DOI 10.1016/j.comcom.2017.03.008

Distributing Data Analytics for Efficient Multiple Traffic Anomalies Detection

Alba P. Vela*, Marc Ruiz, and Luis Velasco

Universitat Politècnica de Catalunya (UPC), Barcelona, Spain.
*Corresponding author: apvela@ac.upc.edu

***Abstract*—Traffic anomalies can create network congestion, so its prompt and accurate detection would allow network operators to make decisions to guarantee the network performance avoiding services to experience any perturbation. In this paper, we focus on origin–destination (OD) traffic anomalies; to efficiently detect those, we study two different anomaly detection methods based on data analytics and combine them with three monitoring strategies. In view of the short monitoring period needed to reduce anomaly detection, which entails large amount of monitoring data to be collected and analyzed in a centralized repository, we propose bringing data analytics to the network nodes to efficiently detect traffic anomalies, while keeping traffic estimation centralized. Once an OD traffic anomaly is detected, a network reconfiguration can be triggered to adapt the network to the new traffic conditions. However, an external event might cause multiple related traffic anomalies. In the case of triggering a network reconfiguration just after one traffic anomaly is detected, some Key Performance Indicators (KPI) such as the number of network reconfigurations and the total reconfiguration time would be unnecessarily high. In light of that, we propose the Anomaly and Network Reconfiguration (ALCOR) method to anticipate whether other ODs are anomalous after detecting one anomalous OD pair. Exhaustive simulation results on a realistic network scenario show that the monitoring period should be as low as possible (e,g., 1 min) to keep anomaly detection times low, which clearly motivates to place traffic anomaly detection function in the network nodes. In the case of multiple anomalies, results show that ALCOR can significantly improve KPIs such as the number of network reconfigurations, total reconfiguration time, as well as traffic losses.**

***Keywords*—Traffic anomalies; data analytics placement; network reconfiguration**

I. OD Traffic Anomalies

Traffic anomalies are short-living events that do not follow expected patterns (see a survey in [1]). They can create network congestion and stress resource utilization in packet nodes and hence, its prompt detection becomes essential since it allows preparing the network e.g., by reconfiguring the virtual network topology in multilayer network scenarios [2]. Anomaly detection can be used to trigger lightpath provisioning and network re-configuration when a traffic anomaly is detected [3], which entails analyzing monitoring data to anticipate traffic congestion. It is clear that developing efficient techniques to detect traffic anomalies in real time would empower network operators to prevent grave consequences induced by such anomalies affecting end users. However, detecting anomalies is a difficult task because anomalous patterns need to be extracted and interpreted from large amounts of high-dimensional, noisy data.

In order to detect traffic anomalies, it is essential to monitor traffic at the nodes and to model such traffic [4]. For packet networks specifically, the traffic monitoring function allows identifying (classifying) the traffic belonging to a specific service or destination, so as to apply specific policies. Monitoring traffic samples are produced at the packet nodes; according to the ITU-T [5], performance events are counted second by second over every 15-minute period. At the end of a period, they are collected in a repository for further analysis [2], [6]; for instance, data analysis can be used to create predicted traffic matrices for the near future; please refer to [7] for a list of use cases of traffic monitoring. Among use cases, that of identifying network failures [8], [9] and problems (or anomalies) is undoubtedly of the interest of many network operators. It is clear that when analytics are applied to data collected every 15 minutes, the expected traffic anomaly detection times will be as well in that order of magnitude. Consequently, the monitoring period should be reduced, which in turn increases the amount of monitoring data to be sent to the centralized data repository.

Many works in the literature can be found focused on intrusion and denial-of-service (DoS) detection (see e.g., [10]). Regarding traffic anomalies, authors in [11] proposed a general method that entails monitoring traffic in links and correlate monitoring time series to detect volume anomalies, identify the origin-destination (OD) pair, and estimate the amount of traffic involved in the anomalous OD pair. The method is based on applying Principal Component Analysis (PCA) to separate the multidimensional space occupied by a set of network traffic samples into

disjoint subspaces corresponding to normal and anomalous traffic conditions. Note that this method requires from previous collection and analysis of anomalous traffic, which is not always feasible due to both, its very rare occurrence and that traffic anomalies might not follow a predictable pattern.

Regarding traffic periodicity, authors in [12] analyzed the performance of several methods in detecting link traffic anomalies with respect to the traffic pattern on a 24h typical day. As traffic varies throughout the day, it is essential to consider the concrete traffic period in which the anomaly occurs. Authors in [13] suggested a model based on splitting a 24 hours day period into 16 non-overlapping intervals of 90 min. Their on-line change detection algorithm identified relevant changes in link utilization and reported those links to a centralized controller for further analysis. Finally, authors in [14] compared three traffic aggregation formalisms such as ingress routers, input links, and OD pairs and concluded that traffic aggregation level has a significant impact on the number of anomalies detected and on the false positive rate; they showed that aggregating traffic by OD pairs is the most appropriate choice.

Two different methods for anomaly detection are studied in this paper: *traffic-based* and *score-based*. Both methods have already been proposed in different contexts as in traffic changes detection [2] and anomaly detection based on hypothesis testing [15]; in this paper we adapt them for OD traffic anomaly detection. In addition, aiming at reducing the amount of monitored data to be conveyed to the central repository, we propose to dynamically configure the monitoring period. Once a traffic anomaly involving a single OD pair $o \rightarrow d$ has been detected, a virtual network reconfiguration can be immediately triggered to increase the capacity allocated for traffic between origin node $o \in R$ and destination node $d \in R$ (e.g., by setting up new connections on the underlying optical network), where R represents the set of nodes in the packet network.

Nonetheless, multiple anomalies can be caused under special circumstances, e.g., when a disaster affects the network [16]. In such scenarios, several ODs leaving from a common origin node ($o \rightarrow (M \subset R)$) or arriving to a common destination node ($(N \subset R) \rightarrow d$) can be produced, where M represents the set of destinations from o and N is the set of origins to d. Here, reconfiguring the virtual network after an individual OD traffic anomaly is detected can result in both, an intolerable number of virtual network reconfiguration and traffic losses and in a far from optimal network configuration in terms of resource utilization.

In view of the above, several architectures need to be studied to evaluate different data analytics algorithms placements to reduce all three: *i*) traffic anomaly detection time, *ii*) detect multiple related traffic anomalies and *iii*) the amount of monitoring data to be conveyed to the central repository. Note that the storage required for traffic traces has greatly expanded due to increasing network speeds [17]. Regarding data analytics placement, a similar study has been proposed in the context of global iceberg detection, e.g., distributed DoS attack; distributed monitors first measure local traffic for iceberg candidates and then, they report the measured datasets to a central server, which finds the most frequent ones [18].

In this work, we extend our previous works in [19] focused on single OD anomaly detection with different monitoring strategies and [20] devoted to study multiple OD anomalies and virtual network reconfiguration. Specifically, the contributions of this paper are the following:

1) Two different OD traffic anomaly detection methods together with three monitoring strategies are studied to efficiently detect OD traffic anomalies in section III. The traffic-based method uses predictive models to detect sequences of consecutive atypical traffic values, whereas the score-based method is a probabilistic classifier that considers both normal and atypical traffic data to measure how likely is to classify an OD as anomalous.
2) The Anomaly and Network Reconfiguration (ALCOR) method is proposed in section IV to improve the number of network reconfigurations, the total reconfiguration time, and traffic losses (key performance indicators (KPI)) in the event of multiple OD traffic anomalies. The method consists in anticipating whether other ODs are anomalous after detecting one anomalous OD pair.
3) Two architectural approaches are studied in sections III and IV, where the anomaly detection algorithm is placed in a centralized controller or distributed inside packet nodes. The monitoring parameters to be configured in the nodes and the data in the notifications that they have to send towards the controller are specified. In both architectures, ALCOR together with the repositories and additional modules are placed in the centralized controller since they need from data (monitored data and threshold notifications) from several nodes.

The discussion is supported by the results from exhaustive simulation over a realistic scenario in section V.

II. OD Traffic Anomalies

Before detecting OD traffic anomalies, traffic behavior needs to be firstly characterized. To this aim, some OD traffic models need to be fitted, so as to generate predictions against which monitored values can be compared. OD traffic models can be computed for the expected average by predicting two response variables: the *mean* (μ_{od}) and the *standard deviation* (σ_{od}). For the sake of simplicity, hereafter we use just μ and σ in the understanding that those refer to the mean and standard deviation, respectively for a specific OD pair.

