Adaptive Multi-task Monitoring System Based on Overhead Prediction [*]

Imed Lassoued, Chadi Barakat
Planete Project-Team, INRIA Sophia-Antipolis, France
{Imed.Lassoued, Chadi.Barakat}@sophia.inria.fr

ABSTRACT

The remarkable growth of the Internet infrastructure, the tremendous success of the Internet-based applications and their rapidly changing characteristics have made the management and monitoring of ISP networks a complex process. The design of a new monitoring system that takes into account the requirements of multiple monitoring tasks and variations in the traffic is becoming an inevitable trend. In this paper, we propose a monitoring system that adaptively adjusts its configuration according to network conditions and measurement accuracy. Our system relies on an optimization method consisting of: *(i)* overhead prediction to track short-term and long-term changes in the traffic, and *(ii)* a global weighted utility function to deal with multiple monitoring tasks. To assess the performance of our system, we propose an exhaustive experimental methodology for the evaluation of monitoring applications. We present our experimental platform and provide a global study of the operation of the proposed system and the impact of the different parameters on its behavior.

1. INTRODUCTION

Recently, there has been an increasing interest in passive monitoring and traffic analysis. In practice, NetFlow is the most deployed monitoring tool. However, while this solution lowers the cost of processing and storage resources, it clearly reduces the accuracy of measurements and incurs a loss of information. Some recent proposals try to find a solution to this discrepancy by providing monitoring infrastructures that coordinate monitoring responsibilities between different monitors e.g. [11, 5]. However, despite these available solutions, monitoring applications still present some shortcomings including the problem of overhead prediction and that of improving accuracy of multiple tasks.

The optimization of monitoring applications requires the estimation of overhead in order to find the appropriate configuration that keeps the overhead within a target value while providing the best possible accuracy. The majority of existing solutions use information about links' load and define the overhead as being the total number of packets that can be sampled in the entire network. Clearly, such approach leads to an inefficient use of resources since the load of links varies over time. We argue that an advanced module for overhead prediction can significantly increase the monitoring capabilities of a network and cope with variations in the traffic. Note that we define the overhead $\mathcal{O}$ as the total number of flow reports that are exported in the entire network (modeling processing and storage resources).

In this paper, we introduce an adaptive monitoring system that adjusts its configuration according to network conditions and measurement accuracy. Our system consists of four design primitives: *(i)* a global estimator module to minimize the amplitude of estimation errors, *(ii)* an overhead prediction based on an Exponential Weighted Moving Average filter to track long-term and short-term variations in the traffic, *(iii)* a global weighted utility function to deal with multiple monitoring tasks at the same time, and *(iv)* an optimization algorithm that configures monitors to address the tradeoff between resource consumption and accuracy of different tasks.

Given the lack of a universal experimental platform for monitoring applications, we present our own platform together with an exhaustive experimental methodology for the evaluation of our proposal. We have implemented a real experimental platform for traffic sampling and monitoring using real traffic traces and real monitoring tools. We have also carried out a global study of the performance of the proposed system and the impact of the different parameters on its behavior.

The rest of the paper is organized as follows. In Section 2, we present some related work. Then we present the system architecture in Section 3. A description of the experimental platform is given in Section 4. Evaluation results are presented in Section 5. Finally, conclusions and future work are given in Section 6.

[*]This research work is funded by the European Commission through the ECODE project (INFSO-ICT-223936) of the European Seventh Framework Programme (FP7).

2. RELATED WORK

The interest in passive monitoring for the understanding and diagnosis of core IP networks has grown at an astonishing rate. Currently, NetFlow [6] is the most widely deployed measurement solution by ISPs. However, this solution still presents some shortcomings, in particular the problem of setting sampling rates to low values to cope with the increasing trend in line speed. Some recent proposals have explored ways to improve NetFlow by providing network-wide monitoring infrastructures that distribute the work between different monitors. For instance, the authors in [11] present a network-wide approach that uses a hash-based flow selection to eliminate duplicate measurements in the network and a framework for distributing responsibilities across routers. However, these solutions deal with a single monitoring task and use a simple method to estimate overhead. The authors in [5] use link load information to configure the total number of packets that can be sampled in the network. Our architecture provides a solution to these problems that allows a further reduction of the load, while improving accuracy during the traffic estimation phase.

