# Minimizing the Monitoring Cost in Network Management

*J. Jiao, S. Naqvi*
*Bell Labs, 600 Mountain Ave,*
*Murray Hill, NJ 07974*
*USA*
{*jia, shamim*}*@research.bell-labs.com*

*D. Raz, B. Sugla*
*Bell Labs, 101 Crawfords Corner Road,*
*Holmdel, NJ 07733*
*USA*
{*raz, sugla*}*@research.bell-labs.com*

**Abstract**

Many rapidly-changing environments need to be monitored to ensure that they stay within acceptable parameters. The monitoring consists of *measuring* properties of the environment, and of *inferring* an aggregate predicate from these measurements.

In many cases it is too complex, or too expensive to conduct explicit monitoring at all times. In these cases, information *(integrity constraints)* on the evolution of this environment can often allow us to use past measurements to infer the future behavior, thus reducing the monitoring cost.

We provide a formal description of the problem of monitoring rapidly-changing data, which we call the monitoring problem. We then classify this problem in terms of the integrity constraints that govern the evolution of the environment, and propose different algorithms for each of these classes. For the most restricted case, we can find a greedy algorithm which is optimal, while for the more general cases we use competitive analysis and show that optimal worst and average case cost measuring algorithms exist. We then present heuristics for low cost low complexity measuring algorithms. We believe that the results of this paper can serve as a framework for further studies.

**Keywords:** Monitoring, polling, competetive analysis

## 1 Introduction and Motivation

Monitoring forms the basis of control and management systems. Continuous monitoring is essential to determine the current state of the managed system. A typical set of the activities that are required to determine the current state of the monitored system are as follows. First, a set of state variables are defined (e.g. SNMP MIBs [10]). Second, intervals at which these state variables need to be sampled are determined based upon the granularity of the control actions that are required. Finally, all the samples that are collected are processed continuously for interpretation and action. Therefore, the volume of data that is collected directly impacts the performance of the network used for collection, and the demands on the collector that does the processing. Hence any technique that helps reduce the volume of data that needs to be collected is useful and important.

The question of how to monitor integrated networks was addressed by the work of Mazumdar and Lazar [6, 7]. They mainly considered the problem of how to decide which variables should be monitored, and how to specify the appropriate ranges, so that the information required to achieve a certain management goal is available. More recent works deal with the problem of achieving high level management goals while maintaining the amount of system resources used for management purposes as small as possible [8, 2]. However, while these methods may reduce the amount of resources used for polling in certain scenarios, they lack a theoretical framework that will allow comparing the actual (or worst case) cost of using them.

In this paper we propose and analyze novel schemes to reduce the volume of data necessary to determine the state of a system. The fundamental notion that is exploited here is that typically many state variables have constraints on their evolution. Given the present value of a state variable(s), these constraints limit the range of values the state variable(s) can take at a future time. For example, consider the case that a 1-directional highway is partitioned into cells, and some mobile phones are moving from the left to the right (see Figure 1). Let $x$ be the number of phones that are present in cell A and $y$ be the number

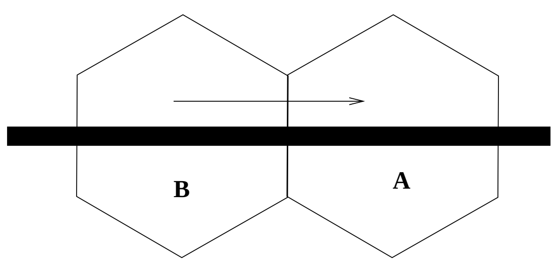


Figure 1: The mobile cell example

of phones that are present in cell B. If we know that the speed limit is 60mph, the length of a cell is 10 miles, and our time interval is one minute, then we can definitely say that the number of phones that will be present in cell A in the next time step is no more than $x + y$. This information can be used to save monitoring cost.

In network management, the manager may need to conduct polling of some network elements, and often management information base (MIB) files are used to store a significant number of current parameters to be probed by the manager during the polling process [10]. By utilizing certain constraints regarding the parameters, we may be able to reduce the polling cost. Such cost reduction could be very beneficial, both in reducing the load on the network, and reducing the CPU usage at the routers.