Fig. 1 illustrates the main steps of the process. Monitoring traffic values for every OD pair are collected from the packet nodes at a given monitoring period, denoted as δ. OD traffic models are fitted with these data (Fig. 1a). Upper and lower bounds, computed as $\mu \pm 3\sigma$, and traffic samples for a given *od* pair and for a typical day are shown in Fig. 1b. Received monitored data can be now compared against its *od* traffic model and those out-of-bound values are considered as *atypical* (Fig. 1c). Notwithstanding, its detection does not entail a traffic anomaly evidence. In fact, the decision of whether an atypical sample is considered as a traffic anomaly cannot be based on just one single sample, but in observing some previous samples and computing a sort of likelihood of that *od* to be anomalous; we call this as *score*. Score values are compared against a defined threshold value (ε_A) and those scores exceeding such threshold are considered as anomalies. An example is depicted in Fig. 1d, where an anomaly is detected after receiving two atypical values and considering some other previous within-bound samples. As pointed out in the introduction, a virtual network reconfiguration can be triggered after an anomaly is detected so as to adapt its topology to the new traffic conditions.

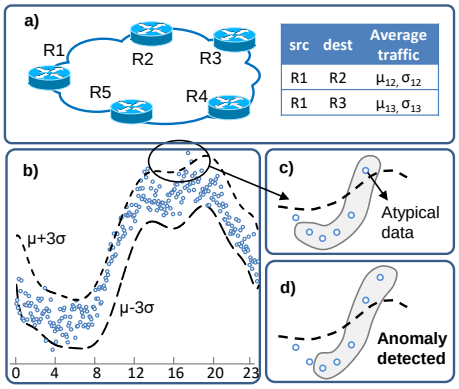


Fig. 1. a) Monitoring samples and estimation boundaries. b) Atypical. c) Atypical and d) anomaly detection.

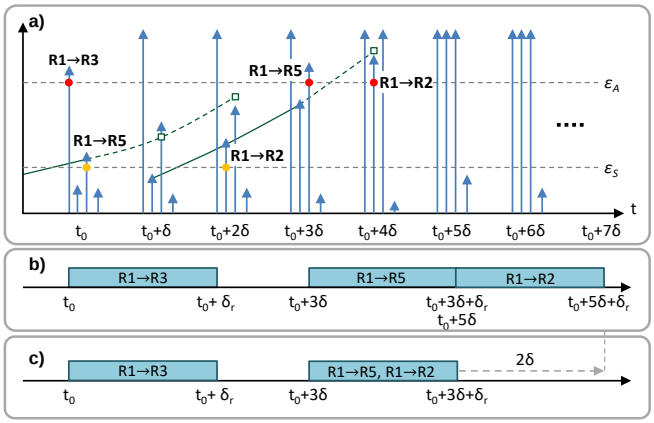


Fig. 2. Example of multiple anomalies. a) Score vs. time. b) Single and c) multiple OD reconfiguration.

Let us assume now that an event causes several related OD traffic anomalies. For illustrative purposes, Fig. 2 shows an example where a traffic anomaly has been detected in OD *R1→R3* (note that threshold ε_A has been violated at time t_0); the score evolution with time for several monitoring intervals is shown in Fig. 2a. Note that a purely *per-OD* reconfiguration would immediately trigger a network reconfiguration after the *R1→R3* OD traffic anomaly is detected (Fig. 2b). Let us assume that the monitoring period δ is fixed to 1 minute and the reconfiguration process takes δ_r, e.g., 2 minutes. Then, network reconfiguration process for OD pair *R1→R3* will end at $t_0 + \delta_r$. Later, at $t_0+3\delta$, an OD traffic anomaly in pair *R1→R5* is confirmed and thus, the controller triggers a new reconfiguration for such OD that finishes at $t_0+3\delta+\delta_r$. While the network reconfiguration for OD pair *R1→R5* is still in progress, another anomaly for OD pair *R1→R2* is detected at $t_0+4\delta$, but the reconfiguration for that OD anomaly needs to be delayed until that for OD pair *R1→R5* finishes. Therefore, reconfiguration for OD pair *R1→R5* starts at $t_0+5\delta$ and finishes at $t_0+5\delta+\delta_r$.

To reduce the number of reconfigurations and minimize the time when the last reconfiguration finishes thus reducing traffic losses, a procedure to detect multiple OD anomalies would be useful. To that end, we propose the ALCOR method to analyze other ODs and to compare their scores against a different threshold ε_S for suspicious ODs. ALCOR aims at improving the considered KPIs by reducing the number of network reconfigurations, as well as the total reconfiguration time in the case of multiple anomalies. This is achieved by avoiding the case when anomalies are detected while the reconfiguration triggered by other anomalies is still in process. Thus, ALCOR anticipates the reconfiguration of some suspicious ODs with sufficient evidence to shortly become anomalous. Since those anomalies are related, we propose to focus that analysis on outgoing OD pairs *R1→N*, as well as on incoming ODs *M→R3*.

In the example in Fig. 2, a threshold violation ε_S is detected for OD pair *R1→R5* after the detection of an anomaly in OD pair *R1→R3* at time t_0. After predicting the evolution of the *R1→R5* score by means of extrapolation based on past score values (lines with square markers in Fig. 2a), the occurrence of an anomaly during next 2δ in such suspicious OD pair is discarded and hence, reconfiguration of OD pair *R1→R3* is triggered at t_0. Later on, OD pair *R1→R5* anomaly arises and OD pair *R1→R2* is identified as suspicious. In this case, there is sufficient evidence to declare the latter as anomalous in the next time interval and consequently, it is jointly reconfigured with *OD* pair *R1→R5* at time $t_0+3\delta$. As a result, preparing the network for the new traffic conditions takes 2δ time units less than the total time required by the per-OD reconfiguration.

III. OD Traffic Anomaly Detection

A. OD traffic anomaly detection method

As stated in the introduction, two different methods for anomaly detection are studied: *traffic-based* and *score-based*, where the methods, already proposed in the literature, are adapted for OD traffic anomaly detection. The adapted *traffic-based* method consists in detecting anomalies after receiving a number of consecutive atypical monitoring data with respect to the $\mu\pm3\sigma$ confidence interval.

The *score-based* method is a probabilistic classifier with two labels for the response: normal and anomaly. The algorithm (see Table I) is based on a multi-response model to predict whether a sequence of consecutive traffic data belongs to the normal class or, on the contrary, there is sufficient evidence to declare it as anomalous. Since this method considers previous (not only atypical) traffic monitoring data, it has the capability of potentially anticipating traffic anomalies thus, reducing detection time compared to that of the traffic –based method. The algorithm starts when a traffic monitoring data of a given OD pair $y(t)$ is received in time t; let $\hat{y}(t)$ be the normalized value of $y(t)$ with respect to the average model, i.e.,:

$$\hat{y}(t) = \frac{y(t) - \mu(t)}{\sigma(t)}, \tag{1}$$

where $\mu(t)$ and $\sigma(t)$ are the mean and standard deviation, respectively, of the traffic model for such OD pair at time t.

Note that a normalized value equal to k means that $y(t) = \mu(t)+k\cdot\sigma(t)$. After normalization, $\hat{y}(t)$ is stored in a fixed-size data series $\hat{Y}$ containing the last m normalized traffic data received. Therefore, at a given time t, $\hat{Y}$ contains the following normalized traffic data:

$$\hat{Y}(t) = \left\{\hat{y}(t-i), \forall i \in 0..m-1\right\} \approx N\left(0_{mx1}, I_{mxm}\right), \tag{2}$$

where N represents the multivariate Gaussian distribution with mx1 zero vector mean and mxm identity covariance matrix. This multivariate distribution is the key result of normalizing traffic by means of μ and σ models. Note that the identity covariance matrix indicates unitary standard deviation for every single $\hat{y}$ value and no correlation between any pair of elements in $\hat{Y}(t)$. Hence, we can conclude that $\hat{Y}(t)$ contains independent and identically distributed random variables each following the univariate standard Gaussian distribution $z\sim N(0,1)$.

Table I. Score-based algorithm for traffic anomaly detection.