3. SYSTEM ARCHITECTURE

Given a list of measurement tasks $\mathcal{T}$ and an overhead constraint (Target Overhead $\mathcal{TO}$), our system adaptively adjusts its configuration. A configuration is a selection of packet sampling rates on the different interfaces of network routers (or monitors). This configuration is periodically updated as a function of a prediction of the overhead and in a way to optimize the accuracy of the considered measurement tasks. In this section, we present the architectural ideas behind our system.

3.1 Global estimator

To minimize the amplitude of estimation errors, we implement a global estimator in the NetFlow collector. For each monitoring task $T_i \in \mathcal{T}$, this global estimator takes as input the local estimations of T_i, $(\hat{T}_{ik})_{k\in\mathcal{R}}$, calculated from the reports sent to the collector by the different monitors; $\mathcal{R}$ is the set of monitors. Then, it constructs the global estimator of the task T_i as a weighted sum of the different local estimators. This weighted summation is known to be the best linear combination in terms of mean square error [8],

$$\hat{T}_i = \sum_k \lambda_k \hat{T}_{ik} \quad \text{with} \quad \lambda_k = \frac{\frac{1}{Var(\hat{T}_{ik})}}{\sum_l \frac{1}{Var(\hat{T}_{il})}}. \qquad (1)$$

The weights λ_k are inversely proportional to the local estimation errors, which in turn are inversely proportional to the configured sampling rates. Thus, local estimates with smaller error variance have a larger impact on the global estimator than those with larger errors.

3.2 Overhead prediction

To limit the overhead of collected reports within $\mathcal{TO}$, we should be able to write analytically the expression of the overhead prediction during the period $t+1$, $\mathcal{O}^{(t+1)}$ as a function of the sampling rates $(p_k^{(t+1)})$. Then, we can use it as a constraint in the optimization algorithm.

Consider $N_k^{(t)}$, the number of 5-tuple flows crossing a monitor k whose sampling rate is $p_k^{(t)}$ during the period t. Let s be the size of a given flow, then the probability that this flow is sampled is equal to $1-(1-p_k^{(t)})^s$, which can be approximated by $p_k^{(t)}.s$ for small $p_k^{(t)}$. The number of sampled flows $M_k^{(t)}$ can then be approximated by $p_k^{(t)}.S_k^{(t)}.N_k^{(t)}$, where $S_k^{(t)}$ is the mean size of 5-tuple flows. Hence, the number of flows crossing the monitor k at the instant t can be estimated by:

$$\hat{N}_k^{(t)} = \frac{M_k^{(t)}}{p_k^{(t)}.S_k^{(t)}}. \qquad (2)$$

This estimator of the number of 5-tuple flows only uses the most recent observation while our objective is to track short-term as well as long-term variations in the traffic. To this end, we use the Exponentially Weighted Moving Average (EWMA) which is a memoryless moving average whose weights are exponentially decreasing from more recent historical samples to older ones.

Let $\overline{N}_k^{(t)}$ be the smoothed version of the number of flows across monitor k during period t,

$$\overline{N}_k^{(t+1)} = \alpha N_k^{(t)} + (1-\alpha)\overline{N}_k^{(t)}, \qquad (3)$$

where $N_k^{(t)}$ can be approximated by the estimator given in equation (2). $\alpha = \frac{2}{(n+1)}$ is the smoothing factor where n is the window length over which we smooth the traffic. This factor allows us to choose the time scale β of tracking variations in the traffic. For instance, if we want to track changes on hourly scale (i.e. $\beta = 3600s$), we calculate the window length $n = \frac{\beta}{d}$, where d is the period of configuration updates.

Let $c_k^{(t)}$ be the total number of sampled packets in monitor k at period t. The estimator of the total number of packets crossing the monitor k that maximizes the likelihood is known to be: $\hat{C}_k^{(t)} = \frac{c_k^{(t)}}{p_k^{(t)}}$. Hence, we can derive a first estimation of the mean flow size crossing the monitor k: $\hat{S}_k^{(t)} = \frac{\hat{C}_k^{(t)}}{\hat{N}_k^{(t)}}$. The smoothed version of the mean flow size using the EWMA is given by the following equation:

$$\overline{S}_k^{(t+1)} = \alpha \hat{S}_k^{(t)} + (1-\alpha)\overline{S}_k^{(t)}. \qquad (4)$$

