Hidden Anomaly Detection in Telecommunication Networks

Aurore Junier, Anne Bouillard, Ronot Benoit

Hidden Anomaly Detection in Telecommunication Networks

Anne Bouillard , Aurore Junier , Benoit Ronot

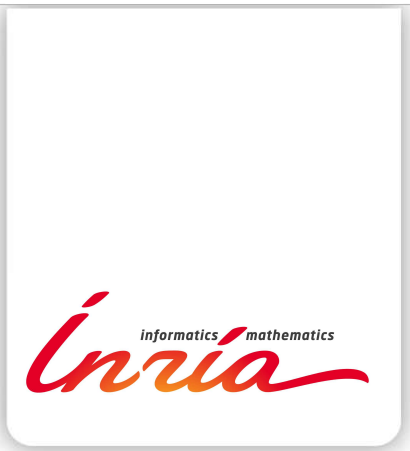
RESEARCH REPORT

N° 7979

May 2012

Project-Teams Distribcom and TREC

ISSN 0249-6399 ISRN INRIA/RR--7979--FR+ENG



Hidden Anomaly Detection in Telecommunication Networks

Anne Bouillard *, Aurore Junier †, Benoit Ronot ‡

Project-Teams Distribcom and TREC

Research Report n° 7979 — May 2012 — 22 pages

Abstract: Nowadays one of the challenges of telecommunication systems management is to detect in real-time unexpected or hidden malfunctions in extremely complex environments. In this report, we present an on-line algorithm that performs a flow of messages analysis. More precisely, it is able to highlight hidden abnormal behaviors that existing network management methods would not detect. Our algorithm uses the notion of constraint curves, introduced in the Network Calculus theory, defining successive time windows that bound the flow.

Key-words: Network management, General management, Fault management, Network Calculus theory, Fault prediction, hidden anomaly detection, Abnormal behavior detection.

* ENS / INRIA, anne.bouillard@ens.fr
† INRIA / IRISA, aurore.junier@inria.fr
‡ Alcatel-Lucent Bell Labs,benoit.ronot@alcatel-lucent.com

RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Détection d'anomalies cachées dans les réseaux de télécommunications

Résumé : Aujourd'hui l'un des challenges du management des réseaux de télécommunication est de détecter en temps réel les mauvais fonctionnements, inattendus ou cachés, dans des environnements extrêmement complexes. Dans ce rapport, nous présentons un algorithme en ligne qui procède à une analyse de flots de messages. Plus précisément, il est capable de détecter des comportements cachés anormaux que les méthodes existantes de management de réseaux ne détecteraient pas. Notre algorithme utilise la notion de courbes de contraintes, introduites par la théorie du Network Calculus, définissant des fenêtres temporelles successives qui bornent le flot.

Mots-clés : Management de réseau, Management général, Management de fautes, Théorie du Network Calculus, Prédiction de fautes, Détection d'anomalie cachées, Détection de comportement anormaux.

1 Introduction

The all IP convergence in telecommunication networks and the fact that operators always add new functionality and new services to their network bring a vertical and horizontal multilayer matrix of data exchange with an extreme complexity of synchronization and correlation. The emerging of new technologies that ease the network management and improve its efficiency, such as autonomous networking [12], and proactive care concept [15], will not stop this evolution.

The software implications in this matter of fact are the main factor of issues and malfunctions that affect the network performance. Errors in nodes or services configuration are the most frequent causes of malfunction, but errors in the code of processes also create hidden and unpredictable issues that we have to take into account with these new technologies. It is admitted that the industry average error rate is about 15-50 errors per thousand lines of code (KLOC) of delivered code [10]; the NASA has a defect density of 0.004 bugs/KLOC but this has a cost of $850.0/LOC [16]; telecommunication networks are then subject to hidden bugs in their components that lead into unpredictable network behavior [1]. Errors like memory unreleased by a process need a node restart 3 or 4 times per year to fix the resulting issue, will not be investigated by the management team.

Our assumptions are that hidden anomalies, provided by missed errors in network process design and coding, cannot be detected by actual management teams and systems. It clearly appears that new concepts and their associated algorithms are essential to manage this evolution. Such algorithms will have to face strong operational constraints: real-time computation, multiple context adaptation, trend setting for proactive or predictive technologies. They also have to deliver their results to both human and machine interfaces.

In this article, we present an algorithm that will fit such constraints and that can deliver a stability indicator of the exchanges between the nodes or the processes of the network. Our algorithm is dedicated to monitor data exchanges flows between processes to detect anomalies that cannot be explained by the normal behavior of the network. Each flow of data based on messages exchanges in modern networks is constrained by the design of their original process. An issue will then be detected if the flow does not respect this design. However, process configuration changes or occurring issues will influence these exchanges and must be taken into account in such detection.

Our algorithm uses the Network Calculus theory [2, 7] to define constraint linear curves on an arrival flow. In the classical Network Calculus, a flow satisfies some minimal and maximal constraints that frame the flow at all time and are usually given as an assumption of the flow. Here we carry out the other way round and try to find simple curves that bound a flow. As the flow is analyzed on the fly and then not known in advance, we allow the constraints to change to fit its variations. More precisely, the main contribution consists in modeling and predicting a time-window for the next message of a flow. If it does not belong to that time-window new constraints are defined. Issues are then detected when the slope and its trend present too many variations from the theoretical model. Once designed, we confront our algorithm to the monitoring of OSPF flows captured from a virtualized 17 nodes network testbed. We run several issues scenario to be detected from basics to malicious cases.

The rest of the paper is organized as follows. In Section 2, we present existing methods for anomaly detection. Section 3 defines the technical framework, based on constraint curves, of our algorithm that is described in Section 4. Finally, we give some experimental results based on an operational testbed in Section 5, before concluding in Section 6.

2 State of the art

Many approaches have been developed to address the problem of congestion or failure detection. Here we present an overview of some existing detection techniques.

Methods based on data analysis Among those methods, some are based on a set of data from which the detection of abnormal behavior is computed. The *distributed index management method* (MIND), in [8], is a structure made of two logical components: a set of traffic monitors distributed and a query system that allows to separate data according to their anomaly class. *Principal Component Analysis* (PCA), [6, 14] is a mathematical process that converts a set of correlated observations in a set of uncorrelated values named *principal components* (PC). The set of PC forms the normal subspace, which is smaller than the initial one. PCA aims at detecting network flows anomalies based on that normal subspace. This method is presented, in [6], as an efficient way to solve this kind of problem, but [14] shows limitations of PCA application in anomalies detection: first it can detect a fault that is not one, then large anomalies can contaminate the normal subspace, and finally most of the time it is difficult to find the initial point of failure.

Method based on alarm and fault correlation In [3], a method based on the definition of an *alarm-fault causal graph* is presented. For each alarm a_i, a probability $p_{i,j}$ is assigned with respect to each fault F_j, which divides alarms in two classes: *significant alarms* and *candidate alarms*. Then if fault F_j happens, the set of significant alarms is found in the emitted alarms set, and the candidate alarms could be found in the emitted alarms set. The problem is that this method is not utterly automated and alarm management is complex, so an expert has to be involved to manage networks this way.

Specific method for a particular protocol Many articles have been written on synchronization of a particular protocol. For instance, in [5], the authors focused on the TCP protocol to evaluate its performance. This work builds a link utilization rate bound to avoid synchronization.

Methods based on statistical analysis These set of methods are mainly based on statistics. Generalized likelihood Ratio Approach in [17, 18]) is a method developed for discrete-time linear stochastic systems that are subjected to abrupt jumps. The objective is to construct an adaptive filtering that determines changes in the network. To do that, it contains a Kalmann-Bucy filter to model the dynamic system studied and then instantiate a secondary system that evaluates the measurements made by the filter. Netscope also uses statistics to characterize network links from end-to-end path measurement ([4]). It uses a combination of first and second-order moments plus an end-to-end measurement (classically seen as a system of linear equation). It collects information about the network with these computations and then uses them to characterize the minimum set of links which loss rates cannot be accurately computed.

Today, lot of methods exist to detect anomalies in networks. The main difficulty resides in the capacity of treating a large set of data to drag bad flows behaviors like congestion or failure, and to react quickly. Obviously, studying the complete set of information is not an option. Thus some methods developed use powerful mathematical tools but remain too complex for now. Other methods study a particular protocol, or use statistics. Here, we introduce a reactive method to detect anomalies in network for all kinds of protocols without using statistics.

