A&QT-R 2002 (THETA 13)
2002 IEEE-TTTC - International Conference on Automation, Quality and Testing, Robotics
May 23 – 25, 2002, Cluj-Napoca, Romania

Efficient Monitoring of Networks

Claudia Rus-Sturza

S.C. IPA Cluj, e-mail: claudia@automation.ro

Abstract. In many cases, data networks need to be monitored to ensure that they stay within acceptable parameters. The paper presents two techniques of monitoring combined in two basic monitoring algorithms. The performance of the algorithms is presented for 2 to 12 interfaces and for 3 different types of generated data: uniform change, normal change and self – similar.

Key words: monitoring, polling, event reporting, alarm conditions.

1. INTRODUCTION

Efficient management assumes having reliable information about the managed system. The only way to maintain such information at the management station is a continuous monitoring of the system parameters which affect management decisions.

The increasing complexity of managed systems and services provided by them generates a need for monitoring of more and more parameters. Minimizing the amount of monitoring related traffic is an important goal.

There are two types of monitoring:

1. **Statistical monitoring**
2. **Reactive monitoring**

1. In statistical monitoring, the management station derives some statistical properties of the network.
2. In reactive monitoringis the management station needs information about the network state in order to react to certain alarm conditions that may develop in the network.

Two basic techniques are used for network monitoring:

a) **polling**
b) **event reporting**

a) Polling is a process in which the management station sends requests to network elements in order to obtain the state information. Polling is done periodically with a fixed frequency. Periodical polling is expensive.
b) Event reporting is a process where a local event triggers a report, that is sent to the management station.

In [1], when one talks about monitoring and measurement, one assumes the polling model. But their polling is an inteligent one. In [2], both techniques are used for network monitoring, the polling and event reporting to obtain more efficient monitoring.

2. OVERVIEW OF THE APPROACH

In [1] the main idea is to use some *integrity constraints* to reduce the monitoring cost. Many states variables have constraints on their evolution. Given the present value of a state variable, these constraints limit the range of values the state variable can take at a future time. For example, assume that we need to monitore if the total number of phones that are present in a particular cell exceeds a certain threshold. It is clear that the total number of the phones in a particular cell at a future time is constrained by the sum of the number of phones in that cell at the present time and the number of phones in adjacent sites.

In the monitoring problem, there are a number of variables, each having an associated measurement cost. The problem is: *how to detect alarm conditions with a minimum of measurement cost.*

Ex1. 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 are hold:

* If $x < 90$ at time t, then $x < 100$ at time t+1.
* If $90 \leq x < 100$ at time t, then $90 \leq x < 100$ at t+1 or $x \geq 100$ at t+1.
* If $x \geq 100$, it will stay there.

Assuming that $x < 90$ at time t, we see that we can wait until time t+2 to measure again because at time t+1 alarm condition cannot be true. Hence, the knowledge on the evolution of the process can save monitoring cost.

3. GENERAL FRAMEWORK

Monitoring algorithm – the algorithm that decides which variables to measure, based upon values obtained in the past and the integrity constraints.

Such an algorithm is *correct* if it always detects alarm conditions, and is *optimal* if its cost is always no greater than the cost of any other correct algorithm.

For example, we formalize Ex1:

Ex2. R1: $x_t < 90 \rightarrow x_{t+1}+1 < 90$ OR $90 \leq x_{t+1} < 100$.
R2: $90 \leq x_t < 100 \rightarrow 90 \leq x_{t+1} < 100$ OR $x_{t+1} \geq 100$.
R3: $x_t \geq 100 \rightarrow x_{t+1} \geq 100$.

The following strategy is at least as good as any other:

1) Measure x.
2) If $x_t < 90$, wait until time t+2 and go to 1). If $90 \leq x_t < 100$, wait until time t+1 and then go to 1). If $x_t \geq 100$, report „alarm“ and exit.

In conclusion:

- Knowledge on the process can improve monitoring algorithms.
- Smart polling can save monitoring cost.

But, this knowledge, the integrity constraints here, may either hard to discover or do not exist, hence the applicability of the approach is limited. For solving this problem, a solution is to combine polling with event reporting. This is the main idea in [2] and here it is shown how to choose the right algorithm for the type of monitored data. In

particular their results show that for Internet traffic these algorithms can save more than 90% of the monitoring traffic.

4. BASIC MONITORING ALGORITHMS

We are given n real – valued variables $x_1, x_2,...,x_n$.
x_i has a positive cost c_i (representing the cost of measuring x_i at any time)
x_{it} – value of x_i at time t
Consider a global function $f(x_1, x_2,...,x_n)$
$f_t = f(x_{1t}, x_{2t},...,x_{nt})$ – value of f at time t
T – a global threshold
$f(x_1, x_2,...,x_n) \geq T$ – alarm condition.

The effectivenes of the presented algorithms will be compared to the algoritm *that measures all the variables at all time steps and then compute the value of f and compare it to T.* This algorithm is called Ob – obvious algorithm. It's clear that Ob is correct but it has the largest cost.