Using equations (3) and (4), we can now give the analytical expression of the overhead prediction:

$$O_k^{(t+1)} = \overline{N}_k^{(t+1)}.\overline{S}_k^{(t+1)}.p_k^{(t+1)}. \qquad (5)$$

```
Data: Measured flows and packets in all monitors during
      the previous period $t$, $(M_k^{(t)})$ and $(c_k^{(t)})$.
The previous estimations: $\overline{N}_k^{(t)}$ and $\overline{S}_k^{(t)}$, and the
previous sampling rate vector $(p_k^{(t)})$.
Time scale $\beta$ and computation period $d$
Result: The expression of the overhead prediction $\mathcal{O}^{(t+1)}$
begin
    $n \leftarrow \frac{\beta}{d}$; $\alpha \leftarrow 2/(n+1)$ ;
    foreach  $k \in \mathcal{R}$ do
        calculate $\hat{N}_k^{(t)} = \frac{M_k^{(t)}}{p_k^{(t)}.S_k^{(t)}}$ ;
        \\Estimate the number of 5-tuple flows.
        $\overline{N}_k^{(t+1)} \leftarrow \alpha\hat{N}_k^{(t)} + (1-\alpha)\overline{N}_k^{(t)}$ ;
        calculate $\hat{C}_k^{(t)} = \frac{c_k^{(t)}}{p_k^{(t)}}$ ; calculate $\hat{S}_k^{(t)} = \frac{\hat{C}_k^{(t)}}{\hat{N}_k^{(t)}}$ ;
        \\Estimate the mean size of 5-tuple flows.
        $\overline{S}_k^{(t+1)} \leftarrow \alpha\hat{S}_k^{(t)} + (1-\alpha)\overline{S}_k^{(t)}$ ;
        \\Derive the expression of the overhead
        \\prediction during the next period $t+1$.
        $O_k^{(t+1)} \leftarrow \overline{N}_k^{(t+1)}.\overline{S}_k^{(t+1)}.p_k^{(t+1)}$ ;
    end
    \\Derive the global overhead prediction expression.
    $\mathcal{O}^{(t+1)} = \sum_k O_k^{(t+1)}$ ;
    return $\{\mathcal{O}^{(t+1)}\}$
end
```

Figure 1: Overhead prediction method.

```
Data: The expression $L(P^{t+1})$ and the previous
      sampling rate vector $P^t = (p_k^t)$
Result: The new sampling rate vector $P^{t+1}$
begin
    $i \leftarrow 0$ ; $\overrightarrow{P}_i \leftarrow P^t$ ;
    while $i < max_{iteration}$ and $min_{found} == false$ do
        Evaluate the gradient vector $\nabla L(\overrightarrow{P}_i)$ ;
        Evaluate the Hessian matrix $H(\overrightarrow{P}_i)$ ;
        \\Compute the search direction
        $\overrightarrow{S}_i \leftarrow -\nabla L(\overrightarrow{P}_i)(H(\overrightarrow{P}_i))^{-1}$ ;
        \\Calculate the best step $\sigma_b$ value using line
        \\search algorithm
        $\sigma_b \leftarrow lineSearch(\overrightarrow{P}_i + \sigma\overrightarrow{S}_i)$ ;
        \\calculate the next point
        $\overrightarrow{P}_{i+1} \leftarrow \overrightarrow{P}_i + \sigma_b\overrightarrow{S}_i$ ;
        \\Perform the termination test for minimization
        if $L(\overrightarrow{P}_{i+1}) - L(\overrightarrow{P}_i) < precision$ then
            $min_{found} \leftarrow true$ ;
        end
        $i \leftarrow i+1$ ;
    end
    return $\{\overrightarrow{P}_i\}$
end
```

Figure 2: Optimization procedure.

This overhead prediction is no other than the smoothed version of the number of sampled flows.

The overhead prediction method works as follows. For each monitor k, first we look for initial values for the number of flows $\overline{N}_k^{(t)}$ and the mean flow size $\overline{S}_k^{(t)}$. To do so, we can use values of the same period of the last week or the last day. Then, we start using the collected traffic to update estimators and predict the overhead. For this we use the algorithm presented in Figure 1 and implementing the EWMA filter for prediction according to equation (5). The smoothing factor α plays a crucial role in the overhead prediction. In fact, using short time scale can disrupt the system with unnecessary details specific to a particular observation period while the use of a large time scale can lead to the loss of important changes in the traffic. We have to find the suitable time scale that addresses the tradeoff between these two extremes.