3 Flows and constraints

In this section, we first present data flows. Then, we introduce the *arrival curve* functions, that frame the incoming traffic at a router. The objective is to study the characteristics of flows arriving at a router.

In today's protocols, each router transmits information (topology, routers' health, etc.) to its neighbors in order to keep them aware of network changes: we define data flows as sequences of messages. More formally, a flow is a non-decreasing sequence $(x_n)_{n\in\mathbb{N}}$, where $\mathbb{N} = \{0, 1, \ldots\}$, $x_0 = 0$ by convention and where x_n is the arrival date of the n-th message of the flow. We assume that $\lim_{n\to\infty} x_n = \infty$.

Graphically, this flow can be represented by the graph $(P_n)_{n\in\mathbb{N}} \in (\mathbb{R}_+ \times \mathbb{N})^{\mathbb{N}}$ where $\forall n \in \mathbb{N}$, $P_n = (x_n, n)$. This graph represents the cumulative number of incoming messages. An example of such a graph is represented on Figure 1.

Arrival curve is a fundamental notion in *Network calculus* ([7], [2]), a theory developed to compute deterministic performance bounds in networks. Arrival curves determine constraints on flows by bounding the number of packets that can arrive during any interval of time. We take here the concept of arrival curve and try to find, given a data flow, the constraints it satisfies. The definition of a constrained flow adapted to our framework is the following:

Definition 1 (Constrained flow)**.** *Let $\underline{\alpha}, \overline{\alpha} : \mathbb{R}_+ \to \mathbb{R}_+$ be two non-decreasing functions, $m < n$ be two non-negative integers. The flow (x_n) is* $(\underline{\alpha}, \overline{\alpha})$-constrained on the interval $[m, n]$ *if* $\forall m \leq m' \leq n' \leq n$,

- $\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m'$; *(the flow is lower-constrained)*
- $\overline{\alpha}(x_{n'} - x_{m'}) \geq n' - m'$ *(the flow is upper-constrained).*

The flow (x_n) is $(\underline{\alpha}, \overline{\alpha})$-constrained *if it is $(\underline{\alpha}, \overline{\alpha})$-constrained on the interval* $[0, +\infty]$.

In this article, we aim at finding constraints for the arrival flows on long intervals. We will use simple functions: affine for the upper constraints and the maximum of an affine function with 0 for the lower constraints. More precisely we will use the following functions and notations: $\overline{\alpha}_{\rho,\sigma} : t \mapsto \sigma + \rho t$ and $\underline{\alpha}_{\rho,T} : t \mapsto \max(\rho(t - T), 0)$. The variable ρ corresponds to the long term arrival rate of messages, and σ represents the maximal burst, which means that at most σ messages can arrive simultaneously. Finally, we define T as the maximal delay between two messages.

Lemma 1 (Uniqueness)**.** *Let $(x_n)_{n\in\mathbb{N}}$ be a flow of messages. Then*

- *either there exists no ρ for which there exist σ and T such that (x_n) is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained;*
- *or there exists a unique ρ, there exist σ, T such that (x_n) is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained. Moreover, for every $\sigma' \geq \sigma$ and $T' \geq T$, (x_n) is $(\underline{\alpha}_{\rho,T'}, \overline{\alpha}_{\rho,\sigma'})$-constrained.*

Proof. The last statement is trivial and a proof can be found in [7]. We only have to show that (x_n) cannot be $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$ and $(\underline{\alpha}_{\rho',T'}, \overline{\alpha}_{\rho',\sigma'})$-constrained with $\rho > \rho'$. If it were the case, then one should have $\forall n \in \mathbb{N}$, $\rho(x_n - T) \leq n \leq \sigma' + \rho' x_n$, but this cannot hold for $x_n > \frac{\rho T + \sigma'}{\rho - \rho'}$. □

Graphically, a flow $(x_n)_{n\in\mathbb{N}}$ is $\underline{\alpha}_{\rho,T}$-lower constrained (resp. $\overline{\alpha}_{\rho,\sigma}$-upper constrained) if $\forall n \in \mathbb{N}$, $\forall m > n$, P_m is above $\underline{\alpha}_{\rho,T}$ (resp. below $\overline{\alpha}_{\rho,\sigma}$) drawn from P_n (respectively denoted as $P_n + \underline{\alpha}_{\rho,T}$ and $P_n + \overline{\alpha}_{\rho,\sigma}$).

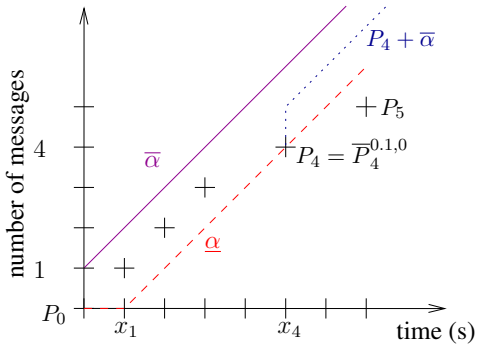


Figure 1: Data flow and constraints.

Example 1. *Consider the data flow of Figure 1. During* 30 *seconds, messages arrive every* 10 *seconds and then every* 20 *seconds. Consider the curves* $\overline{\alpha}_{0.1,1} : t \mapsto 1 + 0.1t$ *and* $\underline{\alpha}_{0.1,10} : t \mapsto \max(0.1(t-10), 0)$*. The flow is* $(\underline{\alpha}_{0.1,10}, \overline{\alpha}_{0.1,1})$*-constrained on* $[0,4]$ *but not on* $[0,5]$*, since* $5 - 0 = 5 \leq \underline{\alpha}_{0.1,10}(x_5 - x_0) = 6$*.*

Our aim is to guess arrival rates of the messages. For fixed σ and T, if the traffic is very regular, there exists ρ, such that $(x_n)_{n\in\mathbb{N}}$ is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained, one wishes to find ρ. If the traffic is not regular, the aim is to find the successive arrival rates of the messages. For this, given constraint curves, we need to introduce some messages of interest namely the first outgoing message and the critical messages.

Definition 2 (First outgoing message)**.** *Let* (x_n) *be a data flow,* $\underline{\alpha}$ *and* $\overline{\alpha}$ *be lower and upper constraints curves and* $n \in \mathbb{N}$*. The first outgoing message from* n *regarding* $\underline{\alpha}$ *and* $\overline{\alpha}$ *is*

$$o = \min\{p \geq n \mid (x_m) \text{ is not } (\underline{\alpha}, \overline{\alpha})\text{-constrained on } [n,p]\}.$$

We need to detect dates where (x_n) is not constrained any more by the current constraints.

For example, from the definition, checking that (x_n) is lower-constrained by $\underline{\alpha}_{\rho,T}$ implies testing that for each n and m with $m < n$, $\underline{\alpha}_{\rho,T}(x_n - x_m) \leq n - m$. In fact we do not need to study every element of (x_n) but those giving the strongest constraints. They are called *upper* and *lower critical messages.*

Definition 3 (Critical messages)**.** *Let* (x_n) *be a data flow,* $\rho, T, \sigma \in \mathbb{R}_+$ *and* $m < n \in \mathbb{N}$*. Suppose that* (x_n) *is* $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$*-constrained on* $[m,n]$*.*

The respective lower *and* upper critical messages *of flow* (x_n) *for message* n *from* m *regarding* $\underline{\alpha}_{\rho,T}$ *are*

$$\underline{c}_n^{\rho,m} = \min\{p \in [m,n] \mid \max_{q\in[p,n]}(\frac{q}{\rho} - x_q) = \frac{p}{\rho} - x_p\} \text{ and}$$

$$\overline{c}_n^{\rho,m} = \min\{p \in [m,n] \mid \max_{q\in[p,n]}(x_q - \frac{q}{\rho}) = x_p - \frac{p}{\rho}\}.$$

We will note these points: $\overline{P}_n^{\rho,m} = (x_{\overline{c}_n^{\rho,m}}, \overline{c}_n^{\rho,m})$ *and* $\underline{P}_n^{\rho,m} = (x_{\underline{c}_n^{\rho,m}}, \underline{c}_n^{\rho,m})$*.*

Note that these critical messages only depend on ρ and not on T and σ.