INPUT: $y(t)$, $\hat{Y}$
OUTPUT: *Anomaly*

1: $\hat{y}(t) \leftarrow$ Normalize ($y(t)$) (eq. (1))
2: remove oldest value from $\hat{Y}$
3: $\hat{Y}(t) \leftarrow$ add ($\hat{Y}$, $\hat{y}(t)$)
4: **if** $\hat{y}(t) < 3$ (atypical) **then**
5: **return** false
6: $s(t) \leftarrow$ computeScore ($\hat{Y}(t)$) (eq. (3))
7: **if** $s(t) < \varepsilon_A$ **then**
8: **return** false
9: **return** true

According to the aforementioned properties of the normalized traffic, let us define the probability $p(i)=1-P(z\leq|\hat{y}(t-i)|)$ as an indicator of how likely is to consider $\hat{y}$ as a normal traffic value. Therefore, we assume that smaller (i.e., less probable) $p(i)$ values will be observed in case of an anomaly. Based on these individual probabilities, we define a score function $s(t)$ to compute how likely is that a data series $\hat{Y}(t)$ does not belong to the normal class. The score $s(t)$ is defined as:

$$s(t) = \frac{1}{\sqrt[m]{\prod_{i=0..m-1}\left[1 - P\left(z \le \left|\hat{y}(t-i)\right|\right)\right]}} \tag{3}$$

In view of eq. (3), it is worth noting that the lower the probabilities of $\hat{y}$ variables, the lower the product of probabilities and inversely, the higher the score. To decide whether an anomaly is detected, we simply compare $s(t)$ against the ε_A threshold that normal data series $\hat{Y}(t)$ do not practically exceed. Then, there is sufficient evidence to detect an anomaly in OD pair in time t if $s(t) \geq \varepsilon_A$.

B. Proposed architecture and monitoring strategies

To efficiently implement OD-based traffic anomaly detection methods, we propose the modules depicted in Fig. 3 that are all of them, for the moment, assumed to be placed in the network controller. In such *centralized* architecture, traffic samples are collected from the packet nodes and stored in the collected data repository. Collected data can be conveniently summarized in modeled data, e.g., by computing average values. The Estimator module applies data analytics on samples from the modeled data repository to estimate the specific models for every OD pair, which are stored in a model repository. Models predict response variables for the average OD traffic (i.e., $\mu(t)$ and $\sigma(t)$). An anomaly detection module is in charge of detecting traffic anomalies; it first verifies whether a just arrived monitored OD traffic value is out of bounds and, only in such case, its current score is computed and compared against threshold ε_A. Upon the detection of an anomalous OD pair, the ALCOR module is in charge of deciding the ODs for which VNT reconfiguration needs to be triggered.

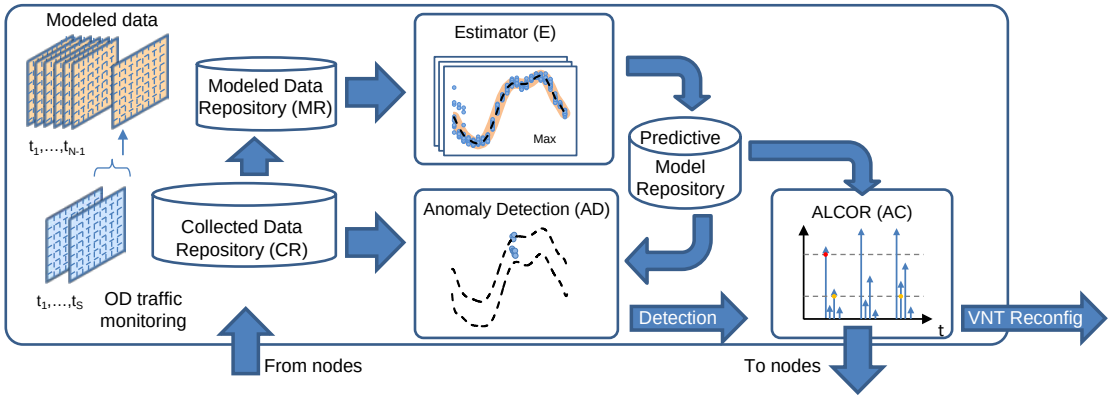


Fig. 3. Architecture for OD traffic anomaly detection.

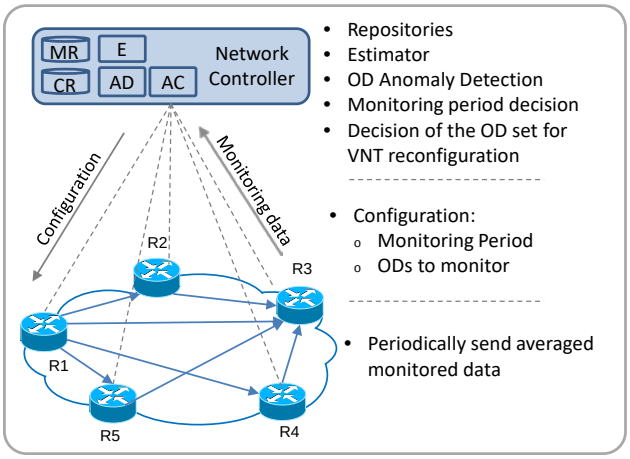


Fig. 4. Centralized monitoring architecture.

Fig. 4 presents a general view of the centralized architecture and details module placement, as well as the monitoring parameters that the network controller can configure in the network nodes. Analyzing the placement, it is clear that repositories need to be centralized, since data can be used for several purposes that might require a global view of the network. Regarding model fitting, it can be carried out in the controller from monitoring data collected every 15-minute period. However, that monitoring period would impact on the time to detect OD traffic anomalies

and hence, shorter monitoring period would be preferred from that viewpoint. For instance, if the target time to detect traffic anomalies is within the 5 minutes after they appear, the monitoring period cannot exceed that value (results are presented in section V). In consequence, δ is a key parameter to study since reducing it entails increasing the amount of monitoring data to be conveyed from the network nodes to the collected data repository in the controller.

Aiming at limiting the amount of collected data, in this paper we propose studying the performance of the following monitoring strategies: *i*) the traditional *fixed* monitoring period strategy but reducing its period to accelerate anomaly detection; *ii*) a *dynamic* monitoring strategy, where the monitoring period can be re-programmed during the day; and *iii*) a *reactive* monitoring strategy (*c*:*f*) that uses a coarse monitoring period (*c*) and re-configures it to a finer period (*f*) after detecting the first atypical monitoring data. From the possible combination of methods and strategies, we focus on studying the four most relevant approaches (Table II): *i*) traffic-based with fixed monitoring (*traffic-fixed*); *ii*) score-based with fixed monitoring (*score-fixed*); *iii*) score-based with dynamic monitoring (*dynamic*); and *iv*) score-based with reactive monitoring (*reactive*).

Table II. Anomaly detection methods and monitoring strategies

Monitoring Strategy	Detection method	
	Traffic-based	Score-based
Fixed	X	X
Dynamic		X
Reactive		X

Another different option is to place the traffic anomaly detection functionality inside the network nodes, as depicted in Fig. 5; we call this as the *distributed* architecture. This way, data analytics for anomaly detection has access to fine-grained monitoring data every δ period, which allows achieving low detection times while keeping a larger monitoring period (e.g., 15 minutes) for model fitting thus, reducing the amount of monitoring data to be collected.

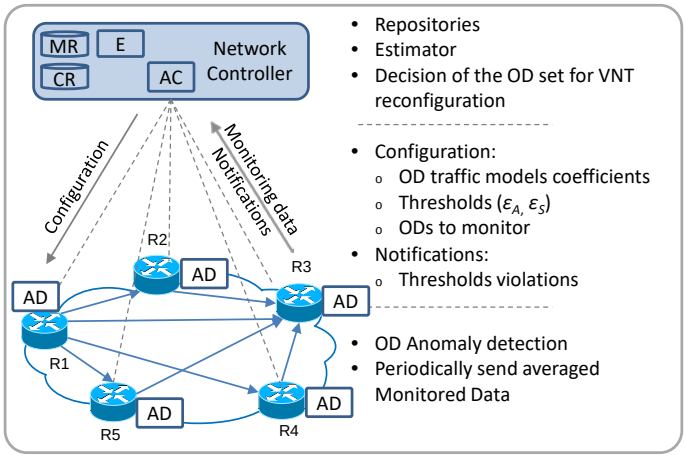


Fig. 5. Distributed monitoring architecture

IV. The Anomaly and Network Reconfiguration (ALCOR) method

This section presents the ALCOR method that identifies multiple related anomalies; the score value defined in section III is used to decide whether to trigger a network reconfiguration with only confirmed anomalous OD pairs or anticipate further anomaly evidences in suspicious OD pairs. Assuming the distributed architecture (Fig. 5), ALCOR configures every node in the network specifying, among other parameters, the anomaly detection thresholds (ε_A and ε_S). Initially, nodes only monitor outgoing OD traffic and periodically send average values towards the controller, which estimates the coefficients to model OD traffic (Fig. 6a). The coefficients for every OD pair are forwarded to the origin and destination nodes and are used together with the ε_A threshold to detect traffic anomalies. Whenever an OD traffic anomaly is detected by a node (i.e., OD pair R1→R3 in Fig. 6), a notification is sent to the controller and, as a result, ALCOR decides to activate the notification for threshold ε_S in origin node for all outgoing OD pairs $o \rightarrow N$, as well as to activate traffic monitoring for the incoming ODs $M \rightarrow d$ and ε_S threshold notification in the destination node (i.e., R1→{R2, R3, R4, R5} and {R1, R2, R4, R5}→R3 in Fig. 6b).