3.3 Optimization method

The optimization method is motivated by the need to coordinate responsibilities across the different monitors to improve the accuracy. This method is fed by the list of tasks T_i, their associated weights γ_i, and the normalized variance of the global estimation of each task $\hat{T}_i$, $Var(\hat{T}_i)$. Our objective is to find the optimal sampling rate vector that minimizes the utility function:

$$U = \sum_i \gamma_i Var(\hat{T}_i), \qquad (6)$$

under the following constraints:

$$\mathcal{O} \leq \mathcal{TO} \qquad (7)$$

$$p_k \leq SR_{max} \quad ; \quad \forall k \in \mathcal{R} \qquad (8)$$

$$p_k \geq SR_{min} \quad ; \quad \forall k \in \mathcal{R} \qquad (9)$$

SR_{min} and SR_{max} are respectively the minimum and maximum sampling rate values.

To solve this constrained optimization problem we define the corresponding Lagrangian:

$$L = U + \delta(O - TO) + \sum_k a_k(p_k - p_{max}) + \sum_k b_k(p_{min} - p_k).$$

(δ, a_k, b_k) is the set of Lagrange multipliers that enforce the satisfaction of the constraints (7), (8) and (9). We solve this Lagrangian by an iterative procedure using the Newton method (refer to [4], Chapter 9.5). The idea of the method, summarized in Figure 2, is as follows. We start with an initial guess of the optimal sampling rate vector. Then, at each iteration, we use the Newton method to go into a better direction while using a sophisticated line search algorithm to find the best step value σ_b. We continue until, we either reach the global minimum or we exceed the maximum number of iterations.

4. EVALUATION METHODOLOGY

In order to evaluate the performance of our system, we have developed our own experimental platform [9]. This platform has the following main features: *(i)* it is fed by real traffic captured on a transit link then spread and played over an emulated network topology, *(ii)* it includes real NetFlow-like tool for traffic monitoring on all router interfaces of the emulated topology, and *(iii)* it implements the central unit as it should be in reality.

4.1 Experimental platform

As shown in Figure 3, our experimental platform, is composed of three services: *(i)* the traffic emulation service, *(ii)* the traffic monitoring and sampling service, and *(iii)* the data collection and analysis service. Routers can be either virtual nodes connected by virtual

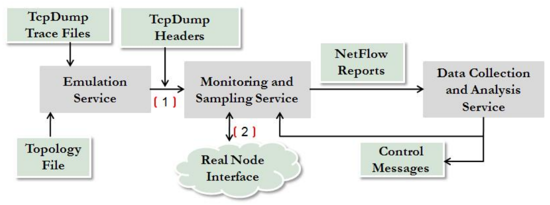

Figure 3: Experimental platform.

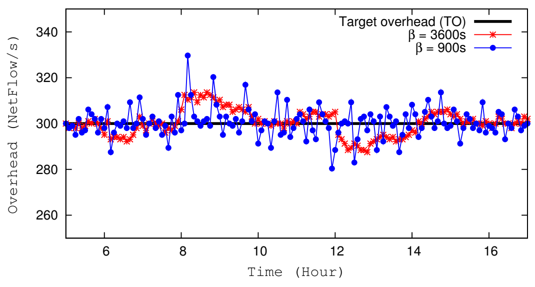

Figure 4: Resulting overhead vs. time using two time scales to track variations in the traffic.

links, or real routers connected by real links. The first service is responsible of generating the emulated traffic across the network routers. The second service implements packet sampling and flow monitoring a la NetFlow on each router interface. The later functionality is provided by SoftFlowd [3], an open source free software capable of NetFlow measurements in high speed networks. The third service mainly consists of the Flowd tool in the SoftFlowd package.

SoftFlowd requires network traffic in the TcpDump format. Unfortunately, obtaining real traffic data from an entire backbone network is a hard issue. To cope with this limitation, we proceed in the following way. We first seek unsampled packet level traces collected on high speed transit links. We consider for this study the ones coming from the Japanese MAWI project [2]. We parse the traces for the IP prefixes, and we dispatch them over the Autonomous Systems (ASes) connected to the edge routers of the emulated topology. The dispatching is done randomly according to some predefined weights that determine the importance of each stub AS. Our system allows to define the length of the prefix as a function of the granularity of the dispatching we want to achieve. For this work, we consider the /16 prefix as the basic unit for IP address assignment to ASes.