Example 2. *Consider again Figure 1. For* $q \in \{0,1,2,3\}$*,* $x_q - \frac{q}{\rho} = 0 < x_4 - \frac{4}{\rho} = 10$*. Then* x_0 *is the upper critical message for* $q \leq 3$ *from 0 and* x_4 *is the upper critical message for 4 from 0. Graphically, this means that* $P_4 + \overline{\alpha}$ *is below* $\overline{\alpha}$*. We write* $\overline{c}_4^{0.1,0} = 4$*.*

Proposition 1. *Given a data flow $(x_n)_{n\in\mathbb{N}}$ and $\rho \in \mathbb{R}$, the lower and upper critical messages from m can be recursively computed by the following formula:*

$$\underline{c}_n^{\rho,m} = \begin{cases} m & \text{if } n = m \\ n & \text{if } \frac{n}{\rho} - x_n > \frac{\underline{c}_{n-1}^{\rho,m}}{\rho} - x_{\underline{c}_{n-1}^{\rho,m}} \\ \underline{c}_{n-1}^{\rho,m} & \text{otherwise.} \end{cases}$$

The same formula stands for $\overline{c}_n^{\rho,m}$, replacing $>$ by $<$.

Proof. The proof is straightforward from the definition. □

Now, given a flow that is $(\underline{\alpha}, \overline{\alpha})$-constrained on $[m, n]$, checking that it is constrained on $[m, n+1]$, only requires testing that $\underline{\alpha}(x_{n+1} - x_{\underline{c}_n^{\rho,m}}) \leq n + 1 - \underline{c}_n^{\rho,m}$ and $\overline{\alpha}(x_{n+1} - x_{\overline{c}_n^{\rho,m}}) \leq n + 1 - \overline{c}_n^{\rho,m}$. Lemma 2 gives a simple relation between critical and first outgoing messages.

Lemma 2. *Consider $\underline{\alpha}$ and $\overline{\alpha}$ lower and upper constraint curves with rate ρ. Let o be the first outgoing message from m. If the lower constraint is broken, then $\overline{c}_o^{\rho,m} = o$ and if the upper constraint is broken, then $\underline{c}_o^{\rho,m} = o$.*

Proof. The two cases are symmetric, so we only consider the first. Let $\overline{c} = \overline{c}_{o-1}^{\rho,m}$, the upper critical message just before message o. Message $\overline{c}$ is not an outgoing message, so $o - \overline{c} < \rho(x_o - x_{\overline{c}})$ and from Proposition 1, $\overline{c}_o^{\rho,m} = o$. □

Algorithm 1 describes elementary functions that test that the current message (P) satisfies the current constraints (`IsLowerConstrained` and `IsUpperConstrained`) and update the critical messages (`CriticalUpdate`). These functions will be used in Algorithm 2. The notations are the following: $P = (x, n)$ is the current message (on which the constraints are checked), $\underline{P} = (x_{\underline{c}}, \underline{c})$ and $\overline{P} = (x_{\overline{c}}, \overline{c})$ are the respective current lower and upper critical messages.

Algorithm 1: Elementary functions

`CriticalUpdate`(ρ, $\overline{P}$, $\underline{P}$,P)
if $n < \rho(x - x_{\overline{c}}) + \overline{c}$ **then** $\overline{P} \leftarrow P$;
else if $n > \rho(x - x_{\underline{c}}) + \underline{c}$ **then** $\underline{P} \leftarrow P$;
`IsLowerConstrained`(T, ρ, P,$\underline{P}$)
if $n \geq \rho(x - x_{\underline{c}} - T) + \underline{c}$ **then** True **else** False
`IsUpperConstrained`(σ, ρ, P,$\overline{P}$)
if $n \leq \rho(x - x_{\overline{c}}) + \sigma + \overline{c}$ **then** True **else** False

4 Long term behavior computation

In this section, we present our core algorithm, that finds successive curves that constrain a data flow and its multi-layered version. The first algorithm detects the messages that break the constraints (outgoing messages) as they arrive and computes a new rate. There might be frequent outgoing messages, for minor variations of the rate. The multi-layered version of the algorithm discards those minor variations of the rates and rather computes global behavior and only detects strong variations.

4.1 The core algorithm

Let us first focus on the core algorithm (Algorithm 2), that computes arrival rates of the messages of a flow. We use the same notations as in Algorithm 1. The parameters σ and T are fixed (this will be discussed in 4.3) and a flow (inputFlow) is analyzed. Each time a new message is received, the loop (lines 22-25) is executed, except for the initialization: the initial rate ρ is defined as the inverse of the first inter-arrival time, with the convention that message 0 arrives at time 0.

Algorithm 2: Rates computation

```
   Data: T, σ, inputFlow.
   Result: RatesList, outputFlow.
 1 RateUpdate(T, σ, ρ, P̄, P̲, P_p, P)
 2 if not IsLowerConstrained(T, ρ, P, P̲) then
 3 |   ρ ← (n − c̄)/(x − x_c̄);
 4 |   P̲ ← P; P̄ ← P ;
 5 |   outputFlow ← outputFlow :: P;
 6 |   if not IsLowerConstrained(T, ρ, P, P_p) then
 7 |   |   ρ ← (n − n_p)/(x − x_p);
 8 |   write(RatesList, (ρ, x));
 9 else if not IsUpperConstrained(σ, ρ, P, P̄) then
10 |   ρ ← (n − c̲)/(x − x_c̲) ;
11 |   P̲ ← P; P̄ ← P;
12 |   outputFlow ← outputFlow :: P;
13 |   if not IsUpperConstrained((σ, ρ, P, P_p) then
14 |   |   ρ ← (n − n_p)/(x − x_p);
15 |   write(RatesList, (ρ, x));
16 begin
17 |   P ← (receiveDate(inputFlow), 1) ;
18 |   ρ ← 1/x;
19 |   n ← 2 ;
20 |   P_p ← P;
21 |   while true do
22 |   |   P ← (receiveDate(inputFlow), n);
23 |   |   RateUpdate(T, σ, ρ, P̄, P̲, P_p, P);
24 |   |   CriticalUpdate(ρ, P̄, P̲, P);
25 |   |   P_p ← P; n ← n + 1;
26 end
```

In line 22, the coordinates of the new message are set (basically, we count the messages as they arrive). Line 23 calls function `RateUpdate` (lines 1-15) that checks that the current constraints ($\underline{\alpha}_{\rho,T}$ and $\overline{\alpha}_{\rho,\sigma}$) are satisfied (lines 2 and 9). If not, then a new rate is computed using the current message and upper critical message if the lower constraint is broken (lines 3-4) or the lower critical message if the upper constraint is broken (lines 10-11). Lines 6-7 and 13-14 are some marginal improvements: if the two last messages do not satisfy the new constraints, then the rate is updated with those two last messages (the intuition is that the variation in the arrival rate is potentially sharp). Lines 5, 8, 12, 15 will be useful for the multi-layered version and we

will comment on these in the Section 4.2. Finally, line 24 calls `CriticalUpdate` to maintain the critical messages.

Note that if a flow is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained and ρ has been computed by the algorithm, the constraints will always be satisfied for all the next messages and no new rate will be computed. We say that the algorithm has converged in finite time. The remaining of this section is devoted to study cases where this algorithm converges in finite time. The class of periodic flows can be easily studied and we focus our study on that class.

Periodic flows In order to get a more precise idea of the behavior of Algorithm 2, let us first focus on the class of the periodic flows.

Definition 4 (periodic flow)**.** *The flow $(x_n)_{n\in\mathbb{N}}$ is* N-periodic *if* $\forall n \in \mathbb{N}$

$$x_{n+N} - x_{n+1+N} = x_n - x_{n+1}.$$

Proposition 2. *Let $(x_n)_{n\in\mathbb{N}}$ be an N-periodic flow. There exist $\rho, T, \sigma \in \mathbb{R}_+$ such that (x_n) is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained.*

Proof. A straightforward computation with $\sigma = N$, $T = x_N$, and $\rho = x_N/N$ leads to the desired result. □

Proposition 3. *Let $T, \sigma, \rho \in \mathbb{R}_+$. If (x_n) is a N-periodic, $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained flow, then Algorithm 2 with input $(T, \sigma, (x_n))$ either finds a rate ρ in finite time (and the rate will not be updated anymore) or ultimately has a periodical behavior.*

Proof. Either there exists a finite time when there is no update anymore in the rate computed: the algorithm has found a rate ρ such that (x_n) is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained. Or there is no such finite time. In that case, suppose that the mean arrival rate of messages is ρ and at some time, the algorithm finds a slope $r < \rho$, then the lower critical message will be updated at least once every period. Then, when the rate is updated, it is computed as $\frac{n-m}{x_n - x_m}$ for $n - m < N$. The situation is symmetric if $r > \rho$. Then, the number of rates computed is finite and there is some k, m, n such that the pairs of messages to update the rate are (m, n) and $(m + kN, n + kN)$. From there on, the behaviour is periodic. □

One can expect that Algorithm 2 converges after a finite time to the arrival rate of a periodic flow. Unfortunately, it is not the case, and increasing σ and T does not help much, as illustrated in Example 3.