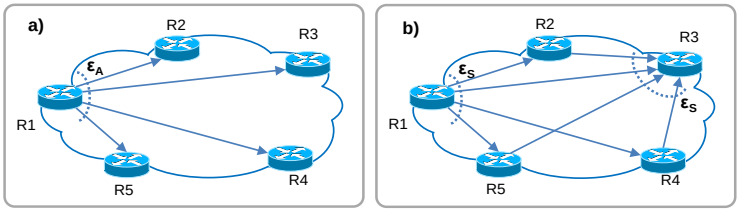


Fig. 6. a) OD pair R1→R3 anomaly detection. b) Anomaly threshold notification reconfigured in R1 and R3.

ALCOR decision problem can be addressed by solving the algorithm presented in Table III. The algorithm receives an *od* in the set of OD pairs in the network, for which an anomaly threshold *thr* (either ε_S or ε_A) has been exceed at time *t* and returns a set of tuples with the OD pairs to be reconfigured together with the threshold violated and the time of such event. Note that in normal conditions, network nodes are configured to send notifications only after threshold ε_A is exceeded and therefore, no notification is sent when the score for any OD pair exceeds ε_S. Only when one network node notifies about an OD pair threshold violation some network nodes will be configured to send notifications for ε_S violations in some of the monitored OD pairs. In that regard, line 1 in Table III checks whether ALCOR can start making decisions.

In case that no decision process is started yet, auxiliary sets are initialized and variable *decisionOngoing* is set (lines 2-4 in Table III). Next, the network controller configures monitoring parameters to detect ε_S violations for every other OD pairs leaving node *o* and entering node *d* (line 5). Note that initially only outgoing OD traffic was monitored to detect ε_A violations. Sets of monitored ODs are updated, containing those ODs being monitored for threshold ε_A (M_A), outgoing ODs for threshold ε_S (M_{So}) and incoming ODs for threshold ε_S (M_{Si}). The partial solution *Sol* is updated with the detected anomaly and a timer is started waiting for new evidences (lines 7-8). When the controller receives an ε_S violation, *Sol* is updated and timers disabled (lines 10-12).

The function *decideReconfiguration*(·) detailed in Table IV is used to select which ODs in *Sol* needs to be actually reconfigured. Recall that *Sol* contains both anomalous and suspicious ODs; the former subset will be always reconfigured (lines 3-4 in Table IV) whereas ODs in the latter group are analyzed one by one and selected (or not) for reconfiguration (lines 6-12). Specifically, the normalized monitored traffic $\hat{Y}$ is used to estimate the expected normalized traffic for the next δ_r time. Although normalized traffic presents a stationary behavior around zero mean under normal conditions, the presence of an anomaly dramatically alters this behavior causing sharp increasing trend during anomaly lifetime.

The procedure to predict future normalized traffic is based on linear extrapolation from observed $\hat{Y}$ data series assuming that observations in $\hat{Y}$ are linearly correlated with time, i.e., $\hat{y}(t)$ values in $\hat{Y}$ can be expressed a linear function of *t*. To guarantee that linearity, we obtain $\hat{Y}'$ as the transformed $\hat{Y}$ data series after applying a Box-Cox transformation [21] (line 7). In brief, that procedure finds the transformation parameter λ that returns $\hat{Y}'$ with the highest linear correlation with respect to time. Once λ is obtained, every $\hat{y}'$ in $\hat{Y}'$ is computed as follows:

$$\hat{Y}'(t) = \begin{cases} \dfrac{\hat{y}(t-i)^{\lambda} - 1}{\lambda}, \forall i \in 0..m-1 & \lambda \neq 0 \\ \log\left(\hat{y}(t-i)\right), \forall i \in 0..m-1 & \lambda = 0 \end{cases} \tag{4}$$

Fig. 7 shows examples of data series $\hat{Y}$ under normal traffic and under an anomaly, where plots show values in $\hat{Y}$ (markers), as well as the trend line of those values with respect to time (dashed line). Under normal traffic (Fig. 7a), normalized traffic values present stationary behavior around 0, as well as no linear correlation. However, under unexpected traffic $\hat{Y}$ shows a remarkable increasing trend with time, which might be non-linear. We propose the Box-Cox transformation in equation (4) to prepare data for linear regression fitting (Fig. 7c). Therefore, linear extrapolation can likely predict expected future normalized traffic in the event of an anomaly.

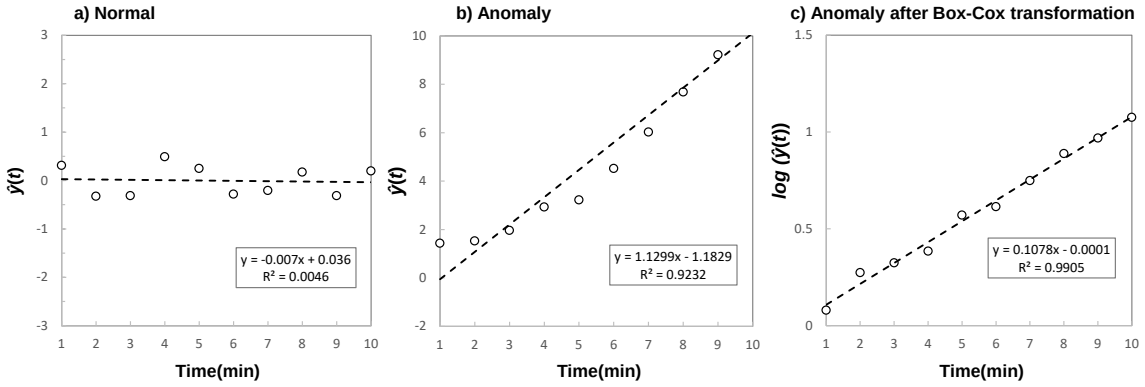


Fig. 7. Example of data series $\hat{Y}$ under normal traffic (a) and under an anomaly (b). Box-Cox transformation is applied in (c).

Table III. ALCOR algorithm.

INPUT: *od, thr, t*
OUTPUT: *Sol*

```
1:  if decisionOngoing=false AND thr= εS then return Ø
2:  if decisionOngoing=false then
3:      Sol ←Ø; MSi ←Ø; MSo ←Ø
4:      decisionOngoing = true
5:  if thr= εA then
6:      <MA, MSi, MSo>←reconfigureODMonitoring(MA, MSi, MSo, od)
7:      Sol ←Sol ∪ {<od,thr,t>}
8:      configureTimer (time, ALCOR(Ø, 0, 0))
9:      return Ø
10: if od≠ Ø then
11:     Sol ←Sol ∪ {<od,thr,t>}
12:     removeTimer()
13: Sol←decideReconfiguration(Sol,t) then
14: removeMonitoring(MSi)
15: configureMonitoring(MSo, εA)
16: MA ←MA ∪ MSo
17: decisionOngoing = false
18: return Sol
```

Table IV. *decideReconfiguration* algorithm.