It should be mentioned that when we talk about monitoring and measurement, we are assuming the polling model, i.e. the value of any variable can be obtained only through polling instructions issued by the monitoring station. It is well-known, however, that in practical applications such as network management, asynchronous traps can be defined on devices, so that a notification can be generated without explicit polling. This can be more efficient since the trap is generated only when the alarm condition occurs. If the variable to be monitored is local to a device, then it would be in general more beneficial to set a trap on the device for the variable threshold, instead of explicit polling as discussed in this paper. However, if the variable can not be obtained locally but is a function of some parameters from multiple devices, it would not be possible to set traps on the global variable, and a polling method must be used. Note that setting a trap may require more CPU resources from the network element, as the value of the variable at hand should be constantly monitored (locally), and in many cases CPU is the bottleneck resource.

In addition, the basic assumption we make is that the integrity constraints are given to us so we can utilize them to reduce the monitoring cost. But in reality, these predictive rules are not readily available, and it is highly non-trivial to discover them, if they exist. In our simple example of mobile cells, since we have enough domain knowledge, we can derive the constraints through analytical methods. For more complex domains however, this may not be feasible, we may have to use statistical methods to obtain certain bounds based upon historical behavior of the variables. This aspect of the work is still on-going, and it is out of the scope of this paper. For this paper, we can assume that the constraints are provided by the user without considering how they can be obtained.

Competitive analysis of on-line algorithms was used to address a somewhat similar problem of moving data by S. Kahan [4]. He gave provably optimal on-line algorithms for a restricted family of functions, and for linear constrains. For the monitoring problem, as far as we know this is the first formal study of the problem, and many of our results are preliminary in nature. We are still carrying out on-going work on many of the issues, including: the evaluation of the cost for different on-line monitoring algorithms and the relationship to the cost of off-line algorithms, and the different ways the probabilistic nature of the constrains can be used. We believe that the results of this paper can serve as a framework for further studies of this important problem.

The rest of the paper is organized as follows: Section 2 gives an overview of our approach while Section 3 provides a formal framework. Section 4 expans this framework in the spirit of competetive analysis. Section 5 describes practical algorithm for the general case. Finally, in section 6 we briefly describe our conclusions.

## 2 Overview of the Approach

In the monitoring problem, there are a number of variables, each having an associated measurement cost. The optimization problem being considered in this paper is how to detect certain conditions, called *alarm conditions*, with minimum measurement cost. The configuration of variables in the system can be represented by an evolving state, and we must report alarm conditions at the earliest time. If the evolution is completely random then we must measure all the variables at each time step. However, there are usually predictive rules that restrict the evolution of the system.

**Example 1** Suppose we are monitoring the number of mobile users $x$ in a single cell, and the alarm condition is $x \geq 100$. And suppose the following rules hold due to constraints on the mobility of the users, which set the upper bound of the net increase in the value of $x$ from $t$ to $t+1$.

- If $x < 90$ at time instant $t$ then $x < 100$ at time $t+1$.
- If $90 \leq x < 100$ at time $t$ then it will either stay between 90 and 100 at $t+1$ or go above 100 at $t+1$.
- Once $x$ is above 100 it will stay there.

We can use Figure 2 to model the evolution of the mobile users over time. In the figure, an edge represents a possible transition from time $t$ to $t+1$.

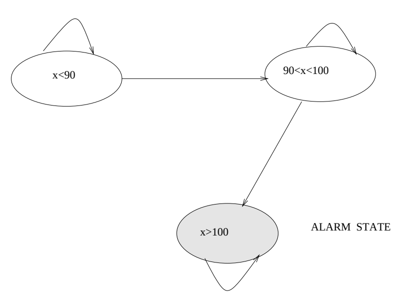


Figure 2: State Transition Graph for the Mobile Cell Example

Now assuming that $x < 90$ at time $t$, we see that we can wait until time $t+2$ to measure again. It is not necessary to measure at $t+1$ because the alarm condition can not be true. On the other hand if we wait until $t+3$ we may fail to detect the alarm at the earliest time.