Example 3. *Let us consider a period with 9 messages lasting 180 seconds such that: $x_1 = 15s$, $x_2 = 35s$, $x_3 = 55s$, $x_4 = 80s$, $x_5 = 100s$, $x_6 = 120s$, $x_7 = 140s$, $x_8 = 160s$, $x_9 = 180s$. This flow is constrained with $\rho = 0.05$, $\sigma = 1$ and $T = 10$. Algorithm 2 alternatively finds $\rho_1 = 0.06667$ and $\rho_2 = 0.04$. This computation is represented in Figure 2: the first rate computed is $\rho_1 = 1/x_1$. Then, message 4, arriving at x_{o_1} is the first outgoing message and its upper critical message is message 3 arriving at x_{c_1}. The new rate computed is then $1/(x_4 - x_3) = 0.04$. The next outgoing message is message 10 and its lower critical message is message 9. The rate computed is the same as the initial rate (one period shift).*

Figure 3 shows the behavior of the algorithm when T increases. For example, if $T = 20$ (plain curve), then the first outgoing message is message $o^{20} = 5$ and its upper critical message is $c^{20} = 4$. The new constraints computed are with $\rho = 0.05$ and the algorithm has converged in finite time. But if $T = 60$ (dashed curves), the behavior of the algorithm will again be periodical: the first outgoing message is message $o^{60} = 13$ and its critical message is message $c^{60} = 12$. The rate computed is still $\rho_2 = 0.04$ and the behavior is still periodic.

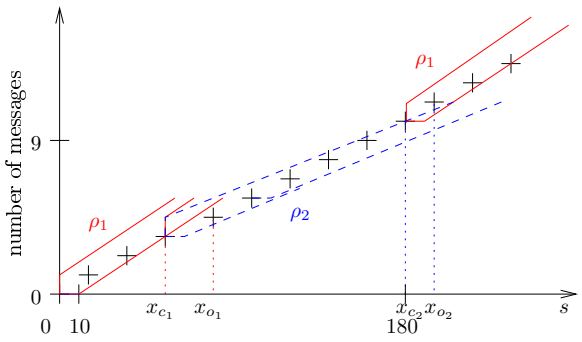


Figure 2: Example of rates computed with Algorithm 2.

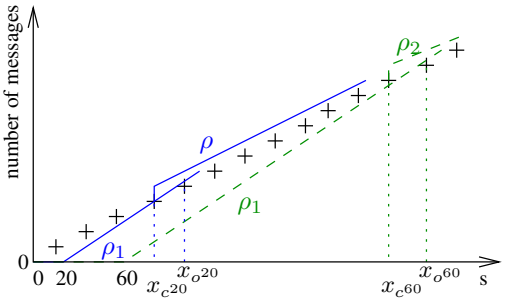


Figure 3: Effect of increasing T on Algorithm 2.

Nevertheless, when the rate computed at some point is close enough from the arrival rates of the messages, Algorithm 2 can converge in finite time.

Theorem 1 (Convergence)**.** *Let (x_n) be a N-periodic, $(\underline{\alpha}_{\rho,T'}, \overline{\alpha}_{\rho,\sigma'})$-constrained flow. There exists ϵ such that if Algorithm 2 with inputs (x_n), $\sigma > \sigma'$ and $T > T'$ can compute $r \in [\rho-\epsilon, \rho+\epsilon]$, it converges to rate ρ in finite time.*

Proof. There are two cases to consider: $r < \rho$ and $r > \rho$. As they are similar, we only consider the case $r > \rho$.

As $T > T'$, $\rho_1 = \rho\frac{x_N - T'}{x_N - T}$ is such that $\forall r \geq \rho_1$, the constraints will not be broken during the first period of (x_n) (for all $t \in [0, x_N]$, $\underline{\alpha}_{\rho_1,T} \leq \underline{\alpha}_{\rho',T'}$).

The set $S = \{\rho x_n - n \mid n \in \mathbb{N}\}$ is finite. Let $n_0 = \min\{n \mid \rho x_n - n = \min_{s\in S} s\}$. We know that $n_0 \leq N$ and n_0 is found upper critical by the algorithm uses the rates r and ρ.

Now, let us have a look at the upper critical messages (when the algorithm uses r, since when it uses ρ, the critical messages will not be updated anymore). As said before, they are updated at least once every period, and the new rate is computed between two messages separated by at most one period. Thus, only a finite number of rates can be found. Take $\rho_2 = \min\{\frac{x_n - x_m}{n-m} \mid 0 \leq m \leq n \leq N \text{ and } \frac{x_n - x_m}{n-m} > \rho\}$. If $\rho \leq r \leq \rho_2$, then, when the upper critical message is updated, the rate between the two critical messages is necessarily ρ (it is necessarily less than ρ, but it cannot be strictly less by construction of S).

But by Lemma 2, we can also conclude that the next rate computed will be exactly ρ, as it is the slope between two upper critical messages. So, it suffices to take $r \leq \min(\rho_1, \rho_2)$ to guaranty

the convergence. □

Convergence for balanced flows

Definition 5 (Balanced flow)**.** *The balanced flow* (x_n) *with rate* ρ *is such that* $\forall n \in \mathbb{N}$,

$$x_n = \lceil \frac{n}{\rho} \rceil.$$

This definition is related to that of balanced words: the n-th letter of the word is defined for each time slot. There is a 0 if no message arrives to skip to the next time slot and k 1s in a row if k messages arrive at a time slot. More details about those words can be found in [9]. We choose a rational slope so that the flow is periodic. By construction, if $\rho > 1$, there can be several messages with the same arrival time.

The structure of a balanced word with rational slope ρ is the following: $\exists a_1, \ldots, a_k > 0$ such that for each $\ell \in \{1, \ldots k-1\}$, the balanced word can be decomposed into the following pair of factors: $w_\ell = w_{\ell-1}^{a_\ell} w'_{\ell-1}$ and $w'_\ell = w_{\ell-1}^{a_\ell - 1} w'_{\ell-1}$, with $w_0 = 0$ and $w'_0 = 1$. The periodic pattern of the word is $w_k = w_{k-1}^{a_k - 1} w'_{k-1}$.

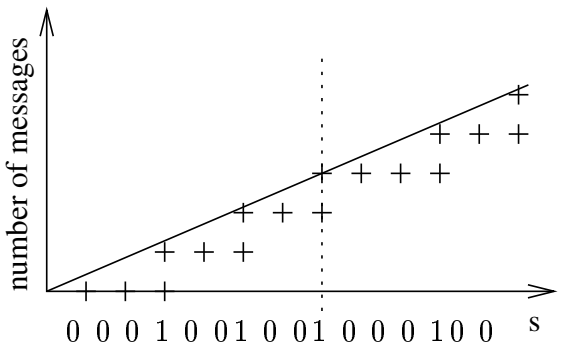


Figure 4: Example of a balanced flow with a slope $\rho = \frac{3}{7}$.

Example 4. *Consider the balanced flow represented on Figure 4. Initials factor are* $w_0 = 0$ *and* $w'_0 = 1$*. Then the others factors are* $w_1 = 0001$*,* $w'_1 = 001$ *and so* $w_2 = 0001001$ *(*$a_2 = 1$*),* $w'_2 = 001$*. Finally, the periodic pattern is* $w_3 = w'_3 = w_2 w'_2 = 0001001001$ *(*$a_3 = 1$*).*

The slope of a finite word is the number of 1 divided by its number of 0: $\rho(w) = \frac{|w|_1}{|w|_0}$, where $|w|_a$ is the number of occurrences of a in w. The slope of a periodic word is the slope of its periodic patterns: $\rho(w^\omega) = \rho(w)$. A simple induction shows that $\rho_\ell = \rho(w_\ell) \leq \rho(w'_\ell) = \rho'_\ell$.