INPUT: *Sol, t*
OUTPUT: *Sol**

```
1:  Sol* ← Ø
2:  for each i in Sol do
3:      if i.thr = εA then
4:          Sol* ← Sol* ∪ {i}
5:      else
6:          Ŷ(t) ← getNormalizedTraffic(i.od)
7:          Ŷ'(t) ← BoxCoxTransform(Ŷ(t))
8:          for k=1.. δr do
9:              ŷ'(t+k) ← linearExtrapolation(Ŷ'(t+k-1))
10:             ŷ(t+k) ← BoxCoxTransform-1(ŷ'(t+k))
11:             Ŷ(t+k) ← update(Ŷ(t+k-1), ŷ(t+k))
12:             s(t+k) ← computeScore(Ŷ(t+k))
13:             if s(t+k)≥εA then
14:                 Sol* ← Sol * ∪ {i}
15:                 break
16: return Sol*
```

Normalized traffic for the future δ_r time is then forecast (line 8) and the obtained value is used to update $\hat{Y}$ (lines 9 and 10); updated $\hat{Y}$ contains monitored normalized traffic in the first *m-x* positions and forecast normalized traffic in the last *x* positions. This data is used to compute the predicted score (see section III) and compared against ε_A (lines 11 and 12).

V. Illustrative Results

In this section, we first study the performance of the proposed strategies and methods for traffic anomaly detection, where each anomaly affects one single OD pair. Next, we apply ALCOR when multiple anomalies arise together and study the obtained performance.

A. Scenario

For evaluation purposes we developed an ad-hoc event-driven simulator in OMNET++ and considered a scenario with ten packet nodes (i.e., 90 OD pairs). OD traffic and traffic anomalies were generated separately and subsequently combined (Fig. 8). Traffic (Fig. 8a) is generated as the summation of two different functions: mean and noise, where traffic mean represents a normal day with values varying along day hours and noise is a random function with mean zero and a given standard deviation. Regarding anomalies, they are generated following a pulse function (Fig. 8b), where the raising front consists of an exponential function and are used as a multiplicative factor over traffic (Fig. 8c). Anomalies can be configured to be triggered at any specific time and with any specific duration and scaling factor. As an example, an anomaly can be generated to multiply traffic by x1.5, last for two hours and reach 90% of its maximum value at the first 30 minutes.

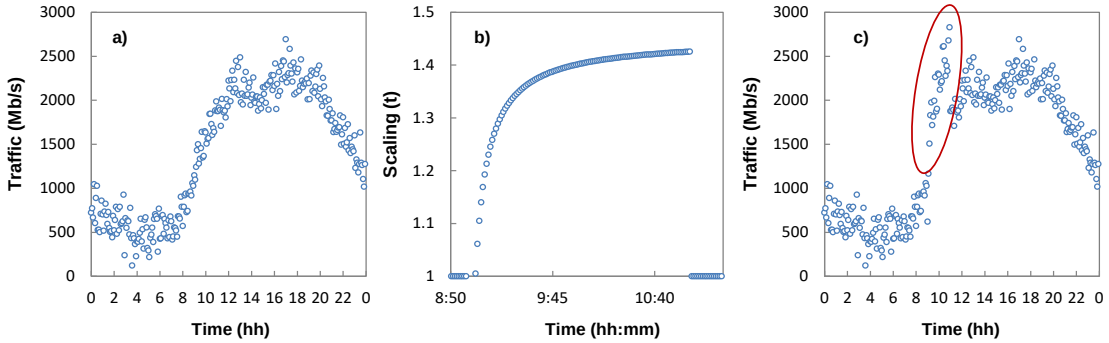


Fig. 8. a) OD traffic profile in a typical day. b) Traffic anomaly at 9am. c) OD traffic with anomaly.

The Estimator module was implemented in C++ and integrated in the simulator, whereas the ALCOR module implementing the anomaly detector was developed in R and kept as a separated standalone module. Generated traffic values were used as input of an R function that computed the score of every OD pair and returned whether an OD exceeded a threshold.

Before evaluating the performance of the previously proposed strategies and methods, some key parameters need to be determined. To this aim, we run some simulations without adding traffic anomalies so as to produce monitoring data for the normal traffic. From those simulations, we observed that the maximum amount of consecutive atypical monitoring traffic data in an OD pair was 2. In consequence, anomalies will be detected by the *traffic-based* method proposed in section III when 3 or more consecutive atypical values are monitored. Regarding the *score-based* method, we considered m=5 and δ_r = 2 min.

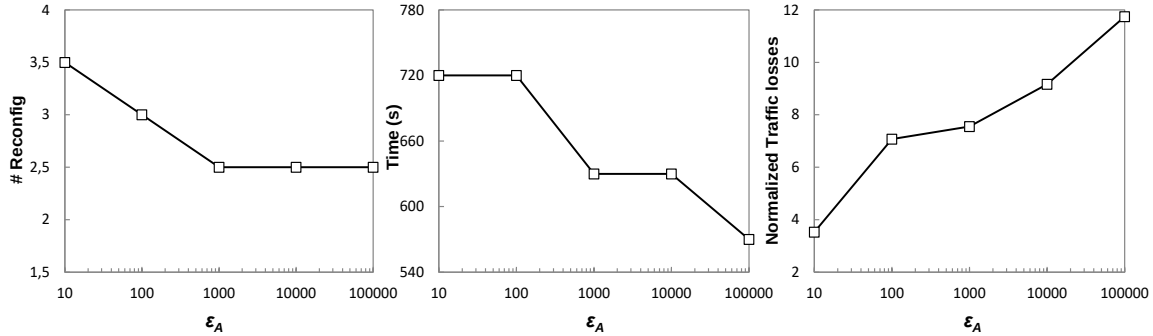


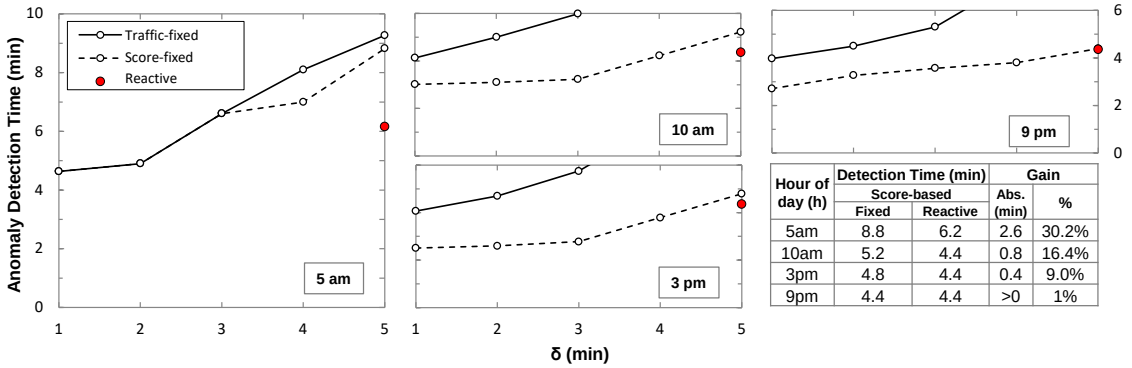
Fig. 9. KPIs vs ε_A.

To tune ε_A and ε_S values, we analyzed their impact on the defined KPIs. The choice of the ε_A was focused on minimizing the values of these KPIs. Regarding the value of ε_S, we selected a value of 10 that is high enough for the sample to be out of the interval $\mu \pm 3\sigma$ and it is low enough to allow that many OD traffic pairs to be considered by the ALCOR method. To tune ε_A we run experiments to study how the influence of that threshold on the selected

KPIs: *i*) the number of reconfigurations that need to be performed when multiple anomalies arrive; *ii*) the total time required to detect all the anomalies and to reconfigure the network; and *iii*) normalized volume traffic that is lost due to network congestion before reconfiguration finishes (we assume that all traffic exceeding 3σ will be lost). Fig. 9 shows the results; we can conclude, in view of the figure, that $\varepsilon_A = 1000$ provides the best overall results.

B. Single OD Traffic Anomalies

Graphs in Fig. 10 plot, for several hours of the day, the anomaly detection time for the considered detection methods and monitoring strategies, where the monitoring period is in the interval [1-5] minutes. We observe that although anomaly detection time varies for the different considered hours of day, detection time increases remarkably with the *traffic-fixed* approach when the monitoring period increases. This is in contrast to the moderated increment achieved by the s*core-fixed* one. In the case of the *reactive* approach, where we assume a (*c*=5:*f*=1) min. monitoring strategy, slightly lower detection times with respect to the previous approaches can be observed. The table in Fig. 10 reports the gains in detection time for the studied hours of day, where using a finer monitoring period after an out-of-bound traffic sample is detected provides gains between 1% and 30%.