This example has only a single variable. When multiple variables are present, we need to examine if there exists interdependence among them. For instance, if there is a second variable $r$, representing the ratio of the users in the cell that actually are using their mobile phone at a given time, and it is known that $r$ will not affect $x$, then it is not necessary to measure $r$. If variables are independent, non-required variables need not to be measured.

However, a more complex scenario occurs if the variables are interdependent. If some non-required variables could affect the future measurement time, then it might be beneficial to measure them as well. For example, suppose $y$ is the number of mobile users in an adjacent cell. Since it is known that users are moving between the cells, although $y$ will not trigger an alarm condition by itself, if we measure $y$ together with $x$ we may be able to wait longer to measure $x$ again in case $y$ is very small. On the other hand, if $y$ is big then the measurement cost will only be higher. So for such general cases, the best we can do is trying to reduce the average case cost. We will address such general cases in Section 5.

## 3 General Framework

We are given $n$ real-valued variables $x_1, \ldots, x_n$. Time $t$ is an integer, beginning at $t = 1$. Let $x_{it}$ denote the value of $x_i$ at time $t$.

**Definition 3.1** A *history line* from time 1 to time $t$ is a sequence $< s_1, .., s_t >$, where $s_j$ is an assignment of values to all the variables, i.e., $\{x_{1j} = a_1, x_{2j} = a_2, ..., x_{nj} = a_n\}$. Intuitively, a history line represents one possible sequence of values for the $x_i$'s at times $1, 2, 3, \ldots t$. A history line is said to be *legal* if it satisfies all the integrity constraints.

We can now formulate the monitoring problem as follows:

Each variable $x_i$ has a fixed, positive cost $c_i$, representing the cost of measuring it at any time, and its value lies in a partition of the real line into non-intersecting ranges $r_{i1}, \ldots, r_{ik_i}$. The intuition is that we are not interested in the value of each variable, but only in the range of which it is a member. (The reason behind this is that if we allow variables to take on continuous real values the problem can be undecidable). A *predicate* is an expression of the form $x_{it} \in r_{ij}$, where $r_{ij}$ is a range. We are also given a set $I$ of *integrity constraints* on the evolution of the $x_i$'s, each of them is of the form:

$$p_{1,t_1} \wedge p_{2,t_2} \wedge ... \wedge p_{m,t_m} \rightarrow pr_1 \vee pr_2 .. \vee pr_l$$

where $pr_1, pr_2, ..pr_l$ are predicates regarding a single variable at time $t + k$ for some $k$, and each $p_{i,t_i}$ is a predicate regarding some variable at time $t_i$ where $t \leq t_i < t + k$'; i.e., each integrity constraint restricts the value of some variable at time $t+k$, if certain conditions hold for the values of some variables at time steps $t, t+1, ..., t+k-1$.

The alarm condition $q$ is defined by a special rule whose body is purely disjunctive:

$$\bigvee_{1 \le i \le n_f} F_{it} \in RF_{ij_1} \vee \ldots \vee F_{it} \in RF_{ij_{k_i}} \rightarrow q_t$$

where $F_{it}$ is the value of the function $F_i$ which is determined by its supporting subset of variables at time $t$ only, and $RF_{ij}$ is a range of the function $F_i$.

A *monitoring algorithm* is an **on-line** algorithm that, given the past partial history line $< p_1, .., p_t >$ and the integrity constraints, determines $V_{t+1}$, the set of variables to be measured at the next time step.

A monitoring algorithm is correct if for any legal history line, the algorithm detects the alarm condition at the first time the alarm condition becomes true. The algorithm may terminate after detecting the alarm condition.

The *monitoring problem* is to construct correct efficient monitoring algorithms for any given set of variables, integrity constraints and alarm conditions. The efficiency of such an algorithm is measured by the total measuring cost.

**Example 2** We formalize Example 1 with the following set of rules.

$R_1: \; x_t < 90 \rightarrow x_{t+1} < 90 \vee 90 \le x_{t+1} < 100$
$R_2: \; 90 \le x_t < 100 \rightarrow 90 \le x_{t+1} < 100 \vee x_{t+1} \ge 100$
$R_3: \; x_t \ge 90 \rightarrow x_{t+1} \ge 100$

And
$x_t \ge 100 \rightarrow q_t$

In this simple example, it is easy to see that the following strategy is at least as good as any other. We assume that the time $t$ is initially 1.