Example 5. *Consider again the balanced flow of slope* $\rho = \frac{3}{7}$ *represented on Figure 4. The slope of the factors of this periodic pattern are* $\rho_1 = \frac{1}{3}$*,* $\rho'_1 = \frac{1}{2}$*, then* $\rho_2 = \frac{2}{5}$*,* $\rho'_2 = \frac{1}{2}$*, and finally* $\rho_3 = \rho'_3 = \frac{3}{7}$*.*

Lemma 3. $\forall \ell$*, there exist* w *and* v *such that* $w'_\ell = w1$ *and* $w_\ell = w0v$*.*

Proof. This is true for $\ell = 0$. Suppose it is true for $\ell - 1$. $w'_\ell = w_{\ell-1}^{a_\ell - 1} w'_{\ell-1}$ and $w_\ell = w_{\ell-1}^{a_\ell - 1} w_{\ell-1} w'_{\ell-1}$, so this is true for ℓ. □

Lemma 4. *Let x be a proper prefix of w_ℓ (resp. w'_ℓ). Then $\rho(x) \leq \rho_\ell$.*

Proof. By induction. This is true for $\ell = 1$, $w_\ell = w_{\ell-1}^{a_\ell} w_{\ell-2}^{a_{\ell-1}-1} w_{\ell-3}^{a_{\ell-2}-1} \cdots w_1^{a_2-1} w'_1$ and $w'_\ell = w_{\ell-1}^{a_\ell-1} w_{\ell-2}^{a_{\ell-1}-1} w_{\ell-3}^{a_{\ell-2}-1} \cdots w_1^{a_2-1} w'_1$. □

Theorem 2. *Fix $\rho > 0$, $\sigma > \lceil \rho \rceil$ and $T \geq \lceil 1/\rho \rceil$. If (x_n) is a periodic balanced flow of slope ρ, then Algorithm 2 with parameters σ and T converges to ρ in finite time.*

Proof. We show by induction on ℓ that the patterns w_ℓ or/and w'_ℓ correspond to the slopes discovered by the algorithm.

Initialisation: the first slope to be discovered is either $\rho'_1 = 1/a_0$ or $\rho_1 = 1/(a_0+1)$.

Induction hypothesis: the slope ρ_ℓ or ρ'_ℓ is discovered in finite time.

Induction step: We will use Lemma 2. If $\ell \neq k$, $\rho_\ell < \rho < \rho'_\ell$. Suppose that the slope that has been found is ρ_ℓ. Then the upper constraint will be broken, at a lower critical message.

The word can be decomposed in factors w_ℓ and w'_ℓ. As long as only factors of w_ℓ are found, the upper constraint is not broken and the lower critical message corresponds to the message before the last of the first occurrence of w_ℓ (note that this first occurrence can be a suffix of w_ℓ), and then this is valid for w'_ℓ since $w_\ell = w_{\ell-1} w'_\ell$.

A careful look at the structure of the words w_ℓ and w'_ℓ shows that only the last message of w'_ℓ can be critical. Moreover, the constraint cannot be broken at the first occurrence of w'_ℓ as $\sigma > \lceil \rho \rceil$. Then, necessarily, the next slope computed will correspond to the slope computed between two occurrences of w'_ℓ, and then be either $\rho_{\ell+1}$ or $\rho'_{\ell+1}$.

Now suppose that we start from the slope ρ'_ℓ. One could apply exactly the same argument, but it is slightly more difficult with the decomposition of the words that have been chosen. Another simpler argument is to see that the problem of the upper critical messages is the same as the lower critical messages for the algorithm applied to the balanced work of slope $1/\rho$ (the role of 0 and 1 is exchanged). □

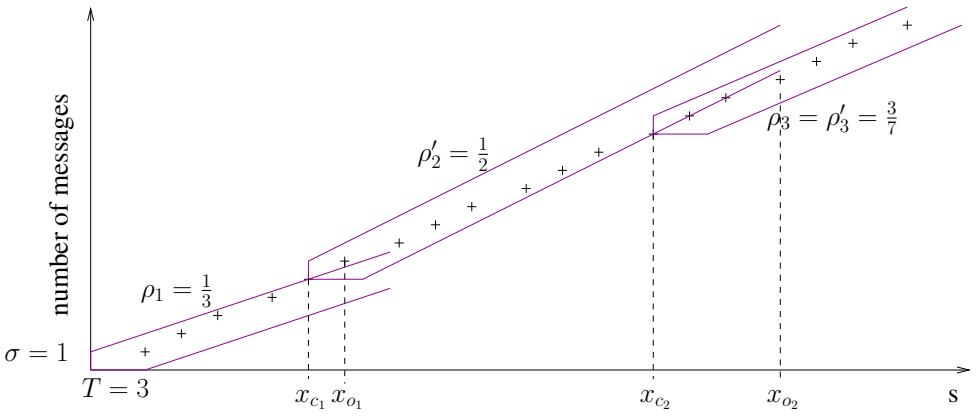


Figure 5: Rates computed with Algorithm 2 for the balanced flow of slope $\rho = \frac{3}{7}$.

Example 6. *Figure 5 represents the actual rates computed by Algorithm 2 when $T = 3$ and $\sigma = 1$, for the balanced flow of slope $\rho = \frac{3}{7}$ represented in Figure 4. The first rate computed is $\rho_1 = 1/x_1 = \frac{1}{3}$. Then, message 6, arriving at x_{o_1} is the first outgoing message and its lower critical message is message 5 arriving at x_{c_1}. The new rate computed is then $1/(x_6 - x_5) = \frac{1}{2}$.*

The next outgoing message is message 16 arriving at x_{o_2} and its upper critical message is message 13 arriving at x_{c_2}. The rate computed is $1/(x_{16} - x_{13}) = \frac{3}{7}$. So, convergence occurs after 38 s, when 16 messages have arrived. Moreover, one can observe that the computed rates belong to the set of slopes of the factors computed in Example 5 as stated in Theorem 2.

4.2 The multi-layered algorithm

In this section we present an adaptation of our algorithm that provides a better analysis of the flows. Algorithm 2 computes short term arrival rates of a flow: as said before, minor variations of the arrival rates of the messages may be detected, whereas longer term arrival rate could be preferred.

To be able to offer such levels of details, we modify our algorithm into a multi-layered architecture. Then the ground layer (or layer 0) provides interesting quantity of details; its objective is to detect small variations in the arrival rate of messages. Finally, the last layer of the algorithm returns long term arrival rates of messages. So if the flow is quite regular, the algorithm will find the mean arrival rate. By contrast, if the arrivals are erratic, the important variations of the flow will still be detected by the successive layers, with fewer details, thus pointing out critical variations (see Section 5 for examples). We also define by experimentation that 3 layers are enough to achieve such analysis.

To do this we now have to explain lines 5 and 12 of Algorithm 2. An output flow of messages is built using the *First outgoing messages* defined in Definition 2. This is a sub-flow of the initial flow, keeping the original numbering of the messages. Then, it is possible to run Algorithm 2 on this output flow (outputFlow). Instead of reading messages from an input flow, the algorithm uses the messages of outputFlow (lines 17 and 22 are modified accordingly).

Figure 6 gives an example of the structure of the multi-layered algorithm with three layers: first, an input flow is given to Algorithm 2 to compute arrival rates as explained in Section 4.1) - this is our layer 0 (or ground layer). Then, Algorithm 2 is run using outputFlow of layer 0 - this builds layer 1 and produces a new output flow, and so on.

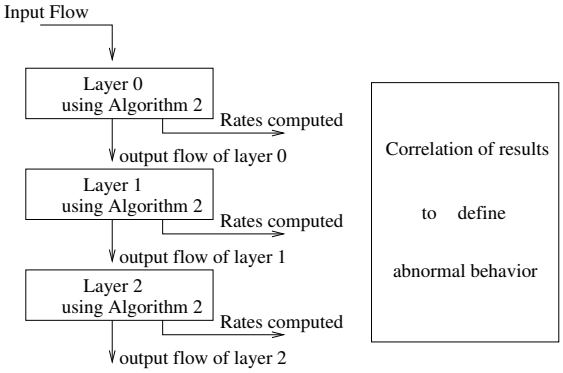


Figure 6: Example of Algorithm 2 used with several layers.