Once addresses are allocated, the packets in the TcpDump trace are split accordingly between the different ASes connected to the emulated topology. Shortest routes are calculated, then packets in the TcpDump trace are associated to the different monitors over their respective paths across the emulated network with the correct timestamps derived from the trace. SoftFlowd instances sample then packets on each router interface, form flows and send them back to the central collector. This sampling and monitoring is done in parallel on all network router interfaces.

4.2 Scenario description

Our platform requires the definition of a network topology over which it dispatches and replays a real traffic. We chose to study the performance of our system by emulating Geant, the European Research network [1].

As target, we consider two accounting applications or tasks: *(i)* traffic matrix estimation (T_1) which consists in jointly estimating edge-to-edge flow sizes. A flow F_i is the set of 5-tuple flows that share the same AS source and AS destination; *(ii)* large AS traffic estimation (T_2) where we aim to estimate the volume of the greediest ASes (those contributing to more than some percentage of the total traffic, we take 7% as example). Except when explicitly mentioned, we assign to these tasks respectively the equal weights $\gamma_1 = \gamma_2 = 0.5$.

5. VALIDATION RESULTS

In this section, we study the efficiency of our adaptive solution. We then provide a global sensitivity analysis to study the importance of the different parameters.

5.1 System efficiency

In this section, we aim to address the performance of our system using real experiments over our platform. For this experiment, we set the update period d to 5 minutes, the time scale β to 3600s, the minimum sampling rate SR_{min} to 0.0005 and the maximum one SR_{max} to 1. The $\mathcal{TO}$ is set to 300 NetFlow-records/s.

In order to evaluate the performance of the overhead prediction method, we plot in Figure 4 the evolution of the measured overhead (exported NetFlow records) over time. We observe that the system profits from the available resources to provide the best possible accuracy. For the two considered time scale values, the system maintains the overhead around the $\mathcal{TO}$. The use of a small time scale ($\beta = 900s$) leads to an oscillating behavior of the overhead since the system tracks more details and fine-grained changes in the traffic. On the contrary and because of a coarser aggregation of NetFlow reports, tracking changes on hourly scale leads to a more stable behavior of the overhead.

Figure 5 shows the value of the average mean relative error for different $\mathcal{TO}$ values. We notice the impact of the $\mathcal{TO}$ on the global estimation accuracy. There is a clear reduction of the overall measurement error when the target overhead increases. The error drops from 0.284 for a $\mathcal{TO}$ equal to 100 NetFlow-records/s, to 0.0279 for a $\mathcal{TO}$ equal to 400 NetFlow-records/s. Indeed, the system tries to provide the best possible accuracy given a monitoring constraint ($\mathcal{TO}$). Allowing

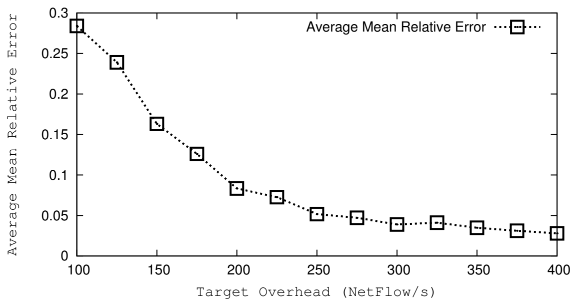

Figure 5: Average mean relative error vs. ($\mathcal{TO}$).

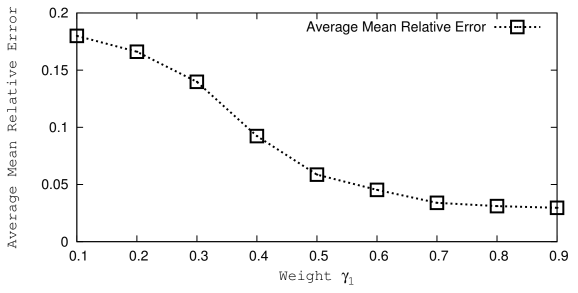

Figure 6: Average mean relative error of T_1 vs. the weight γ_1.

more overhead (resources) gives the system the possibility to increase some sampling rates and to collect and export more data looking for better estimation. The main strength of our system is that it is able to cope with any $\mathcal{TO}$ value and provides for this value the best configuration of monitors.