1. Measure $x_t$.
2. If $x_t < 90$, wait until time $t+2$ and then goto (1). If $90 \le x_t < 100$, wait until time $t+1$ and then goto (1). If $x_t \ge 100$, report "alarm" and exit.

## 3.1 Problem Classifications

We can classify the problem into different categories. The variables are *independent* if the alarm functions are just variables themselves, i.e. $\{F_i = x_i, i = 1, .., n_f\}$, and each integrity rule involves only a single variable. Otherwise, we say the variables are *interdependent*. The integrity rules are *memoryless* if for each rule $R_j$ of the form $B_j \rightarrow H_j$, the expression $B_j$ only involves variable values at time $t$ and $H_j$ involves only variables at time $t+1$. Otherwise it has memory (finite by definition).

**Example 3** In Example 2, variables are independent (there is only one variable) and rules are memoryless. But if we add the following rule with memory:

$R4 : x_t < 90 \land x_{t+1} < 90 \land x_{t+2} < 90 \rightarrow x_{t+3} < 90$

i.e., if the number of mobile users stays below 90 for 3 successive time instants then it will stay below 90 for the next time instant (thus below 90 forever). Now the problem belongs to a different class.

**Example 4** There are two variables $x_1, x_2$. The range of $x_1$ is $(-\infty, 0], (0, 1], (1, \infty)$, the range of $x_2$ is just $(-\infty, 0], (0, \infty)$. We define the rules informally as:

- $x_1$ can never increase by more than 1 from $t$ to $t + 1$.
- if $x_{1,t} > 1$ then $x_{2,t+1} > 0 \lor x_{2,t+1} < 0$. Otherwise $x_{2,t+1} < 0$.
- the alarm condition $q$ becomes true when $x_2 > 0$.

  The measurement cost of both $x_1$ and $x_2$ is one unit.

Here the rules are memoryless, but the variables are interdependent, because $x_2$ at time $t + 1$ depends on $x_1$ at time $t$.

By introducing new auxiliary variables, we can eliminate memory as shown by the next Proposition. The proof is omitted here.

**Proposition 1** *Any problem containing rules with memory can be transformed into an equivalent problem with only memoryless rules.*

## 3.2 A Model of System Evolution

Given that the variable values fall into a finite number of ranges, we can define the variable configuration as distinct *States*. Let the state $S$ at time $t$ be the complete range distribution of variables $(x_1 \in R_{i,i_1}, x_2 \in R_{2,i_2}, ..., x_n \in R_{n,i_n})$. The total number of states is the cross product of the number of ranges of each variable.

**Definition 3.2**

A *State Transition Graph* can be constructed as follows:

- Each node in the graph represents a single state.
- There is a directed edge from node $S_i$ to $S_j$ iff it is possible to go from $S_i$ to $S_j$ in 1 time step ($t$ to $t + 1$).
- A node is a *terminal Node of level 0* with regard to a particular alarm function $F_i$, if the evaluation of $F_i$ based on the state triggers the alarm. A node is a *terminal node of level* $k$ with regard to $F_i$ if all its successors are terminal nodes of level $k - 1$ with regard to the same function. It is not necessary to measure any more when a terminal node of level $k$ is reached, since we can determine that the alarm function will become true in exactly $k$ steps. We also call a terminal node of level 0 an *alarm node*.

If no rules are present, the graph will be complete. There is no edge from $S_i$ to $S_j$ only if some rule forbids it. Notice that if the variables are independent then the evolution of each variable is isolated, and we can consider the state transition graph for each individual variable separately.

The state transition graph describes the inherent transition of the system. However, at any point during the measurement, we may not have complete knowledge of all the variable values, only partial information is available, therefore we may only know a subset of the nodes that contains the actual state. Thus, any state of knowledge corresponds to the subset of node $SN$ in the State Transition Graph in which the present state is contained.

Measuring a variable $x_i$ will result in more specific knowledge, narrowing down the node set $SN$. Given the state of knowledge at time $t$ and the corresponding node set $SN_t$ at time $t$, even with no additional measurements, we can deduce that the set of nodes $SN_{t+1}$ that contains the system state is the set of nodes reachable from $SN_t$ in precisely 1 transition step. By repeating the deduction, we can find $SN_{t+k}$ at time $t+k$ for any $k$.