Example 7. *Previously, in Example 3, we showed that Algorithm 2 may not always converge to the arrival rate for periodic flows. Figure 7 represents the rates computed with the multi-layer algorithm with 2 layers for the input flow and parameters of Example 3. The first (resp. second) time line represents the rates computed on layer* 0 *(resp. layer* 1*). The dates correspond to changes of rate in the corresponding layer. For example, for layer 0, the current rate is*

$\rho_1 = 0.066$ between time $t = 0$ and $t = 55$ seconds. Note that the arrow of layer 1 starts at $t = 80$ seconds: it corresponds to the first time a rate can be computed on this overlay because two messages broke the current constraints in layer 0. Then, on this example, the multi-layer algorithm converges to the arrival rate. Up to our knowledge, this is still an open question whether there exist a layer for which this algorithm will converge for a periodic flow of messages.

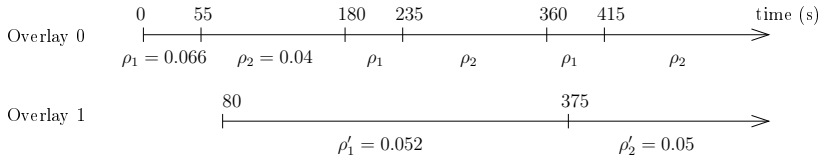


Figure 7: Example of algorithm computation with 2 overlays.

4.3 Discussion for the algorithm

Several points of the algorithm need some discussion: the choice of the parameters σ and T, the choices that were made for the implementation, and solutions that could be implemented to guaranty the convergence of our algorithm.

The role of σ and T In the previous paragraph, we have seen that σ and T do not play an important role: they have no obvious property that make the convergence easier if they are increased (see Example 3). Concerning this matter, the multi-layer version of the algorithm is far more efficient.

However, those parameters have to be chosen by the user (a network administrator for example) in order to define the tolerance to detect the rate variations, particularly in the ground layer. Small parameters will allow a very refined detection. Finally, the choice of these parameters will be made in accordance with the theoretical characteristics of the flow of interest. For example we will see in Section 5 that it can be useful to take small values for the first layers and larger values for the last layers to respectively emphasize the precision and the long term computation. More precisely, we have taken $\sigma = 1$ for layers 0 and 1 and $\sigma = 5$ for layers 2 and 3 in the next section.

Implementation choices In our algorithm, we chose to update the rate when the lower constraint is broken with the upper critical message, and conversely. Another solution would have been to choose the other way round: update the rate with the lower critical message when the lower constraint is broken. Doing this, there is no way to ensure the convergence in finite time, in view of Lemma 2. However, with this implementation, we could experimentally notice some convergence when time goes to infinity.

Convergence of the algorithm Several solutions can be proposed for the convergence of our algorithm for regulated flows (flow of messages respecting some $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constraints).

- *Linear Regression of least squares approach:* This method is a well-known technique to approximate a set of points with a linear function. In our study, this method can be efficient to compute the mean rate on the last overlay. Furthermore, it is an interesting technique to raise the problem of convergence for periodic flows. We did not find any example showing that the multi-layered algorithm does not converge, but for a given layer, it is quite sure that this case may happen. One solution to get the mean arrival rate

would be to compute a linear regression on the last layer to obtain a rate close to ρ. Thus Theorem 1 proved that the last overlay converges to ρ after a finite number of computation.

- *Correlation between the layers:* The idea of this approach is to correlate results from the lower and the upper layers of algorithm: even in the case of non-convergence of the last layer, the rate computed will converge to ρ, as only sub-flows are considered. Then, injecting the rate computed by the last layer to the first layer will eventually ensure the convergence in view of Theorem 1.

- *Mix the implementations:* Even in the single-layer algorithm the convergence can be improved: it suffices to randomly run Algorithm 2 and its adaptation in the previous paragraph.

5 Experimental results

In this part, we perform Algorithm 2 on an OSPF flow monitoring on a telecommunication network which topology is based on the German main cities. Each node is set from an Ubuntu Linux that hosts a running instance of the well-known Quagga Routing Software Suite [13]. We obtain a testbed that will act as a real network and that can be monitored with traditional network management tools.

We then define several scenarii to perform on the network: First a normal and stable network with a fluid OSPF traffic that acts as a reference; then a cyclic and malicious OSPF protocol stack router failure; and to conclude, a multiple competitive router start that provides many OSPF convergence perturbations. For each scenario we monitor all exchanged messages in the testbed as in an operational context. Afterwards the records are given to Algorithm 2 with multiple layers. Finally the resulting rates obtained allow us to prove its operational interest. Some of the figures also represent the *moving average method* (MAM) that will be compared to our results in the last chapter of this section.

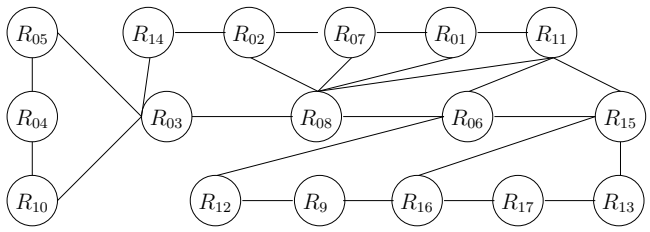


Figure 8: Topology of the network studied.

5.1 The Open Shortest Path First protocol

The Open Shortest Path First (OSPF) protocol ([11]) is a link state protocol that makes intern IP routing. To do so, routers exchange network's information such as topology, metrics, alive routers, etc. These exchanges of data are made using different kind of messages: *Hello* and *Link State Advertisement* (LSA) messages. *Hello* messages are sent every 10 seconds. The *LSA* ones describe the evolution of the routes of the network. Typically, they are sent every 30 minutes, at this moment routers send the information they have recorded about the network topology. *LSA* sending stops when all routers have recorded the same information.

The OSPF protocol is an emblematic protocol of the networking domain. Furthermore, it has the great advantage to be well-known and accessible, so that the relation between algorithm results and the network behavior is clear and immediate.

5.2 OSPF fluid traffic study

In this part, we present an OSPF flow of messages when there is no perturbation in the network. The flow evaluated is the one from router R_{11} to router R_8. Figure 9 represents the arrival of messages on this link between 0 and 8000 s with a zoom on that flow between 1500 and 2500 s. One can observe that the flow seems to be globally linear. This represents the sending of *Hello* messages. But, when having a deeper look (zoom on the figure), one can observe that the linear behavior is perturbed between 1900 and 2250 s: the amount of arrivals intensifies. On the global flow, it is clear that this scheme periodically happens (every 1800 s), corresponding to an *LSA refresh* operation. Consequently, the objective here is to highlight these two behaviors with our algorithm.

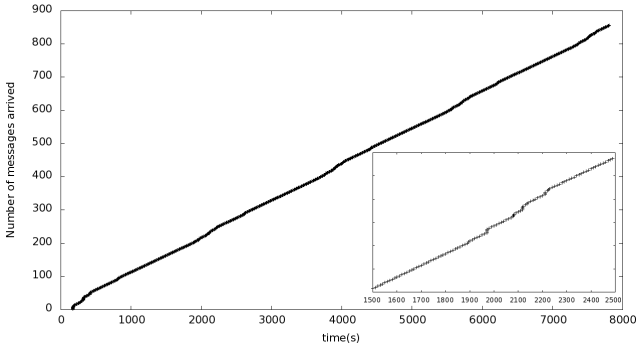


Figure 9: Flow of messages from R_{11} to R_8 during 8000 s with a detailed period (between 1500 and 2500 s) at the bottom right.

The results of the ground layer of our algorithm on the flow studied are shown by the plain curve in Figure 10. One can clearly observe a periodical scheme in this curve. Indeed, there are two kinds of behaviors on the rates computed: the stable case during which at most three rates are computed in 1500 s (these rates have low values that do not exceed a slope of 1 message per second) and the perturbed case that last at most 400 s. During this period, there are in average four rates computed. Furthermore, the slopes have a mean value higher and vary between 0.5 to 1000 messages per second.

Finally, the plain curve of Figure 11 shows the results of the third layer of our algorithm. One can notice that the two behaviors are still present. It is important to observe that each behavior is represented by only one slope that represents the average messages arrival on the period. When only *Hello* messages are sent the slope is a little higher than 0.11. Then, when *LSA* and *Hello* messages are sent the slope equals in mean to 0.14. Between each occurrence of the *Hello* rate and the *LSA refresh* one, there is another slope computed for a few second and that equals to 0.1 message per second. It corresponds to a transition time and is negligible.

