Hidden Anomaly Detection in Telecommunication Networks

Anne Bouillard
ENS / INRIA
45 rue d'Ulm
75230 Paris cedex 05, France
anne.bouillard@ens.fr

Aurore Junier
INRIA / IRISA
Campus de Beaulieu
35000 Rennes, France
aurore.junier@inria.fr

Benoit Ronot
Alcatel-Lucent Bell Labs
Villarceaux,
91620 Nozay, France
benoit.ronot@alcatel-lucent.com

***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 article, 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.**

I. Introduction

The all IP convergence in telecommunication networks and the fact that operators always add new functionalities and new services to their network bring a vertical and horizontal multilayer matrix of data exchanges 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 at 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 needing a node restart 3 or 4 times per year to fix the resulting issue, will not be investigated by the management team.

Our assumption is 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.

978-3-901882-48-7

In this article, we present an algorithm that will meet 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 message exchanges in modern networks is constrained by the design of its 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 a detection.

Our algorithm uses the Network Calculus theory [3], [8] 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 of 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 II, we present existing methods for anomaly detection. Section III defines the technical framework, based on constraint curves, of our algorithm that is described in Section IV. Finally, we give some experimental results based on an operational testbed in Section V, before concluding in Section VI.

II. 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 [9], 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), [7], [14] is a mathematical process that converts a set of correlated observations into 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 [7], 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 [4], 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 into 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 [6], 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 ([5]). 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 whose loss rates cannot be accurately computed.

Today, lots 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 developed methods 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.

III. 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. Due to the lack of space, proofs have been omitted. They can be found in [2].

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, where the data flow is $(0, 10, 20, 30, 50, 70, \ldots)$ and the corresponding graph is $((0,0),(10,1),(20,2),(30,3),(50,4),(70,5),\ldots)$.

Arrival curve is a fundamental notion in *Network calculus* ([8], [3]), 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 $[0, +\infty]$.

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

Example 1: On Figure 1, $\overline{\alpha} : t \mapsto 1 + 0.1t$ and $\underline{\alpha} : t \mapsto \max(0.1(t-10), 0)$ are depicted, as well as $P_4 + \overline{\alpha}$. It should be clear that the flow is upper-constrained by $\overline{\alpha}$, but not lower-constrained by $\underline{\alpha}$. Indeed, P_5 is below $\underline{\alpha}$, and we have $5 - 0 = 5 \leq \underline{\alpha}(x_5 - x_0) = 6$. To conclude, the flow is $(\underline{\alpha}, \overline{\alpha})$-constrained on $[0, 4]$ but not on $[0, 5]$.

In this article, we aim at finding constraints for the arrival flows on long intervals, on the fly. Given a data flow (x_n) and arbitrary curves $\underline{\alpha}$ and $\overline{\alpha}$ such that (x_n) is $(\underline{\alpha}, \overline{\alpha})$-constrained

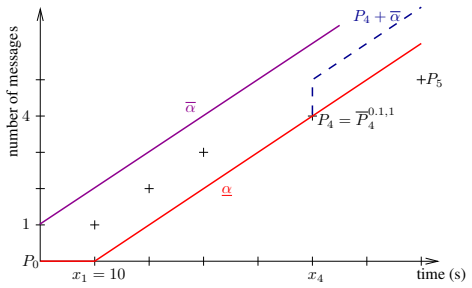

Fig. 1. Data flow and constraints.

on $[0, n]$, checking that the $n+1$-th messages satisfies those constraints requires $O(n)$ operations and $O(n)$ space: for each previous message, one has to check the inequalities with $n' = n+1$ given in Definition 1. However, if $\underline{\alpha}$ and $\overline{\alpha}$ are well-chosen, this complexity can be drastically reduced, to a constant number of operations and constant space. As a consequence, we will focus on two families of simple functions, that fulfill these requirements: the affine functions for the upper constraints and the non-negative parts of affine functions 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 parameter ρ corresponds to the long term arrival rate of messages, and σ represents the maximal burst, which is the maximum number of messages can arrive simultaneously. Finally, T represents the maximal delay between two messages.