# 4 A Competitive Analysis Framework

## 4.1 The Optimal Algorithm

When a system contains only independent variables and memoryless rules, each variable can be considered separately. To find the optimal algorithm we only need to reduce the number of times each variable gets measured.

Given the current value of $x$, we can uniquely determine the corresponding state $S_{xt}$ in the transition graph for $x$. Let $l$ be the length of the shortest path from $S_{xt}$ to an alarm node (i.e., terminal node of level 0). Then for any $t'$ between $t$ and $t+l$, $x$ will not trigger an alarm.

An algorithm called $GREEDY$ will delay the measurement as much as possible, i.e., starting from the initial state, after measuring $x$ at time $t$, if $x$ has entered a terminal node of level $k$, we stop the measurements and report the alarm at time $t+k$. Otherwise, we wait until time $t+l$ to measure again.

**Theorem 1** *The $GREEDY$ algorithm is correct and optimal for any system with independent variables and memoryless rules.*

**Proof** The correctness follows from the observation that for any time step $t$, if $GREEDY$ does not measure $x$ then it is not possible for $x$ to satisfy the alarm condition at time $t$, since it is not possible to reach any alarm state regarding $x$.

We only need to show that $GREEDY$ measures each variable no more than any other correct algorithm. Let $A$ be any correct algorithm. For any single history line and any variable $x_i$, suppose $GREEDY$ measures $x_i$ at time instants $g_0, g_1, g_2..$ and $A$ measures at $a_0, a_1, a_2...$ with $a_0 = g_0 = 1$.

Let the state at time $i$ be $S_i$. We then show $a_k \leq g_k$ by induction. If $a_{k-1} \leq g_{k-1}$ then at time $a_{k-1}$ the shortest path to an alarm node is no more than $g_k - a_{k-1}$ since $S_{g_{k-1}}$ is reachable from $S_{a_{k-1}}$, and the shortest path from $S_{g_{k-1}}$ to an alarm node is $g_k - g_{k-1}$ by the definition of the $GREEDY$ algorithm, so $a_k \leq g_k$ since otherwise $A$ may not be correct.

Suppose $GREEDY$ does not terminate at time $g_k$, then $x_{g_k}$ is not in a terminal node of any level. Since $a_k \leq g_k$, $x_{a_k}$ can not be in a terminal node either, so $A$ can not terminate at $a_k$. Therefore $GREEDY$ measures $x$ no more than $A$ for the same history line. □

For more general classes of the monitoring problem no algorithm is optimal. In particular $GREEDY$ is non-optimal in the general case.

**Example 5** No Algorithm is optimal for all inputs on Example 4. Consider the case where $x_1 > 1$, $x_2 < 0$ at time $t$. Suppose $O$ is the optimal algorithm then it either measures $x_1$ at time $t+1$ or it does not ($x_2$ must be measured by any correct algorithm). If $O$ measures $x_1$, for the case where $x_1 > 1$ for any $t' > t$, it has to measure both $x_1$ and $x_2$ at each step. Another algorithm which measures only $x_2$ at each step yields a lower cost than $O$. If $O$ does not measure $x_1$ at time $t+1$, then in case $x_1 < 0$ at time $t+1$ and stays negative, $O$ will not be able to know that, so it measures $x_2$ at each time step. Another algorithm that measures $x_1$ every other time step thereafter has a lower cost.

Therefore in general no algorithm that is optimal for all input history lines may exist and we need to relax the optimality condition.

## 4.2 A competitive analysis formal framework

We now attempt to analyze the complexity of measurement in the spirit of competitive analysis of on-line algorithms introduced in [11]. First, the integrity rules can be augmented with transition probabilities. Then we introduce the following definitions, which are along the line of defining competitive ratio of on-line algorithms. The usual definition of the competitive ratio of an online algorithm is the ratio of the cost of the online algorithm to the cost of the best off-line algorithm. But here the best off-line algorithm can simply pick the first alarm condition on a time line without any measurement so it can not be used for comparison purpose. Instead, we use the algorithm that measures all the variables at all time steps as the off-line algorithm to compare with. We call this obvious algorithm, which has the highest cost, $O_b$.