Hour of day (h)	Detection Time (min)		Gain	
	Score-based		Abs. (min)	%
	Fixed	Reactive		
5am	8.8	6.2	2.6	30.2%
10am	5.2	4.4	0.8	16.4%
3pm	4.8	4.4	0.4	9.0%
9pm	4.4	4.4	>0	1%

Fig. 10. Traffic anomaly detection time vs. monitoring period for different hours of a day.

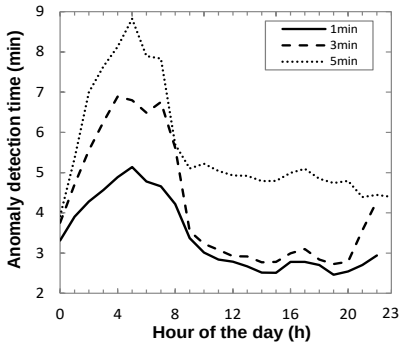


Fig. 11. Anomaly detection time vs. hours of day for different δ values.

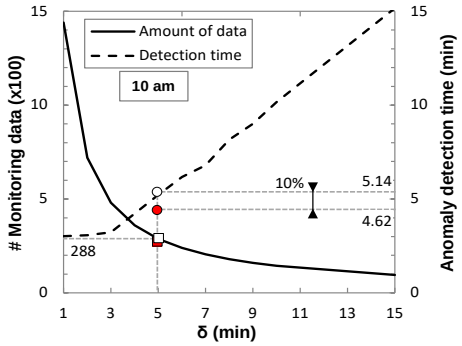


Fig. 12. Amount of collected data and anomaly detection time vs. δ.

Fig. 11 focuses on studying in depth the potentials of the score-fixed and illustrates how anomaly detection depends on different factors such as the changes on traffic volume among the different hours of the day for the same monitoring period. This opens the opportunity to dynamically adapt the monitoring period for different hours of day and achieve the same anomaly detection times (Dynamic monitoring). Note that this is positive since to achieve low detection times, 1-minute period should be fixed. Hence, by relaxing the monitoring period we are effectively reducing the amount of monitoring data to be collected in the centralized repository.

Fig. 12 shows the amount of monitored data to be collected along the day when reducing δ. E.g., assuming a 5 minutes period, 288 monitoring samples per OD and day need to be collected achieving 5.14 and 4.62 min. detection times for the score-fixed and the reactive approaches, respectively. Finally, Table V compares the amount of data to be collected in the centralized and the distributed architecture as a function of the required detection time for both, anomaly detection and traffic modelling. Note that the amount of data in the case of the distributed architecture, is constant and equal to one sample every 15 min, since monitored data is exclusively used for traffic modelling purposes.

Table V. Amount of collected data per OD and day vs required anomaly detection time

Max Detection Time (min)	Centralized Architecture	Distributed Architecture
3	720	96
5	360	96
10	144	96
15	96	96

Even though the different proposed methods for anomaly detection can be improved by changing the monitoring period, the best solution to avoid dealing with huge amount of monitored data is clearly to place the traffic anomaly detection directly into the network node, as proposed in section III.B.

C. Multiple OD Traffic Anomalies

Let us first evaluate the accuracy of the decisions that ALCOR makes based on traffic linear extrapolation. Recall that ALCOR applies traffic linear extrapolation and the result is used as input to compute the expected score and makes reconfiguration decisions based on that forecast data (see *decideReconfiguration*(·) function in Table IV). We define that a decision in time t is incorrect if the *decideReconfiguration*(·) function predicts all the scores in the interval $[t+1, t+\delta_r]$ for a given OD pair to be below ε_A (reconfiguration decision is not made at time t), while later on some score within that interval actually exceeds ε_A (reconfiguration decision is made at time in $[t+1, t+\delta_r]$). In our simulations, the *decideReconfiguration*(·) function was triggered 300 times, resulting in incorrect decisions in only 5.65% of the times. Therefore, the goodness-of-fit of the traffic linear extrapolation model for score prediction purposes is verified.

Let us now consider multiple anomalies affecting different OD pairs. As concluded in the previous subsection, we assume the distributed monitoring architecture. In this regard, let us study the performance of the proposed ALCOR method and compare that against a purely per-OD reconfiguration strategy. In addition, to evaluate the performance of the predictive capacity of ALCOR, we compare its performance against running the ALCOR algorithm under a perfect information assumption (ALCOR-PI), where the decision of reconfiguration is made with perfect knowledge of the future, i.e., it makes perfect decisions.

In order to evaluate the performance of the proposed ALCOR algorithm, we focus on the KPIs defined: *i*) the number of reconfigurations; *ii*) the total time required to reconfigure the network; and *iii*) normalized volume traffic that is lost. Graphs in Fig. 13 and Fig. 14 present the results for two different scenarios, when four traffic anomalies arrive close in time one to the other (scenario 1) or more spaced (scenario 2). Fig. 13a and Fig. 14a plot the evolution of normalized traffic values as a function of the time; under normal conditions they should be centered on zero (the mean value) and within some interval in terms of σ. When a traffic anomaly occurs in an OD, the normalized traffic changes sharply. Notwithstanding that sharp variation, it is difficult to set thresholds for normalized traffic values since the probability of observing values out of normal boundaries (e.g., 3σ) is not negligible and therefore, the score presented in section III that considers previous traffic values is used. Fig. 13b and Fig. 14b plot the computed scores against time for the four affected ODs under the considered scenarios. Note the difference between normalized traffic and score for OD pairs 6→9 and 6→2 in scenario 1 (Fig. 13 a); although the normalized traffic value is around 2σ at minute 5 and minute 8 for OD pairs 6→9 and 6→2, respectively, their score values are under ε_S for OD pair 6→9 and above ε_S for OD pair 6→2.

Comparing both scenarios, it is clear that ALCOR would made different decisions in each case. For instance, in scenario 1, when the first anomaly in OD pair 6→9 is detected at minute 7, OD pair 6→2 has already exceeded ε_S threshold. In this scenario, ALCOR would detect that and run the *decideReconfiguration*(·) function. To clarify score extrapolation, Fig. 15 plots the score values computed when an anomaly has been detected (dark markers) and the predicted score values for the next two minutes (dotted markers) for those OD pairs with scores exceeding ε_S threshold. Then, ALCOR will find that OD pair 6→2 will not become anomalous in the next 2 minutes (see predicted scores for OD pair 6→2 in Fig. 15) and it will trigger a network reconfiguration only for OD pair 6→9. Later, at minute 10 OD pair 6→2 violates ε_A; at that time, score extrapolation anticipate that OD pairs 6→1 and 6→8 will become anomalous in the next 2 minutes (see Fig. 15), so ALCOR triggers a network reconfiguration for all three OD pairs. Interestingly, the *per*-OD reconfiguration would need 3 reconfigurations, 1 more than ALCOR and delays the network adaptation in 2 minutes.

Similarly, the first OD traffic anomaly is detected in pair 6→9 at minute 4 under scenario 2. Nonetheless, since no evidences of other anomalous ODs are found, ALCOR decides to trigger a reconfiguration for that OD pair. When a

new OD traffic anomaly is detected in pair 6→2 at minute 10, OD pair 6→8 also violates ε_S threshold and shows evidences of becoming anomalous in the next 2 minutes, so ALCOR triggers a new reconfiguration for both ODs. Finally, a third reconfiguration is triggered for OD pair 6→1 at minute 15. Note that the *per*-OD strategy needs four reconfigurations and, although the network would be adapted to the new conditions at the same time as in the case of ALCOR, the *per*-OD strategy increases traffic losses in a 27%.

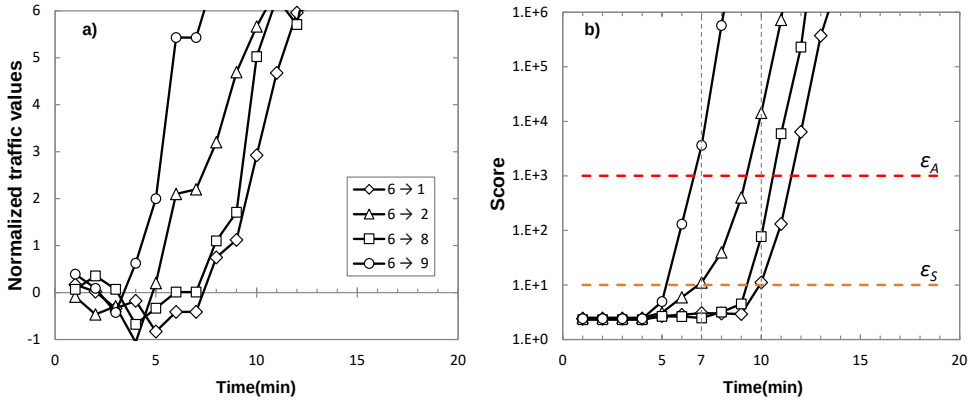


Fig. 13. Normalized traffic and score values against time for the anomaly scenario 1.