Throughout the article, we will assume that σ and T are fixed, and our goal is to guess ρ such that if the flow $(x_n)_{n\in\mathbb{N}}$ is very regular, it is $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-constrained. If the traffic is not regular, our goal is to find the successive arrival rates of the messages. In the former case, Lemma 1 ensures the existence and uniqueness of ρ. In the latter case, if we are given the n first messages, one need to check, for the next one, if it satisfies the current constraints and, should the case arise, to compute a new rate. Due to the shape of the functions we chose, this can be done in constant time. To do this, we introduce the notions of *first outgoing message* and *critical messages*.

Lemma 1 (Uniqueness): Let $(x_n)_{n\in\mathbb{N}}$ be a data flow. 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.

When constraints are defined from the n-th message, one need to check whether the following messages still satisfy the constraint or what is the first message that breaks the constraints. In order to compute this, we introduce some new definitions. The *first outgoing message* is the first message to break the current constraints, and the *critical messages* are messages that give the strongest constraints. Due to the special shape of the functions we consider (affine functions), only one critical message for each curve must be taken into account: graphically, if $P_i + \overline{\alpha}$ is above $P_j + \overline{\alpha}$ at some point, then it will remain above. Figure 1 gives an illustration of that: $P_0 + \overline{\alpha}$ is above $P_4 + \overline{\alpha}$, so P_4 gives a strongest constraint than P_0, hence will be called (upper)-critical. More formally, we have the following definitions.

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

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

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. Message 5 is the first outgoing message as it breaks the lower-constraint. 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 $<$.

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 $\underline{c}_o^{\rho,m} = o$.

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

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

IV. 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.

A. 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 IV-C) 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.

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 last two messages do not satisfy the new constraints, then the rate is updated with those last two 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 IV-B. 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.

Algorithm 2: Rates computation

Data: T, σ, inputFlow.
Result: RatesList, outputFlow.
1 `RateUpdate`(T, σ, ρ, $\overline{P}$, $\underline{P}$, P_p, P)
2 **if** not `IsLowerConstrained`(T, ρ, P, $\underline{P}$) **then**
3 $\quad\rho \leftarrow (n - \overline{c})/(x - x_{\overline{c}})$;
4 $\quad\underline{P} \leftarrow P$; $\overline{P} \leftarrow P$;
5 $\quad$outputFlow $\leftarrow$ outputFlow :: P;
6 $\quad$**if** not `IsLowerConstrained`(T, ρ, P, P_p) **then**
7 $\quad\quad\rho \leftarrow (n - n_p)/(x - x_p)$;
8 $\quad$write(RatesList, (ρ, x));
9 **else if** not `IsUpperConstrained`(σ, ρ, P, $\overline{P}$) **then**
10 $\quad\rho \leftarrow (n - \underline{c})/(x - x_{\underline{c}})$;
11 $\quad\underline{P} \leftarrow P$; $\overline{P} \leftarrow P$;
12 $\quad$outputFlow $\leftarrow$ outputFlow :: P;
13 $\quad$**if** not `IsUpperConstrained`($(\sigma$, ρ, P, P_p) **then**
14 $\quad\quad\rho \leftarrow (n - n_p)/(x - x_p)$;
15 $\quad$write(RatesList, (ρ, x));
16 **begin**
17 $\quad P \leftarrow$ (receiveDate(inputFlow), 1) ;
18 $\quad\rho \leftarrow 1/x$;
19 $\quad n \leftarrow 2$;
20 $\quad P_p \leftarrow P$;
21 $\quad$**while** *true* **do**
22 $\quad\quad P \leftarrow$ (receiveDate(inputFlow), n);
23 $\quad\quad$`RateUpdate`(T, σ, ρ, $\overline{P}$, $\underline{P}$, P_p, P);
24 $\quad\quad$`CriticalUpdate`(ρ, $\overline{P}$, $\underline{P}$, P);
25 $\quad\quad P_p \leftarrow P$; $n \leftarrow n + 1$;
26 **end**

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.

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.

One could 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 $\rho_1 = 0.067$ and $\rho_2 = 0.04$. This computation is represented in Figure 2: the first computed rate 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 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 next computed rate 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 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 computed rate is still $\rho_2 = 0.04$ and the behavior is still periodic.