**Definition 4.3**

The *cost ratio* of algorithm $A$ with respect to a specific problem is $r$ if there exist a constant $c$ such that $Cost(A) \leq r \times Cost(O_b) + c$ for any history line of the problem.

This is a worst case definition. If the transitional probabilities are known, we can also define the *average cost ratio*.

**Definition 4.4**
The *average cost* of algorithm $A$ on the history lines of length $n$ , $ACost_n(A)$ is the weighted average cost of $A$ on all history lines of length $n$ which do not contain an alarm state except perhaps for the last time step, according to the distribution.

The *average cost ratio* of algorithm $A$ on a given problem is $r$ if there exists a constant $c$ such that for any $n$, $ACost_n(A) \leq r \times ACost_n(O_b) + c$.

A monitoring algorithm is *worst case optimal* ($WCO$) if it is correct and its cost ratio is no larger than the cost ratio of any other correct algorithm. Notice that an algorithm that is $WCO$ does not necessarily have a lower cost than other algorithms on a particular history line. It merely has the best upper bound of the cost ratio. A monitoring algorithm is *average case optimal* ($ACO$) if it is correct and its average cost ratio is no larger than that of any other correct algorithm. Both can be shown to exist using game-tree analysis method. We omit the details here.

# 5 More Practical Algorithms

Although the procedures of finding $WCO$ and $ACO$ exist, they have double exponential complexity in the size of the problem. Thus, computing $WCO$ and $ACO$ even for moderate $n$ is infeasible. So we resort to finding algorithms that may not be provably optimal but try to minimize the cost ratio in a greedy fashion and can be derived in polynomial time. We use the *Cost Per Step (CPS)* criterion, which is maintained dynamically through the process of measurement.

**Definition 5.5**
For a variable $x_i$, at any time $t$, let $Cost_{x_i}$ be the total cost of measuring $x_i$ so far, then

$$CPS(x_i) = \frac{Cost_{x_i}}{t}$$

and $CPS_{total} = \sum_{i=1..n} CPS(x_i)$.

The $CPS$ value is essentially the measurement cost amortized over each time step. If we divide it by the sum of cost of measuring each variable, we get the ratio of the cost of the algorithm to the algorithm $O_b$ on the input history line up to a certain point in time. The *Expected Cost Per Step (ECPS)* value, which we will discuss later, will be based on the probabilistic distribution of the outcome of measurements.

## 5.1 The NEXT value and the Cost Per Step criterion

Recall from section 3 that any state of knowledge $K_t$ corresponds to a subset of node $SN_t$ in the State Transition Graph. So we can define $K_t$ alternatively as the subset $SN_t$ in which the present state is contained. Given the state of knowledge we define $NEXT(x_i)$ to be the next scheduled measurement time for variable $x_i$.

**Definition 5.6** $NEXT(x_i)$ is the next time instant that $x_i$ must be measured in order to have a correct algorithm. Given some state of knowledge $K_t$ and the corresponding node set $SN_t$, let $D_{min}$ be the minimum distance between any node in $SN_t$ and the set of alarm nodes with regard to $x_i$, then $NEXT(x_i) = t + D_{min}$. Notice that: (1) $NEXT(x_i)$ is monotonically increasing, and (2) if $K_t$ is more specific that $K'_t$, then $NEXT(x_i)$ derived using $K_t$ is no less than that derived using $K'_t$.

At time $t$, there are some variables that must be measured by any correct algorithm, we call this the $REQUIRED$ set. In addition we may choose to measure more variables. Although this carries extra cost, it will result in more specific knowledge, possibly increasing the $NEXT$ values. Therefore we need to find the best subset of variables to measure at time $t$.

We have defined the *Cost Per Step (CPS)* criterion, and we want to use this criterion to guide the execution of our algorithm. We must decide, for each time instant $t$ where some variables are required to be measured, the additional non-required variables that should be measured together. Let $V_t$ be the set of variables to be measured, then different $V_t$ results in different $NEXT(x_i)$ values after the actual measurement, and therefore different $CPS$ values. However it is not possible to predict the actual $NEXT(x_i)$ and $CPS$ values because the values of the variables in $V_t$ is not known in advance. Depending on different outcomes of measuring variables in $V_t$, different $NEXT(x_i)$ will be generated.

