is done by using the basic frequency;
    C_{11i} (i = 1..n) component uses the basic frequency f_{0i};
  - C_{12} refers to critical components requiring a fixed high polling frequency;
- C_2 contains the remainder which is commonly classified as following,
  - C_{2i} (i = 1..5) represents components having the observed availability value within the interval $[(4+i)\times 0.1, (5+i)\times 0.1]$
  - C_{26} represents those components with a low current availability (h < 0.5 or other constraints) for what the manager does not poll.

Groups polling policy

Components belonging to the same class (C_{ij}) could be grouped in order to diminish the computing task. However, if for different reasons, such as administrative or functional dependencies, components of distinct classes must form a polled group having the same polling frequency, the question is how to determine the reference component.

We propose a selection policy with the most restrictive condition. Among group's compo-

nents, the manager must poll only the representative component of the group. The election is first performed by following the previous classification. Inside a class, the polling frequency is determined by that component having the maximum polling frequency.

4.3 Using a flexible polling frequency policy

Currently, we are experimenting these ideas within the context of automatic reconfiguration management. Several implementations issues belonging to the used platform are presented below. First, these computing formulae considering the behavior parameters inherit timing aspects presented in (Dini Bochmann Boutaba 1996) (Dini 1996), i.e., the availability and dynamics parameters are clock-time-dependent. However, the basic polling frequency currently used in SNMP-base management systems is fixed (10 seconds) between managers and proxy-agents, whereas the polling frequency between proxy-agents and the SNMP-agents could be modified, as presented in (Agbaw 1994). Referring to Figure 2, the proxy agents appear between an OSIMIS manager M and an SNMP-agent mA_i, in order to correctly map specific features of managed objects versus management protocols. Agbaw's formulae include also probabilistical evaluations of costs with respect to the information loss, or control message (unnecessary polling).

Other management aspects can benefit of the present proposal. For example, existing management tools can implement these algorithms to facilitate the automatic management of networks. Additionally, in self-testing systems, where some testing components are charged to automatically test critical functional components, a more accurate information can be obtained if the polling frequency is adapted according to the dynamic results of tests (Dini 1995). For example, when some tests are applied to a system component, the polling frequency may be increased, in order to accurately capture the real state of the concerned component into a small time vicinity.

Current network management stations products, such as *NetView/6000* (IBM), *SunNet Manager* (Microsystems), and *Open View* (Hewlett-Packard) help human operators to manage networks. They consider a graphic facilities to scroll and read MIB information, and a window to choose the polling interval, e.g. MIB Data Collection Dialog Box of NetView/6000. The polling interval value is established by the human operator as results of lecturing MIBs. The operator can also stop and restart the activity of data collection. Decisions of polling frequency and of stopping or restarting the collection are *human-based.* In open large systems this task becomes difficult and with not relevant results. Therefore, existing tools can implement policies to adapt the polling frequency.

5 CONCLUSION AND FUTURE WORK

In order to enhance the management policies in distributed systems we have presented in this paper a variable polling frequency based on the dynamic features of a managed object. Since existing approaches promote a fixed polling formulae for all components, there are no implemented mechanisms to automatically vary the period of polling. The management tools envisaging to easily allow to a human operator to change the polling frequency have not this feature. Our proposal are based on the current evaluation of component properties, that are explicitly described by its availability and dynamics features. The model proposed for the adaptor of a variable frequency permits to compute the correction coefficients and to discriminate between different management polling policies with respect to discriminator criteria. The proposed formulae can adopt a linear or exponential approach to correct the basic polling frequency established by previous studies. In order to better select the polling frequency we propose component classification rules based on the component properties. Consequently, several distinct classes are obtained. Finally, a criterion for component group is proposed. We have equally presented implementation issues referring to our experiment platform using the OSI-

MIS/ISODE distributed environment and SNMP-agents.

The ongoing studies are presently done on formulae including also probabilistical evaluations of costs with respect to the information loss, or control message (unnecessary polling). Additionally, a combination of health criteria and management constraint criteria imposed by costs could be used to build automatic management policies.

6 REFERENCES

Agbaw, C. (1994) Management Data Collection and Gateways, *M.Sc. Thesis*, McGill University, 1994.

Ahuja, V. (1982) Design and Analysis of Computer Communication Networks, McGraw-Hill, Computer Communication Networks, 1982.

Callison, H. R (1994) A Periodic Object Model for Real-Time Systems, *Research Paper, Department of Computer Science and Engineering*, FR-95, University of Washington, Seattle, WA, IEEE 1994.

Cana (1991) ISO/IEC 9596-1:1991, Information Technology - Open System Interconnection - Common Management Information Protocol - Part 1: Specification, CAN/CSA-Z243.142-91.

Claudé, M. (1990) *Unification de Contextes Hétérogènes par un Agent Générique OO*, Thèse de Doctorat de l'Université Pierre et Marie Curie, février 1990.

Dini, P., Bochmann, v. G., Koch, T., Krämer, B. (1997) Agent Based Management of Distributed Systems with Variable Polling Frequency Policies, *Proceedings of The IM'97 Symposium*, San Diego, 1997.

Dini, P., Bochmann, v. G., and Boutaba, R. (1996) Performance Evaluation for Distributed System Components, *The Second IEEE Systems Management Workshop*, Toronto, Ontario, Canada, June 19-21, 1996.

Dini, P. (1996) New Aspects for Run-time QoS Evaluation in Networks and Distributed Systems, Tutorial, *European Simulation Multiconference 1996*, Budapest, Hungary, June 2-6, 1996, pp. 3-11.

Dini, P. (1995) Management Policies of Active and Passive Tests in Distributed Systems, *Technical Report, IGLOO Project*, CRIM/University of Montreal, May 1995.

Konheim, A.G. and Meister, B. (1974) Waiting Lines and Times in a System with Polling, *Journal of the ACM*, 21, no. 3, July 1974, pp. 470-490.

Liu, L. and Gibson, G. (1986) *Microcomputer Systems: The 8086/8088 Family*, Prentice-Hall, 1986

Mib (1990) RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets: MIB II*, eds: K. McCloghrie, M. Rose.

Schwartz, M. (1977) *Telecommunication Networks: Protocols Modeling and Analysis*, Addison-Wesley Publishing Company, 1987.

Schwartz (1987), *Computer-Communication Networks Design and Analysis*, Prentice Hall, Inc., 1987.

Shin G.K. and Parameswaran, R. (1994) Real-time Computing: A New Discipline of Computer Science and Engineering, *Proceedings of the IEEE*, vol. 82, no. 1, January 1994.

Snmp (1990) RFC 1157, *A Simple Network Management Protocol*, M. Schoffstall, M. Fedor, J. Davin, J. Case (05/10/90).

Stalling, W. (1993) *SNMP, SNMPv2, and CMIP: The practical Guide to Network Management Standards*, Addison-Wesley Publishing Company, 1993.