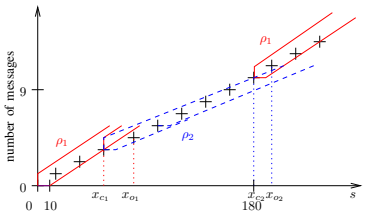

Fig. 2. Example of rates computed with Algorithm 2.

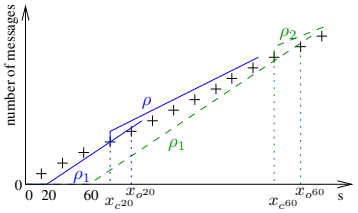

Fig. 3. Effect of increasing T on Algorithm 2.

Nevertheless, when the rate computed at some point is close enough to 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.

For more regular flows, like balanced flows (*i.e.* such that $\forall n \in \mathbb{N}$, $x_n = \lceil \frac{n}{\rho} \rceil$), one can show (see [2]) that Algorithm 2 converges.

B. 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 V for examples).

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 4 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 IV-A) - 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.

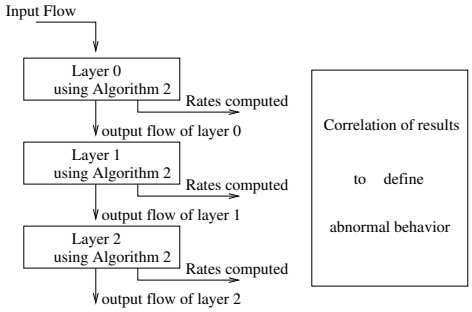

Fig. 4. Example of Algorithm 2 used with several layers.

Example 4: Previously, in Example 3, we showed that Algorithm 2 may not always converge to the arrival rate for periodic flows. Figure 5 represents the rates computed with the multi-layered algorithm with 2 layers for the input flow and the parameters of Example 3. The first (resp. second) time line represents the rates computed on layer 0 (resp. layer 1). The dates correspond to rate updates 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 layer, that is when two messages have broken the current constraints in layer 0. Then, on this example, the multi-layered algorithm converges to the arrival rate. Up to our knowledge, this is still an open question whether there exists a layer for which this algorithm will converge for a periodic flow of messages.

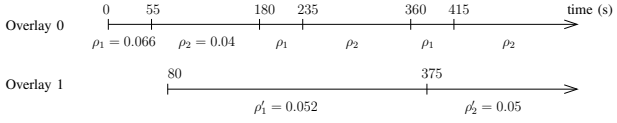

Fig. 5. Example of algorithm computation with 2 overlays.

C. Discussion about 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 the solutions that could be implemented to guaranty the convergence of our algorithm.

a) **The role of σ and T:** In the previous paragraph, we saw 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-layered version of the algorithm is far more efficient.

However, those parameters have to be carefully chosen by the user 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 V 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.

b) **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 using 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.

c) **Convergence of the algorithm***:* Several solutions can be proposed for the convergence of our algorithm for regular flows (data flow 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 proves 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 the 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-layered algorithm the convergence can be improved: it suffices to randomly run Algorithm 2 and its adaptation in the previous paragraph.

V. Experimental results

In this section, we perform Algorithm 2 on two kinds of data flows. First, a flow whose messages arrive slower and slower, and then we show some numerical results based on the OSPF protocol. As said in the previous section, the parameters are $\sigma = 1$ for layers 0 and 1, $\sigma = 5$ for the next ones, and $T = 10$ s for every layer.

A. Slowing down data flow

We consider here a data flow generated the following way: initially, messages have inter-arrival times uniformly distributed between 1.6 and 2.4 s, and the traffic progressively slows down so that in the end, messages have an inter-arrival uniformly distributed on the interval $[16, 24]$. This simulation could characterize the case where a router is overloaded and then slowly communicates with its neighbors. Figure 6 represents such a data flow and Figure 7 shows the rates computed by the multi-layered version of Algorithm2 respectively for the ground layer and the third layer.

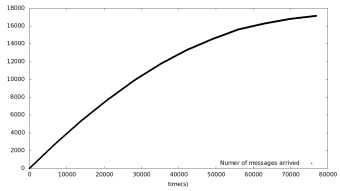

Fig. 6. Data flow where messages arrive slower and slower.