Note that on each figure, the beginning of the flow is slightly different to the rest of the curve. This is normal and corresponds to the starting of the network.

In a nutshell, Algorithm 2 has effectively reached the theoretical rates of the two trends on arrival for the OSPF protocol on the flow from R_{11} to R_8.

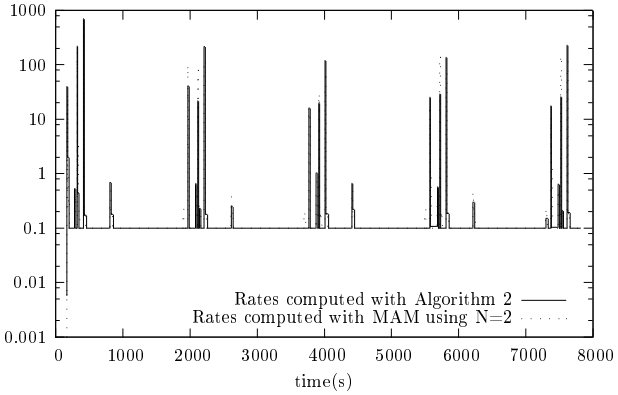


Figure 10: Results of our algorithm on layer 0 and MAM with $N = 2$ on the link from R_{11} to R_8 in fluid traffic case.

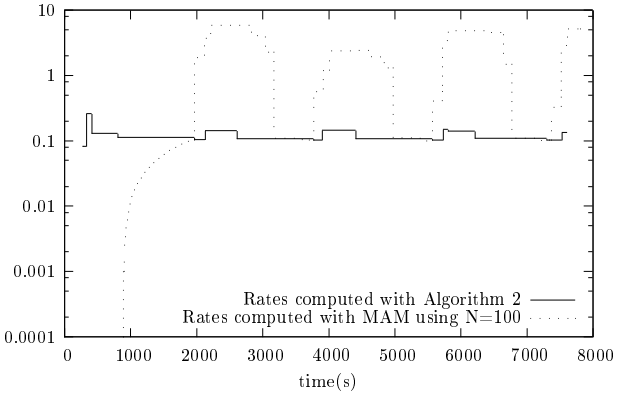


Figure 11: Results of our algorithm on layer 3 and MAM with $N = 100$ on the link from R_{11} to R_8 in fluid traffic case.

5.3 When router R_8 fails

In this part, we present a flow of OSPF messages when in router R_8 a failure occurs every 6 min and lasts 3 min. This corresponds to a bug in the implementation that forces R_8 to reboot frequently. The flow evaluated is from router R_8 to router R_{11}. Figure 12 represents the arrival of messages on this link between time 0 and 4000 s. One can observe that at the beginning messages arrive regularly, which corresponds to *Hello* messages arrival. Then, at time 310 s, arrival of messages stops and restarts 180 s later. This is the time during which router R_8 cannot send messages because it is restarting. Afterwards, at time 500 s, messages are received again: more frequently first, as initially then. Here, R_8 first floods *LSAs* when restarting and sends *Hello* messages again, until it crashes.

The result of the ground layer of our algorithm on the flow of interest is shown in Figure 13. Most of the time, the slope computed is $\rho = 0.1$, which corresponds to the receipt of *Hello* messages. But, each time R_8 restarts, there is a sudden high rate computed - between 10 and 400 - with a mean slope value that equals 30.

Figure 14 represents the rates computed on the third layer of our algorithm. The resulting rates are smoother than the ones of layer 0. Here, rates do not exceed 1 when it could be 400 on

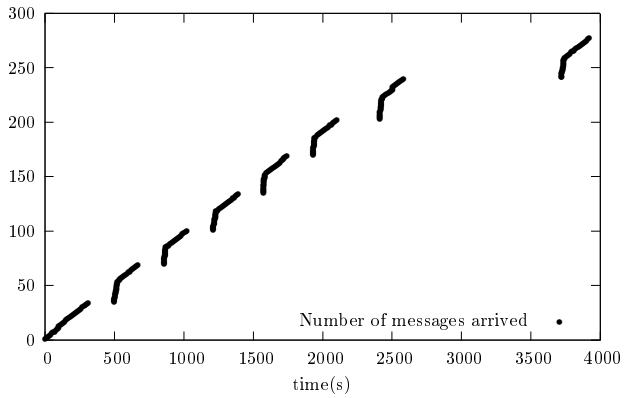


Figure 12: Flow of messages from R_8 to R_{11} during 4000 seconds.

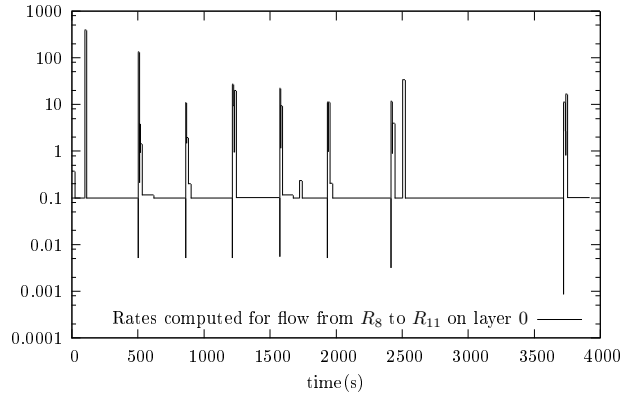


Figure 13: Rates computed on layer 0 with our algorithm on the flow of messages from R_8 to R_{11}.

layer 0. Furthermore, the number of abrupt rate change is reduced on layer 3. Unfortunately, it does not establish additional knowledge on this flow. In this case, a smoother study does not bring additional value and layer 0 exactly represents what happens: each reboot of router R_8 is pointed out by an abrupt rate change.

5.4 OSPF convergence perturbations

In this experiment, each node is started the one after the other and the full run of the current node must be established to start the next one. The start order is: $R_1, R_{10}, R_{11}, \ldots, R_{17}, R_2, \ldots, R_9$. This experiment simulates the conditions for observing a classical problem when several routers have to face multiple changes at the same time. Figure 15 presents an OSPF flow of messages exchanges from router R_{11} to router R_8 between time 0 and 1600 s in this context. From 30 to 100 s *Hello* messages are sent regularly by R_{11} to R_8. But router R_8 has not started yet, and thus does not answer to R_{11}. This implies that between 100 and 140 s R_{11} reduces its *Hello* sending speed. Afterwards, as R_8 has still not started, R_{11} stops flooding messages until R_8 starts (at time 190 s). Now the connection is made, a few *Hello* messages are sent (between time 190 and 230 s) before routers could exchange their database in a LSA operation process. This operation is clearly observable with the burst of messages between time 230 and 271 s. Afterwards, *Hello*

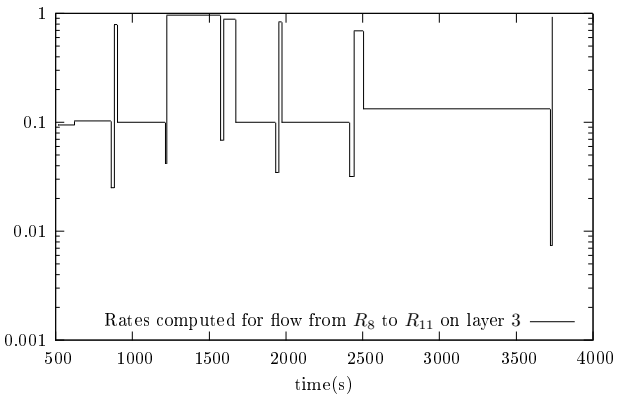


Figure 14: Rates computed on layer 3 with our algorithm on the flow of messages from R_8 to R_{11}.

messages are sent. During this period, there are lost messages (at time 285, 335, and 445 s), because R_8 is still struggling in LSA exchange operations. Finally, one can observe a second perturbation between time 1200 and 1300 s during which R_8 does not answer to *Hello* messages from R_{11} because it is busy on synchronizing with R_9. Note that this perturbation arises quite long after R_8 starts running because in the topology R_9 is far from R_8.