Before measuring $V_t$, we have a state of knowledge $K_t$, which was derived from previous time steps. Let $SN$ be the set of nodes in the state transition graph that corresponds to $K_t$. After measuring variables in $V_t$, we obtain a more specific knowledge $K'_t$, and a corresponding set of nodes $SN' \subset SN$. Let $SN'_1, SN'_2, ..SN'_k$ be all the possible subset obtained from all possible outcomes of measuring $SX$. For each subset $SN'_j$, we can derive $NEXT(x_i)_j$, and let $Cost_{x_i,NEXT(x_i)_j-1}$ be the cost of measuring $x_i$ up to time $NEXT(x_i)_j - 1$, then the predicated $CPS(x_i)$ value is

$$CPS(x_i)_j = \frac{Cost_{x_i,NEXT(x_i)_j-1}}{NEXT(x_i)_j - 1}, \text{ and } CPS(predict)_j = \sum_{i=1}^{N} CPS(x_i)_j.$$

is the overall predicted CPS value for this subset of nodes $SN'_j$.

Assume that we know the probability that $SN'_j$ is obtained from $SN$ after the measurement of $SX$, and call this probability $p_j$, then the expected $CPS$ of measuring $SX$ is:

$$EPCS(V_t) = \sum_{j=1}^{k} CPS(predict)_j * p_j.$$

$ECPS$ is a reasonable criterion to compare the performance of different subset of variables to measure. Our goal is to minimize $ECPS$ by selecting the optimal $SX$ for each time step.

We describe our algorithm next. For the sake of simplicity we first assume that all alarm functions are just individual variables. It is fairly easy to extend the result to general alarm functions.

### 5.2 Description of the algorithm

At a particular point in time, the up-to-date $NEXT$ values are maintained. Let $T = min(NEXT(x_i))$ for all $x_i$, our strategy is greedy in the sense it will not do any measurement before time $T$. However we may measure more variables than the $REQUIRED$ set at time $T$.

```
%%main()
  Let K= initial state of knowledge.
  For i=1 to n NEXT(X_i)=1;
  while (measurement not terminated)
  {
   Let T=min(NEXT(x_i)), i=1...n
   Deduce the state of knowledge at time T and the
   corresponding node set SN in State Transition Graph.
   Let SR be the set of all rules that may be satisfied at T
   Let SX be the set of variables involved in bodies of rules in SR

    Pick an optimal subset SX' of SX according to ECPS.
   The variable set V_T to be measured at T is then SX'+REQUIRED
   Wait until time T to measure variables scheduled.
   Update NEXT values according to the outcome.
    }
```

Next we determine how to pick the optimal subset $SX'$. First, it is not necessary to check every subset of $SX$. For example, if there is only one applicable rule, which involves $x$, $y$ and $z$ in the body, then it is not necessary to check $x, y$ without $z$. We thus use the satisfiable rules to guide the selection of variables. We define a hierarchical structure among the rules according to the *refinement* relation: Let $R_1$ and $R_2$ be two rules that are both potentially satisfiable given the state of knowledge, and $sx_1$ and $sx_2$ be the set of variables appearing in $R_1$ and $R_2$, then we say that $R_1$ *refines* $R_2$ if $sx_1 \supset sx_2$. Here the super set relation is strict, i.e., if $sx_1 = sx_2$ we don't consider this as a refinement relation. The motivation behind the definition is that if $R_1$ refines $R_2$ then we have a choice of either measuring only the variables involved in the body of $R_2$, or measuring the extra variables contained in the body of $R_1$. We can use a recursive procedure to extract all choices of $SX'$ that may impact the $NEXT$ values, and pick the one with the lowest $ECPS$. This is done based on a refinement graph of all the rules involved. A set called $CHOICES$ will be generated which represents all distinct ways of selecting $SX'$. We omit the details here. After obtaining the set $CHOICES$, we compute the $ECPS$ value of each subset of variables to be considered. We pick the subset of variables with the minimum $ECPS$ value. The algorithm just described can be reduced to $GREEDY$ if variables are independent. In fact $GREEDY$ tries to minimize $CPS$ by delaying measurement as much as possible.