Now, we want to study the performance of this global optimization as a function of the weights assigned to each task T_i. We run experimentations using two tasks T_1 and T_2 while changing their assigned weights (γ_1 and $\gamma_2 = 1 - \gamma_1$). We plot in Figure 6 the average mean relative error of T_1 as a function of its assigned weight γ_1. As expected, γ_1 has a clear impact on the accuracy of T_1. The mean relative error varies between 0.0296 and 0.18 for different values of γ_1. For instance, setting γ_1 at a large value (i.e. setting γ_2 at a low value) gives more importance to T_1 in the optimization procedure and decreases the impact of the estimation error of T_2 on the global accuracy. In this manner, the optimal solution that maximizes the global accuracy is the one that satisfies especially the accuracy of T_1. This result confirms the flexibility of our method where the operator can set the weights according to the importance of tasks. By setting the weights to different values, one can achieve high accurate measurements for important tasks at the expense of less important ones.

5.2 Global sensitivity analysis

In the previous part, we have studied the performance of our system and the influence of some parameters on results. Yet, we use global sensitivity analysis to characterize, qualitatively or quantitatively, what impact an input parameter has on the system output and how it compares with the impact of the other parameters. Fourier Amplitude Sensitivity Test(FAST) [7] is considered to be one of the most efficient methods in sensitivity analysis. Among its advantages are: fast implementation, possibility to deal with nonmonotonic models, arbitrary large variations in input parameters, and no need for the knowledge of the mathematical model.

The main idea of FAST is to assign to each parameter a distinct integer frequency (characteristic frequency). Then, for a specific parameter, the variance contribution can be singled out of the model output with the help of the Fourier transformation. Specifically, let us consider a nonlinear model $y = f(x_1, x_2, ..., x_n)$ where x_n are parameters. The search function must let the parameter x_i to oscillate with frequency ω_i. For instance, the authors of [10] have proposed the search function $x_i = \frac{1}{2} + \frac{1}{\pi}\arcsin(\sin(\omega_i s))$, which is a particular case of a more general search function

$$x_i = F_i^{-1}\left(\frac{1}{2} + \frac{1}{\pi}\arcsin(\sin(\omega_i s))\right), \qquad (10)$$

where $F_i^{-1}(\cdot)$ is the inverse cumulative distribution function for x_i. To make more efficient use of the model evaluations, the authors of [10] have suggested the following modification

$$x_i = \frac{1}{2} + \frac{1}{\pi}\arcsin(\sin(\omega_i s + \varphi_i)), \qquad (11)$$

where φ_i is a random phase-shift chosen uniformly in the interval $[0, 2\pi]$. The model output becomes a periodic function with period 2π. Thus, we can represent the model with a Fourier series,

$$y = f(x_1, x_2, ..., x_n) = A_0 + \sum_{k=1}^{\infty}[A_k \cos(ks) + B_k \sin(ks)].$$

If we denote a sample of size N as $S = \{s_1, s_2, ..., s_N\}$, then, using either (10) or (11) as a search function, we can obtain the sampled values of the parameters $X_i = \{x_{i1}, x_{i2}, ..., x_{iN}\}$, and the discrete Fourier transform coefficients $A_0 = \frac{1}{N}\sum_{j=1}^{N} f(s_j)$,

$$A_k = \frac{2}{N}\sum_{j=1}^{N} f(s_j)\cos(s_j k) \text{ and } B_k = \frac{2}{N}\sum_{j=1}^{N} f(s_j)\sin(s_j k),$$

where $f(s_j) = f(x_{1j}, ..., x_{nj})$ and $k = 1, ..., (N-1)/2$.

The variance of the model output can be decomposed into variance components at the integer frequencies,

$$V = \frac{1}{2}\sum_{k=1}^{(N-1)/2}[A_k^2 + B_k^2],$$

Table 1: Parameters of the experiment.