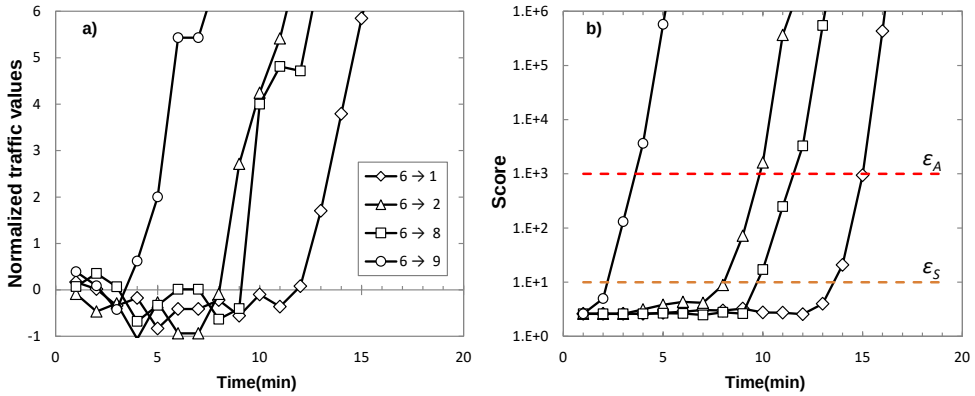


Fig. 14. Normalized traffic and score values against time for the considered for the anomaly scenario 2.

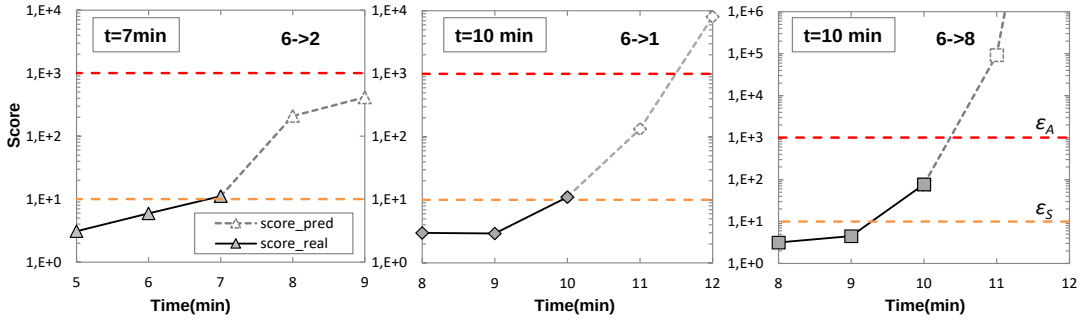


Fig. 15. Real and predicted score values against time for the three OD pairs in anomaly scenario 1.

In view if the above, it is interesting to study the performance of ALCOR under several traffic anomaly inter-arrival times and let us consider the scenario where six ODs are affected and assuming an anomaly scaling factor x2.

Fig. 16 shows that applying ALCOR algorithm results in an improved in the defined KPIs, i.e., reduction in the number of reconfigurations (Fig. 16a), total reconfiguration time (Fig. 16b), and traffic loss (Fig. 16c), compared to the per-OD strategy even when anomaly inter-arrival time is as high as 3 min (larger than that of the considered reconfiguration time). Table VI summarizes the gains from applying ALCOR.

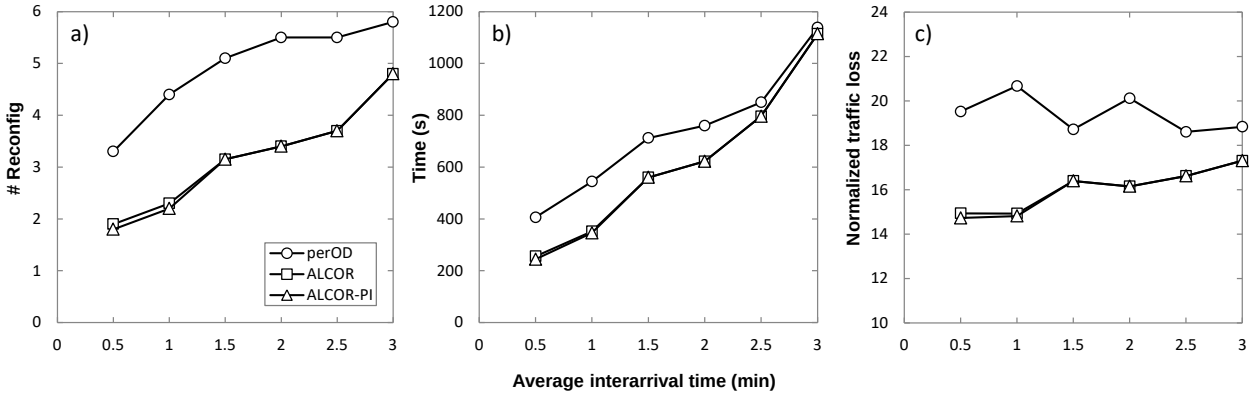


Fig. 16. (a) Number of reconfigurations, (b) total reconfiguration time, and (c) normalized traffic loss against anomalies inter-arrival time

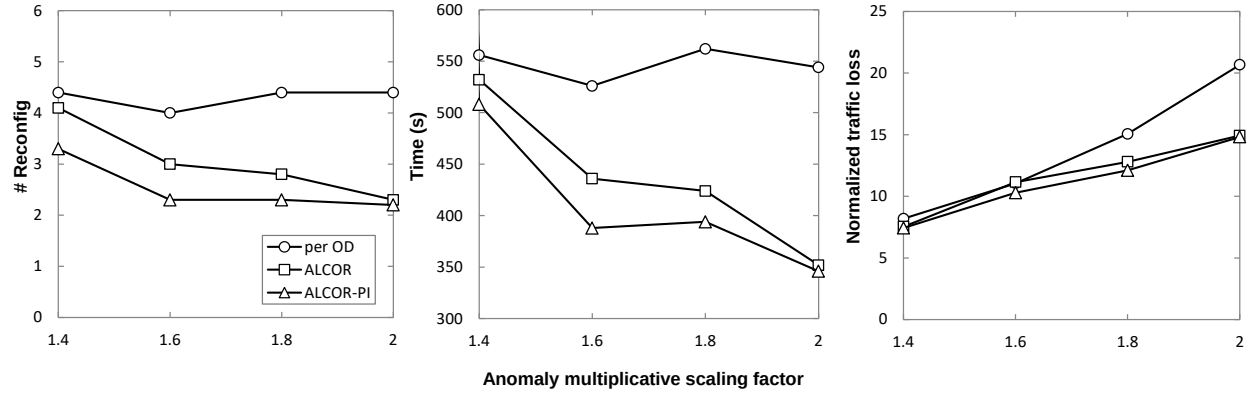


Fig. 17 Number of reconfigurations (a), total reconfiguration time (b) and normalized traffic loss (c) against the anomalies scaling factor.

Table VI. Gains from applying ALCOR (%).

Avg. Interarrival time (min)	#Reconfigs	Time	Traffic Loss
0.5	45.5	39.9	23.5
1.0	47.7	36.4	28.3
1.5	38.2	21.3	12.0
2.0	38.2	18.2	19.7
2.5	32.7	6.4	10.6
3.0	17.2	0	8.1

Interestingly, ALCOR performs virtually like ALCOR-PI; however, this could be as a result of the high value of the considered anomaly scaling factor, so let us analyze the influence of the traffic anomaly intensity on the performance of ALCOR (Fig. 17). We studied the range for anomaly scaling factors from x1.4 up to x2, considering that lower scaling factors can hardly considered as traffic anomalies since they produce traffic values within the limits of normal traffic (see Fig. 1b). One can observe how the intensity affects both the number of reconfigurations (Fig. 17a), the total reconfiguration time (Fig. 17b), and traffic loss (Fig. 17c), showing ALCOR a better performance than that of the per-OD approach in all the cases and getting closer to the ALCOR-PI method as soon as the scaling factor increases.