Here are presented two algorithms which are different based on the way the event reporting is initiated. In the first algorithm a simple value threshold is used, and in the second a threshold on the value rate change is used.

The first algorithm **Simple – value** is based on setting a local value threshold that triggers an event report. This threshold is set in a way that in order for the alarm condition to hold, at least one of the variables must exceed the threshold.

Formal description for Simple – value algorithm:
SVc – algorithm for the centralized monitoring process
SVn – algorithm for the distributed nodes
SVn: at each time t, if $x_{it} \geq T/n$ then send the value x_{it} to node 0.
SVc: at each time t, if received one or more reports then poll all other nodes for their values. If $f = \sum x_i \geq T$ then generates an alarm.

The cost of Simple – value algorithm:
p - probability that $x_i \geq T/n$ (we assume all variables are identical).
$\prod(1-p)$ - probability that none of these values is over the local threshold.
$1 - \prod(1-p) = 1-(1-p)^n$ - probability that at least one value was above the local threshold.
c - the cost of measuring x_i
The expected cost of algorithm is $E[Csv]=n*c*(1-(1-p)^n)$.
Cost ratio is $E[Csv]/E[Cob]=1-(1-p)^n$.
Obs. It's clear that E[Cob] - expected cost of obvious algorithm is n*c.

The second algorithm **Simple - rate**, instead of looking at a local value, it looks at the *changes* in the value.

Formal description for Simple – rate algorithm:
SRc – algorithm for the centralized monitoring process
SRn – algorithm for the distributed nodes
SRn: at each time t, if $x_{it} - x_{i(t-1)} > \delta$ then send the value x_{it} to node 0.

SRc: 1. Init: $t_m \leftarrow 0$; f=0
2. While (TRUE)
3. If $t \geq t_m$ OR report RECEIVED
4. f $\leftarrow$ POLLALL( )
5. if f>T
6. Report ALARM
7. else
8. $t_m \leftarrow t+[(T- \sum x_{it})/\delta*n]$

t_m – the next time to poll.

The cost of Simple – rate algorithm:

- is constructed from two components:
  the first – is due to the local events
  the second – is due to the centralized polling part.

The expected cost is:

$n(1-(1-Pr_s(\delta))^n) + \delta*n^2 /(T-EXP[f(x)])$ (see [2])

where $Pr_s(\delta)$ is the probability that the rate change of x_i is larger than δ.

- the optimal value for δ is the one satisfying:

$(1-Pr_s(\delta))^{n-1} dPr_s(\delta)/d\delta = (-1)/(T-EXP[f(x)])$ (see[2]).

5. RESULTS

They tested the performance of the algorithm for 2 to 12 interfaces anf for 3 different types of generated data: uniform change, normal change (see [2]) and self – similar (see [5]).

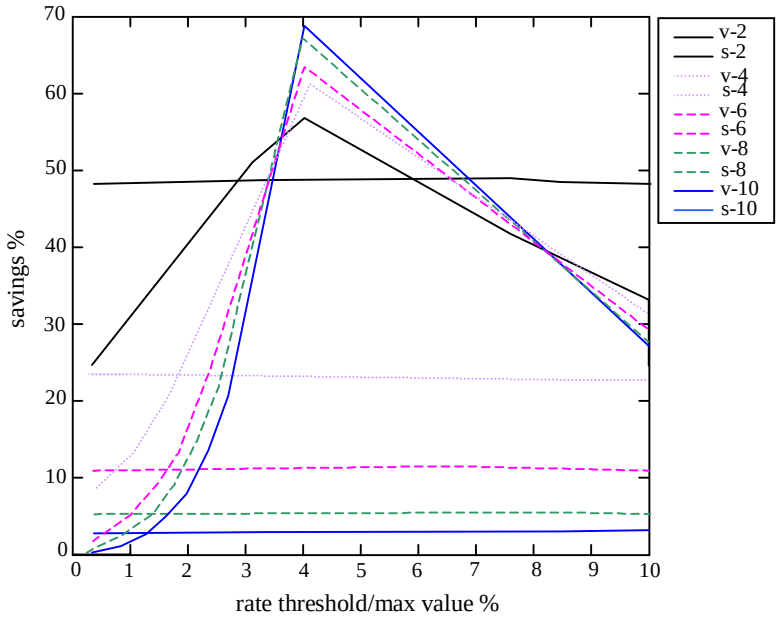

Fig. 1. Amount of traffic saving with uniform increment distribution data

v – l – indicates the performance of algorithm Simple – value with l variables
s – l – indicates the performance of algorithm Simple – rate with l variables

One can see that as expected, the amount of saving achieved by Simple – value algorithm decreases as the number of variables increases. As for Simple – rate algorithm, it is very clear from the figure that for any given n, there is a value of δ that achieves maximal saving.