On both figures, one can observe the messages slowing down. Due to the fact that the fluctuations are quite important (40%), the rate computed on layer 0 frequently changes. But, concerning layer 3, the noise induced by the fluctuations is erased and only the global behavior is observed. This clearly illustrates the fact that the ground layer shows many details that are discarded by the next layers.

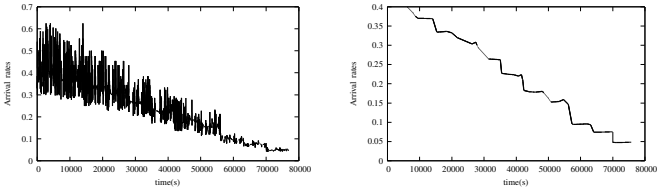

Fig. 7. Layer 0 and 3 of Algorithm 2 for a slowing down data flow.

B. OSPF communication link

Here, we perform Algorithm 2 on an OSPF flow monitoring on a telecommunication network whose 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.

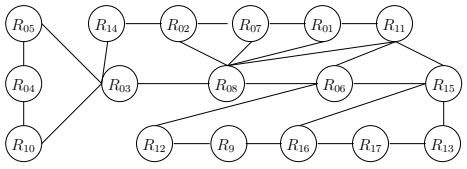

Fig. 8. Topology of the OSPF network studied.

We then define several scenarii to perform on the network: first, a normal and stable network with a fluid OSPF traffic, our reference behavior; second, a cyclic and malicious OSPF protocol stack router failure; and at last, 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. Experimentally, in our context, 3 layers are enough. Finally the resulting rates obtained allow us to prove its operational interest.

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 hold about the network topology. *LSA* sendings stop when all routers get 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.

2) OSPF fluid traffic: We study here an OSPF flow of messages when there is no perturbation in the network. More precisely, we look at the data flow from router R_{11} to router R_8. Figure 9 represents this data flow between time 0 and 8000 s. The box on the bottom right is a zoom on that flow between time 1500 and 2500 s. The flow seems to be globally linear. This corresponds to the sending of *Hello* messages. But, when having a closer look (zoom), one can observe that the linear behavior is perturbed between 1900 and 2250 s: the amount of arrivals intensifies. This scheme periodically happens (every 1800 s), corresponding to an *LSA refresh*.

A simple and natural approach to detect those perturbations would have been to use the *moving average method* (MAM). This method can be used to analyze a data flow: it can suppress instantaneous fluctuations and then study long term behaviors. For each messages, it computes the average inter-arrival time in a window composed of this message and the $N-1$ previous messages, where N is a fixed parameter. For message n, the inter-arrival is $\frac{1}{x_n - x_{n-1}}$.

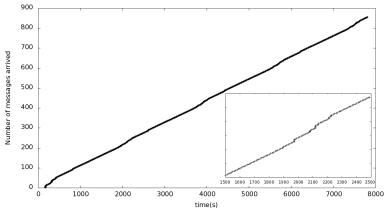

Fig. 9. Flow of messages from R_{11} to R_8 during 8000 s with a detailed period (between 1500 and 2500 s) at the bottom right.

In this paragraph, we compare the results given by Algorithm 2 and the MAM. Figure 10 shows the results for the ground layer (plain curve) and MAM with $N = 2$ (dotted curve). One can observe that these two curves are really close and both composed of a periodical scheme. Indeed, there are two kinds of behavior on the rates computed: there are stable periods during which rates are mostly equal to 0.1 during 1500 s and perturbed periods, lasting at most 400 s, with much higher rates. For each of these periods, our algorithm computes at most three rates in the stable phase and around four rates in the perturbed phase, whereas MAM always computes a linear number or rates in the number of messages.

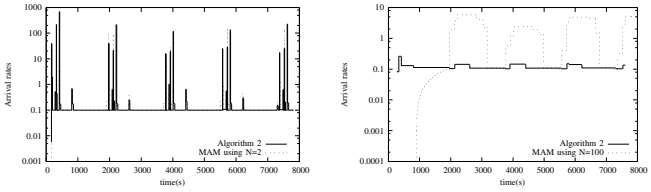

Fig. 10. Rates computation using Algorithm 2 on layer 0 and MAM with $N = 2$ on the left, and on layer 3 and MAM with $N = 100$ on the right when the traffic is fluid.