VI. Concluding Remarks

Traffic anomalies affecting OD pairs are known to disturb the correct network operation, so it is of paramount importance its prompt detection. In this paper, two different anomaly detection methods have been studied: *traffic-based* and *score-based*, where the first consists in detecting anomalies after receiving a number of consecutive atypical monitoring data with respect to the $\mu \pm 3\sigma$ confidence interval, while the latter is a probabilistic classifier that assigns two labels for the response: normal and anomaly. Besides, an algorithm based on a multi-response model has

been presented to predict whether a sequence of consecutive traffic data belongs to the normal class or, on the contrary, there is sufficient evidence to declare it as anomalous.

A traffic anomaly detection architecture has been proposed based on the following components: *i*) traffic samples from every OD pair are collected at a given rate and stored in the collected data repository; *ii*) collected data is used by a traffic anomalies detection module, to compare its value against that of a predicted traffic model; *iii*) an estimator module that pre-processes collected data to fit multi-response predictive traffic models for the average. Predictive models include both, mean and deviation as response variables. All these components were first assumed to be placed centralized in the network controller.

Different function placement architectures and monitoring strategies were explored for the data analytics to perform OD traffic anomaly detection aiming at reducing anomaly detection times. It was shown that 15-minute monitoring cannot provide the short anomaly detection times required to react against unexpected traffic changes. Consequently, four different approaches mixing detection methods and monitoring strategies have been proposed using the centralized architecture; the shortest anomaly detection times were achieved when monitoring every 1 min. However, this represents a large amount of data storage in the centralized controller. In view of the above, an alternative distributed architecture was proposed where the proposed data analytics method for OD anomaly detection were moved to the network nodes thus, relaxing data collection from the centralized controller to the traditional 15-min. period, that can be used for traffic modelling and estimation purposes.

After detecting an OD traffic anomaly, a network reconfiguration can be triggered to avoid traffic losses as a result of capacity exhaustion. However, in the event of multiple related anomalous OD pairs, its detection can be spread along the time, which would entail unnecessary number of network reconfigurations and long total reconfiguration time. In view of that, the ALCOR method was proposed to identify multiple related OD traffic anomalies targeting at reducing traffic losses coming from unexpected traffic increments, as well as the number of network reconfigurations performed to consequently adapt the network capacity. Since ALCOR needs to access monitoring data from several network nodes, it was placed in the centralized controller. Simulation results show remarkable saving in the number of reconfigurations, the total reconfiguration time and the traffic losses compared to a per-OD reconfiguration strategy.

The proposed distributed architecture presents some open aspects that can be summarized as follows: *i*) the network nodes need to be upgraded to introduce the anomaly detection functionality. This entails that the architecture of the network nodes needs to be extended to support extended monitoring and data analytics software, as well as monitoring programmability (i.e., monitoring parameter configuration). In addition, the architecture of the network controllers needs extensions to support modeling traffic from monitoring data, data analytics to make decisions (e.g., to initiate a network reconfiguration), monitoring programming, etc. *ii*) Related to monitoring programmability, the distributed architecture needs more monitoring parameters to be specified (i.e., per-OD traffic models and score thresholds), which requires extending the interface between the controller and network nodes. *iii*) In the migration process from nodes with and without extended capabilities, the data plane could include both node types, which would entail increasing the complexity of managing the data plane. In such cases, a hierarchical approach [22] can be implemented, where monitoring data from non-extended nodes can be collated in some intermediate element implementing the extended capabilities.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Spanish MINECO SYNERGY project (TEC2014-59995-R) and from the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kum “Anomaly Detection: A Survey,” ACM Computing Surveys, vol. 41, pp. 1-72, 2009.

[2] F. Morales, M. Ruiz, Ll. Gifre, L. M. Contreras, V. López, and L. Velasco, “Virtual Network Topology Reconfiguration based on Big Data Analytics for Traffic Prediction,” (Invited) IEEE/OSA Journal of Optical Communications and Networking (JOCN), vol. 9, pp. A35-A45, 2017.

[3] L. Velasco, A. P. Vela, F. Morales, and M. Ruiz, “Designing, Operating and Re-Optimizing Elastic Optical Networks,” (Invited Tutorial) IEEE/OSA Journal of Lightwave Technology (JLT), DOI: 10.1109/JLT.2016.2593986, 2017.

[4] C. Liu, M. Malboubi, and C-N. Chuah, “OpenMeasure: Adaptive Flow Measurement and Inference with Online Learning in SDN,” IEEE Global Internet Symposium, 2016.

[5] ITU-T Recommendation M.2120, 2002.
[6] E. Masala, A. Servetti, S. Basso, and J.C. De Martin, “Challenges and Issues on Collecting and Analyzing Large Volumes of Network Data Measurements,” in New Trends in Databases and Information Systems, 2014.
[7] M. Linsner, P. Eardley, and F. Sorensen, “Large-Scale Broadband Measurement Use Cases,” IETF RFC 7536, 2015.
[8] M. Ruiz, F. Fresi, A. P. Vela, G. Meloni, N. Sambo, F. Cugini, L. Poti, L. Velasco, and P. Castoldi, “Service-triggered failure identification/localization through monitoring of multiple parameters,” in Proc. European Conference on Optical Communication (ECOC), 2016.
[9] A. P. Vela, M. Ruiz, F. Fresi, N. Sambo, F. Cugini, L Velasco, and P. Castoldi, “Early Pre-FEC BER Degradation Detection to Meet Committed QoS,” accepted in IEEE/OSA Optical Fiber Communication Conference (OFC), 2017.
[10] M. Bhuryan, D. Bhattacharyya, and J. Kalita, “Network Anomaly Detection: Methods, Systems and Tools,” IEEE Communications Surveys & Tutorials, vol. 16, pp. 303-336, 2014.
[11] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing Network-Wide Traffic Anomalies,” in ACM SIGCOMM, 2004.
[12] A. Cuadra-Sanchez, J. Aracil, and J. Ramos de Santiago, “Proposal of a new information-theory based technique and analysis of traffic anomaly detection,” International J. of Parallel Emergent and Distributed Systems, vol. 30, pp. 1-14, 2015.
[13] F. Mata, J. García-Dorado, J. Aracil, “Detection of traffic changes in large-scale backbone networks: The case of the Spanish academic network,” Elsevier Computer Networks, vol. 56, pp. 686–702, 2012.
[14] A. Soule, F. Silveira, H. Ringberg, and C. Diot, “Challenging the supremacy of traffic matrices in anomaly detection,” in Proc. ACM SIGMCOMM, 2007.
[15] R. Kuehl, “Design of Experiments”, Thomson Learning, 2000.
[16] Z. Nasralla, T. E. H. El-Gorashi, M. O.I. Musa, and J. M. H. Elmirghani., “Routing Post-Disaster Traffic Floods in Optical Core Networks”, in Proc. ONDM, 2016.
[17] V. Uceda, M. Rodriguez, J. Ramos, J. Garcia-Dorado, J. Aracil, “Selective capping of packet payloads at multi-Gb/s rates,” IEEE Journal on Selected Areas in Communications, vol. 34, pp. 1807-1818, 2016.
[18] G. Huang, A. Lall, C. Chuah, J. Xu, “Uncovering Global Icebergs in Distributed Streams: Results and Implications,” Journal of Network and Systems Management, vol. 19, pp. 84-110, 2011.
[19] A. P. Vela, A. Via, M. Ruiz, and L. Velasco, “Bringing Data Analytics to the Network Nodes,” in Proc. European Conference on Optical Communications (ECOC), 2016
[20] A. P. Vela, M. Ruiz, and L. Velasco, “Reducing Virtual Network Reconfiguration and Traffic Losses under Multiple Traffic Anomalies”, accepted in Asia Communications and Photonics Conference (ACP), 2016.
[21] G. Chen, R. Lockhart, and M. Stephens, “Box-Cox transformations in linear models: Large sample theory and tests of normality,” Canadian Journal of Statistics, vol. 30, pp. 177-209, 2002.
[22] N. Sambo, F. Cugini, A. Sgambelluri, and P. Castoldi, “Monitoring plane architecture and OAM Handler,” IEEE/OSA Journal of Lightwave Technology, vol. 34, pp. 1939-1945, 2016.