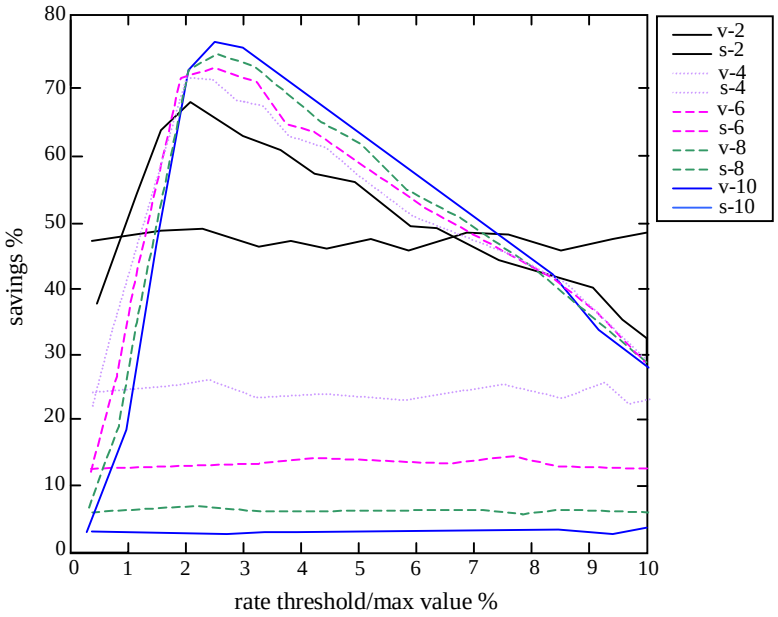

Fig. 2. Amount of traffic saving with normal increment distribution data

In the normal increment case, the two algorithms work almost the same as in uniform increment case.

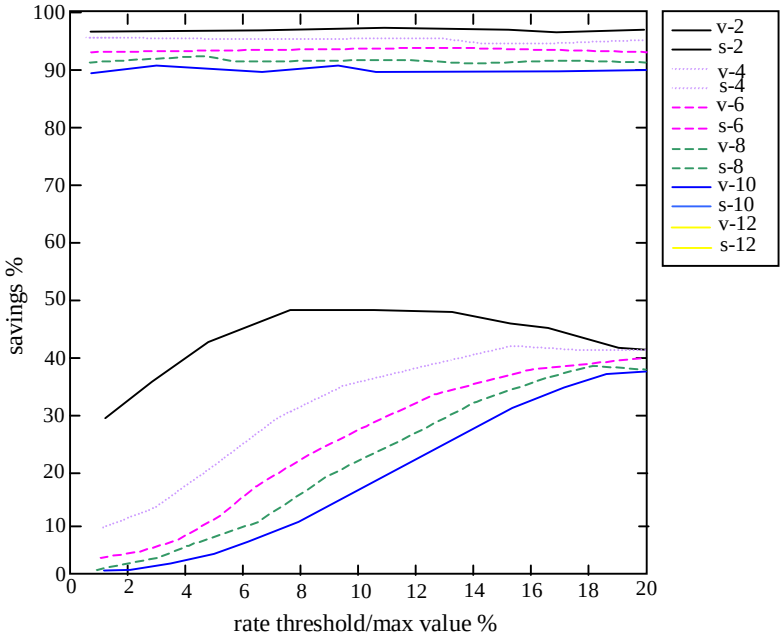

Fig. 3. Amount of traffic saving with self-similar data

In this case the behavior is quite different. Simple – value algorithm performes much better.

An improved – value algorithm is described and can be found in [2]. This algorithm can save 97% for the 5 minutes average data and 92% for the 1 minute average data.

5. CONCLUSIONS

The first work that proposed a model to study reactive monitoring problem was [1]. But they considered only polling, and did not consider event reporting. Their work assumes a certain knowledge of integrity constraints, that restrict the evolution of the variables. But, as we have mentioned before, in practice such constraints are either hard to find or do not exist. In [2] was introduced the idea of combining local events reporting with the global polling. There were presented two algorithms that combine these two techniques to save monitoring related traffic. This makes possible to use some of the results from [1] in a practical setting. The performance of these two algorithms was presented for 2 to 12 interfaces anf for 3 different types of generated data: uniform change, normal change and self – similar.

6. REFERENCES

[1] Jia Jiao, Shanim Naqvi, Danny Raz, and Binary Sugla, (May 2000), Toward Efficient Monitoring, *IEEE Journal on Selected Areas in Communications*, p. 723 - 732.
[2] Mark Dilman, Danny Raz, Efficient Reactive Monitoring.
[3] R.B. Myerson, (1990), Game Theory, Cambridge, MA: Harvard University Press.
[4] A. V. Gheorghe, (1990), Decision Processes in Dynamic Probabilistic Systems, Norwell, MA: Kluwer Academic.
[5] Walter Willinger, Murad Taqqu, Robert Shermann, and David V. Wilson, (1997), Self - similarity through high - variability: Statistical analysis of ethernet lan traffic at the source level, *IEEE/ACM Trans. On Networking*, p. 71 - 86,.