Figure 10 also compares the results of the third layer of our algorithm (plain curve) and the ones of MAM with $N = 100$ (dotted curve). One can notice that the two behaviors are still detected with the two methods. However the rates computed are now really different. Indeed, our algorithm represents each behavior by only one slope that represents the average messages arrival on the period. When only *Hello* messages are sent the slope is around 0.11 and when both *LSA* and *Hello* messages are sent the slope is around 0.14. MAM also detects the two behaviors, but the perturbed period last too long (around 1000 s). Furthermore it still outputs a linear number of rates: 68 output points for Algorithm 2 versus 850 points for MAM over the whole emulation. So our method presents here two advantages compared to the MAM: it requires less resources (time and space computation), and it is more accurate, as it detects the times at which changes occur in the data flow.

Moreover, the quality of the results of the MAM is highly sensitive to N, whereas our algorithm only considers the constraints of the flows. Other methods using sliding windows exist, like the *exponential moving average*, but they did not lead to any conclusive results. 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.

3) Periodic failure of a router: We study here a flow of OSPF messages when a failure occurs in router R_8 every 6 min and lasts 3 min. This corresponds to a bug in the implementation that forces R_8 to reboot frequently. We still look at the data flow from router R_8 to router R_{11}. Figure 11 represents the data flow on this link between time 0 and 4000 s.Initially, messages arrive regularly, which corresponds to *Hello* messages arrival. Then, between time 310 and 490 s, no message arrives. This is the period 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.

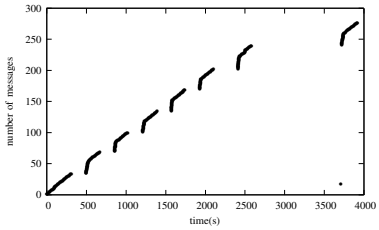

Fig. 11. Data flow from R_8 to R_{11} during 4000 seconds.

Figure 12 shows the result of layers 0 and 3 of Algorithm 2 on the flow of interest. Most of the time, the slope computed is $\rho = 0.1$, which corresponds to the sending 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.

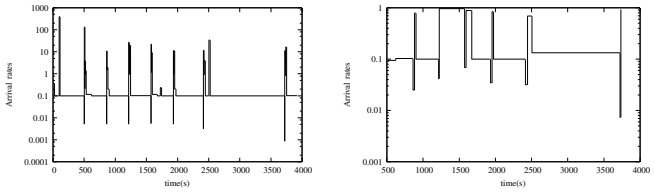

Fig. 12. Rates computed on layer 0 and 3 by Algorithm 2 on the data flow from R_8 to R_{11}.

Concerning layer 3, the resulting rates are smoother than the ones of layer 0. Here, rates do not exceed 1 when it could be 400 on 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 detected by an abrupt rate change.

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 13 presents an OSPF flow of message 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* 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.

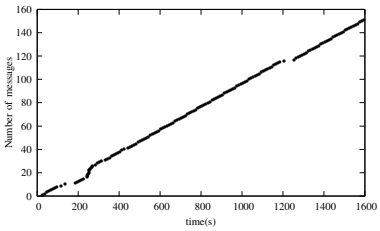

Fig. 13. Data flow from R_{11} to R_8 during 1600 seconds.

Figure 14 shows the ground layer and first 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.

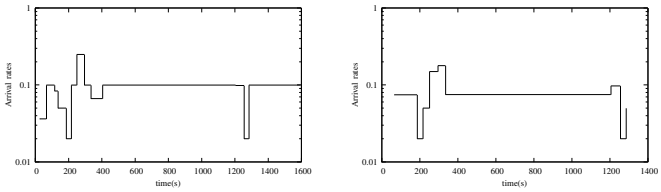

Fig. 14. Rates computed on layer 0 and 1 by Algorithm 2 on the data flow from R_{11} to R_8.

Concerning the first layer, 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.

VI. 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] A. Bouillard, A. Junier, and B. Ronot. Hidden anomaly detection in telecommunication networks. Technical report, RR-INRIA, 2012.

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

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[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.