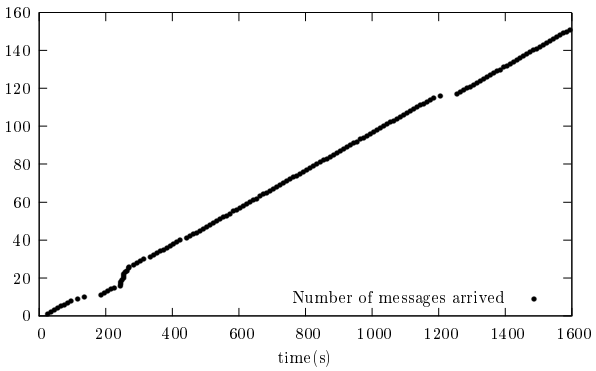


Figure 15: Flow of messages from R_{11} to R_8 during 1600 seconds.

Figure 16 shows the ground layer of our algorithm on the studied flow. One can immediately observe the two perturbations discussed above: the beginning of messages broadcast from R_{11} to R_8 between time 0 and 450 s and the busy period of router R_8 due to its synchronization with R_9 between time 1200 and 1300 s.

Finally, Figure 17 presents layer 1 of our algorithm on the studied flow. One can observe an improvement compared to the ground layer as the two perturbations are still present but with less details, which is what we are looking for. This conclude that our algorithm clearly points out the abnormal behaviors of the flows of messages.

5.5 Comparison with the Moving Average Method

In this part, we compare our algorithm to another method based on moving averages. MAM analyzes a data flow to suppress instantaneous fluctuations and then to study long term behaviors.

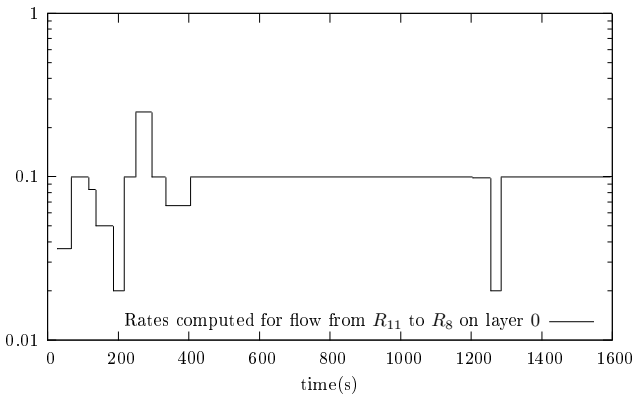


Figure 16: Rates computed on layer 0 with our algorithm on the flow of messages from R_{11} to R_8.

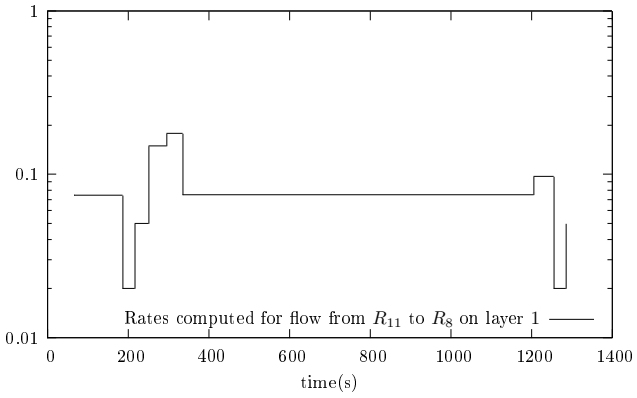


Figure 17: Rates computed on layer 1 with our algorithm on the flow of messages from R_{11} to R_8.

In a data flow, the mean value associated to a given point is computed using that point and the $N-1$ former ones, where N is a fixed parameter. To stick with our algorithm, the data flow we use is the arrival dates messages. For message n, it is defined by $\frac{1}{x_n - x_{n-1}}$.

Figure 10 presents results of the ground layer of our algorithm (plain curve) and MAM with $N = 2$ (dotted curve). The results are similar. Then, Figure 11 compares results of the third layer of our algorithm (plain curve) and the ones of MAM with $N = 100$ (dotted curve). The results given by the two methods are really different. The first noticeable difference is that our algorithm outputs fewer points, stressing only the changes of arrival rate (68 output points versus 850 points). In the case of LSA refresh operations our algorithm highlight it by the slight and short slope changes, quite as MAM if N is put to N = 100. However, for MAM, these changes last too long (around 1000 s) and also there is no convergence to the mean arrival rate. This means that our algorithm is more accurate to detect these changes. Finally, the quality of the results of the MAM is highly sensitive on the value taken for N, whereas our algorithm only considers the constraints of the flows. Other methods using sliding windows, like the *exponential moving average* did not lead to any conclusive results.

6 Conclusion

This article presents a method to study flows of messages arrivals in networks. This work aims at detecting hidden abnormal behaviors.

The algorithm introduced here is very light in computing complexity and in memory usage. Thus, it works on the fly to detect bad behavior immediately. Furthermore, no expert is needed to interpret the results and the method proposed has the great advantage to return flow behavior and thus it cannot statue on false problem. When compared to the Moving Average Method, we have shown that our algorithm supply a really better understanding of the studied flows.

This work will be deepened with other network management contexts studies, such as network security with the detection of DoS/DDoS attacks and network alarms management by detecting abnormal trends in the huge flow generated by a telecommunication network.

References

[1] Z. Ben Houidi, M. Meulle, and R. Teixeira. Understanding slow BGP routing table transfers. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 350–355, 2009.

[2] C. S. Chang. *Performance Guarantees in Communication Networks.* TNCS, Springer-Verlag, 2000.

[3] C. S. Chao, D. L. Yang, and A. C. Liu. An automated fault diagnosis system using hierarchical reasoning and alarm correlation. *Journal of Network and Systems Management*, 9:183–202, 2001.

[4] D. Ghita, Hung Nguyen, M. Kurant, K. Argyraki, and P. Thiran. Netscope: Practical network loss tomography. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[5] H. Han, C.V. Hollot, D. Towsley, and Y. Chait. Synchronization of tcp flows in networks with small droptail buffers. In *44th IEEE CDC-ECC*, pages 6762 – 6767, 2005.

[6] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. *SIGCOMM Comput. Commun. Rev.*, 34:219–230, 2004.

[7] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume LNCS 2050. Springer-Verlag, 2001. revised version 4, May 10, 2004.

[8] X. Li, F. Bian, H. Zhang, C. Diot, R. Govindan, W. Hong, and G. Iannaccone. MIND: A distributed multi-dimensional indexing system for network diagnosis. In *In IEEE INFOCOM*, 2006.

[9] M. Lothaire. *Algebraic combinatorics on words.* Cambridge university press, 2002.

[10] S. McConnell. *Code Complete, Second Edition.* Microsoft Press, 2004.

[11] J. Moy. RFC 2328 OSPF v2. Technical report, 1998.

[12] The UNIVERSELF Project. Univerself, realizing autonomics for future networks, http://www.univerself-project.eu/.

[13] Quagga. A routing software package for tcp/ip networks, http://www.nongnu.org/quagga/.

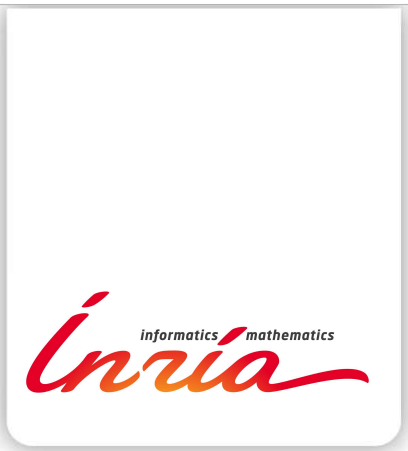
[14] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of pca for traffic anomaly detection. *SIGMETRICS Perform. Eval. Rev.*, 35:109–120, June 2007.

[15] P. Singh. Alcatel-lucent helps telcos to track, fix network problems, http://www.duwire.com/news/alcatel-lucent-helps-telcos-to-track-fix-network-problems-2011-08-23-160017/, 2011.

[16] B. Swaminathan. Agile methodologies overview. http://www.slideshare.net/Siddhi/intro-to-agile, 2007.

[17] A. Willsky and H. Jones. A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems. *IEEE Transactions on Automatic Control*, 21(1):108 – 112, 1976.

[18] A. S. Willsky and H. L. Jones. A generalized likelihood ratio approach to state estimation in linear systems subjects to abrupt changes. In *13th IEEE Conference on Decision and Control*, pages 846 –853, 1974.



RESEARCH CENTRE
RENNES – BRETAGNE ATLANTIQUE

Campus universitaire de Beaulieu
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399