## 6 Discussion

We proposed a formal framework for studying the monitoring problem. Our goal is to use previous knowledge about the variables, in order to reduce the monitoring cost. Except for the simplest case where the greedy algorithm is optimal, it is not feasible to find optimal cost algorithms. We thus used techniques from competitive analysis of online algorithms to compare between different monitoring algorithms. Many of the issues need to be further investigated. In particular, it is very important to find real network variables for which some well behaved constraints apply, and use our framework to save resources when monitoring a real network. Another interesting question is the best adaptation of the competitive analysis techniques for cases where the off-line algorithm may do without any cost. Instead of comparing to the obvious algorithm, as we did, one may try to expand the ideas of [4] and compare the monitoring cost to the cost of the best possible online algorithm for a specific history line. We expect that improvements to our methods can be made in the future, and the framework can be applied to many network management applications.

## Acknowledgments

Shamim Naqvi thanks T. Imielinski and Y. Saraiya for early collaboration on this set of ideas. The authors are grateful to J. Chomicki for suggesting several extensions and improvements to this work.

# References


[1] Ben-Artzi, A. Chandna, and U. Warrier, *Network Management of TCP/IP network: Present and Future*, IEEE management magazine, pages 35-43, 1990

[2] P. Dini, G. V. Bochmann, T. Koch and B Kramer, *Agent Based Management of Distributed System with Variable Polling Frequency Policies* 1997 International Symposium on Integrated Network Management, pp. 553

[3] J.E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Language, and Computation*, Addison-Wesley, 1979.

[4] Simon Kahan, *A Model for Data in Motion* Proc. of the 23th ACM symposium on theory of computing, (STOC91), pp. 267–277, 1991.

[5] Asawaree Kalavade and Pratyush Moghe, *Terminal QoS of Adaptive Applications and its Analytical Computation* IWQoS'97, Columbia University, New York, May 1997.

[6] Mazumdar, S., and Lazar, A.A., "Modeling the Environment and the Interface for Monitoring Integrated Networks," Presented at ICC'91, Philadelphia, May, 1991.

[7] Mazumdar, S., and Lazar, A.A., "Monitoring Integrated Networks for Performance Management," Presented at ICC/SUPERCOM'90, Atlanta, April 16 - 19, 1990.

[8] Pratyush Moghe and Michael Evangelista, *An Adaptive Polling Algorithm*, Proceedings of Network Operations and Management Symposium (NOMS 98), New Orleans, Feb 1998.

[9] R. B. Myerson, *Game Theory*, Harvard University Press, Cambridge, MA, 1991

[10] Marshall T. Rose, *The Simple Book: An introduction to Management of TCP/IP-based Internets*, Prentice Hall, 1991, ISBN 0-13-812611-9

[11] D. Sleator and R. Tarjan. Amortized Efficiency of List Update and Paging Rules. In *Communications of the ACM*, 28(2):202-208, 1985.



**Jia Jiao** received a B.S from TsingHua University, Beijing,China in 1990, an M.S from Rutgers University in 1993, and a Ph.D from Rutgers in 1996, all in Computer Science. His research interests include network management, network performance analysis, parallel and distributed systems, and algorithms.

**Shamim Naqvi** is head of the Network Computing Research Department at Bell Labs, Murray Hill, NJ. His research interests include Internet Telephony, network management, database systems and wireless computing.

**Danny Raz** received his doctoral degree from the Weizmann Institute of Science, Israel, in 1995. From 1995 to 1997 he was a post-doctoral fellow at the International Computer Science Institute, (ICSI) Berkeley, CA, and a visiting lecturer at the University of California, Berkeley. Since October 1997 he is with the Network and Service Management Research Department at Bell-Labs, Lucent technologies. His primary research interest is the theory and application of management related problems in IP networks.

**Binay Sugla** is currently Department Head, Network and Services Management Research Department at Bell Labs, Lucent Technologies. Presently he is working on tools and techniques for IP management. His past work produced tools like the Network Flight Simulator–a real-time simulator for networks, and CAPER–a concurrent application programming environment.