Parameter	symbol	range	impact
Target Overhead	$\mathcal{TO}$	$[20, 500]$	0.431
Time scale	β	$[60s, 7200s]$	0.1147
Computation period	d	$[60s, 300s]$	0.0234
Min sampling rate	SR_{min}	$[0, 0.01]$	0.0876
Max sampling rate	SR_{max}	$[0.01, 1]$	0.0935

By summing the spectrum values $\Lambda_k = [A_k^2 + B_k^2]/2$ for the characteristic frequencies ω_i and their higher harmonics, the partial variance in model output arising from the uncertainty of parameter x_i, V_i, can be estimated by $V_i = \sum_p \Lambda_p \omega_i$, where $p\omega_i \leq (N-1)/2$. The ration V_i/V measures the contribution of parameter x_i. This ratio is also referred to as the first-order sensitivity index [12]. Because the characteristic frequencies are integers, there will be an aliasing effect if one frequency is a linear combination of the others. Therefore, a frequency set is free of interferences to an order M if

$$\sum_{i=1}^{n} a_i \omega_i \neq 0, \quad \sum_{i=1}^{n} |a_i| \leq M + 1,$$

where a_i is an integer and M is a design integer (usually 4 or 6). In order to avoid the interference effect the maximal value of p in calculating V_i should be M. In [7] the authors have proposed the following empirical formula for calculating the characteristic frequencies free of interference up to order $M = 4$: $\omega_1 = \Omega_n$, and $\omega_i = \omega_{i-1} + d_{n+1-i}, \quad i = 2, ..., n.$. The parameters Ω_n and d_k can be found in a table provided in [7].

We have applied the method FAST to our system to characterize the impact of the different parameters used in experimentations on results. Table 1 summarizes the different evaluated parameters with their ranges. The last column presents the impact of each parameter on the system output. It is immediately noticed that the parameter having most important impact on the system output is the target overhead $\mathcal{TO}$. In our system, the $\mathcal{TO}$ is a monitoring constraint set by the operator and can be changed to achieve a given accuracy. We also observe the important impact of the time scale β parameter. Thus, it is so important to set this parameter at a suitable value in order to address the tradeoff between the long-term and short-term variations and to improve results. The other parameters have a light impact on the behavior of the system (less than 10%).

6. CONCLUSIONS

In this paper, we have presented an adaptive monitoring system that coordinates responsibilities between the different monitors in order to achieve the best possible accuracy while respecting monitoring constraints. We have developed an experimental platform to evaluate our system. Results proved the ability of our system to keep the resulting overhead around a target value. We also demonstrated that our system is practical: it provides an efficient method to achieve multiple monitoring objectives using a weighted utility function and it relies on a flexible method to track variations in the traffic according to an adaptable time scale. Moreover, we provided a global study of the impact of the different parameters on the behavior of the system. Our ongoing work is centered on the validation of our system with more applications and on the distribution of the control.

Acknowledgment

We thank Konstantin Avrachenkov for suggesting the sensitivity analysis used in Section 5.2.

7. REFERENCES

[1] Geant: The european research and academic backbone. `http://www.geant.net/`.

[2] Mawi working group traffic archive. `http://tracer.csl.sony.co.jp/mawi/`.

[3] Softflowd flow-based network traffic analyser. `http://www.mindrot.org/projects/softflowd`.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] G. Cantieni, G. Iannaccone, C. Barakat, C. Diot, and P. Thiran. Reformulating the monitor placement problem: Optimal networkwide sampling. In *Proc. of CoNeXT*, 2006.

[6] Cisco. Netflow services and applications. *White Paper*, 2000.

[7] R. Cukier, J. Schaibly, and K. Shuler. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. analysis of the approximations. *The Journal of Chemical Physics*, 1975.

[8] N. Duffield, C. Lund, and M. Thorup. Optimal combination of sampled network measurements. In *IMC 2005*, 2005.

[9] A. Krifa, I. Lassoued, and C. Barakat. Emulation platform for network wide traffic sampling and monitoring. *TRAC*, 2010.

[10] A. Saltelli, S. Tarantola, and K. P.-S. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41:39–56, 1999.

[11] V. Sekar, M. Reiter, W. Willinger, H. Zhang, R. Kompella, and D. Andersen. cSamp: A system for network-wide flow monitoring. In *Proc. 5th USENIX NSDI*, 2008.

[12] I. Sobol. Sensitivity estimates for nonlinear mathematical models. *Math. Model. Comput. Exp.*, 1:407–414, 1993.