Computer Networks 200 (2021) 108503

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Network anomaly detection based on tensor decomposition ☆

Ananda Streit [a], Gustavo H.A. Santos [a], Rosa M.M. Leão [a,*], Edmundo de Souza e Silva [a], Daniel Menasché [a], Don Towsley [b]

[a] Federal University of Rio de Janeiro, Systems Engineering and Computer Science, Graduate School and Research in Engineering, 21941-914 Rio de Janeiro, RJ, Brazil

[b] University of Massachusetts Amherst, College of Information and Computer Sciences, MA 01003, USA

ARTICLE INFO

Keywords:
Network measurement and analysis
Machine learning for networks
DDoS detection
Tensor decomposition
Quality of Experience (QoE)
Network anomaly detection

ABSTRACT

The problem of detecting anomalies in time series from network measurements has been widely studied and is a topic of fundamental importance. Many anomaly detection methods are based on the inspection of packets collected at the network core routers, with consequent disadvantages in terms of computational cost and privacy. We propose an alternative method in which packet header inspection is not needed. The method is based on the extraction of a normal subspace obtained by the tensor decomposition technique considering the correlation among metrics. In its online version, the proposed approach for tensor decomposition allows efficient tracking of changes in the normal subspace. The flexibility of the method is illustrated by applying it to distinct examples that include supervised and unsupervised anomaly detection. The examples use actual data collected at residential routers.

1. Introduction

The problem of detecting anomalous events in computer networks has been widely studied due to its relevance to network operators. However, these events are in general very hard to identify [1]. The problem is challenging due to the wide variety of anomalies, low frequency of occurrences, and the definition of what is considered "expected behavior" [2].

An application example among the countless existing ones is detecting occasional changes in traffic patterns on a communication channel caused by a distributed denial of service (DDoS) attack. DDoS attacks represent a major threat to proper network operation, wasting resources and creating network outages. For instance, DDoS attacks targeted Amazon Web Services in October 2019 and were able to disrupt different services [3].

Another example was the sudden and significant increase in the amount of network traffic which was observed in several countries when social distance measures were enforced at the beginning of the COVID-19 pandemic [4–6]. In addition, throughout the subsequent months, there has been an unprecedented growth in the overall number of cyberattacks [7]; DDoS attacks reached record numbers and became more disruptive [8–10]. These events caused unusual changes in traffic and performance degradation in network services were reported worldwide [11]. We show that both DDoS attacks (Section 6.5) and network performance degradation events (Section 7.3.3) were observed in our dataset during the pandemic and other periods.

In general, anomaly detection is based on the analysis of packet headers at the core of the network, with potentially high computational cost and possible privacy issues. Our methodology differs from others in that it does not use packet headers. Instead, we use only a small amount of information such as byte and packet counts.

Network measurements are key to identifying network problems [12]. In our work we analyzed data from passive and active measurements collected at one-minute intervals from thousands of home routers of a medium-sized ISP during several months.

The methodology is founded on *tensor decomposition* to detect and diagnose anomalous events using multivariate time-series resulting from the data obtained from our partner ISP. Tensor decomposition is useful to extract normal patterns from the considered metrics, at different time intervals, and to identify latent relationships between them. We also devise a new online tensor decomposition method that efficiently tracks changes in the normal subspace. Our results show the effectiveness of our approach to detect anomalies in two different scenarios used as examples. Nevertheless, we emphasize that the methodology is general and can be employed in other scenarios.

This work shares similarities with [13,14], where a normal subspace is defined by applying PCA and the residuals from the model are used

☆ This work was partially supported by grants from CNPq, CAPES, FAPERJ and by international cooperative grants from MCTIC-FAPESP, NSF_EAGER_1740895/MCTIC-RNP and Army Research Labs (ARL) No. W911NF-17-2-0196.

* Corresponding author.
E-mail address: rosammleao@gmail.com (R.M.M. Leão).

https://doi.org/10.1016/j.comnet.2021.108503
Received 8 February 2021; Received in revised form 27 August 2021; Accepted 18 September 2021
Available online 25 September 2021

to detect network anomalies. In our work, extraction of the normal subspace is performed using the PARAFAC model [15], which naturally allows the decomposition of multidimensional data.

The first application of our approach is to detect intentional anomalies caused by DDoS attacks. We applied the offline and online versions of our approach to the dataset that contains time-series of packets and bit rates collected with non-intrusive measurements performed at home routers. The results indicate that the online version is a viable approach in practice. We also provide examples to show that anomalies can be detected even when ground truth labels are not available. That is, our method is applicable either as a supervised or unsupervised technique.

In our second application, we analyze time-series of packet loss and latency collected at home routers aiming to identify periods when performance is degraded to a sufficient degree that affects user's perception of the quality of service (QoS) provided by the ISP. In this application, we are not interested in identifying the possible events that caused performance to degrade either intentionally or non-intentionally. Instead, our goal is to discover the time intervals where performance was affected to such an extent that they adversely impacted user services. The unsupervised method we devised can be used not only to automatically identify intervals with poor network performance but also to locate regions in the ISP topology that suffered from degraded performance during these intervals. The examples we provide are from a real world scenario and indicate that our method is able to detect performance anomalies without previous knowledge of labels. This is done by correlating our results with real events reported by our partner ISP after concluding our analysis.

Contributions. Key contributions are summarized below:

- *Tensor decomposition to detect network anomalies.* Our framework is based on tensor decomposition (Section 4). We show that the PARAFAC model provides an interpretable and efficient way to extract expected normal behavior, taking into account correlations among different metrics. We illustrate the application in two scenarios using different input metrics.
- *New online tensor decomposition method.* Our method is based on a *tensor window* [16] (Section 5.2). The results show the efficiency and good accuracy of the approach.
- *Supervised and unsupervised anomaly detection.* The proposed framework is applicable both when labels for anomalies are available and when they are not (supervised or unsupervised techniques). In the latter case, we further enhanced the framework with statistical modeling or unsupervised clustering applied to the residuals of the PARAFAC model as a means of interpreting the final results.
- *Use of real data collected at home routers.* We use time-series obtained from real network measurements collected at home routers to evaluate the framework. Our method is capable of detecting different types of anomalies, such as DDoS attacks (Section 6) and network performance issues (Section 7), based on simple metrics and without compromising user privacy.
- *Analysis of network behavior during the COVID-19 pandemic.* One of the examples used data collected during the COVID-19 confinement period. The results show that not only did performance degrade during that period but also the occurrence of DDoS attacks. We have identified both an intentional DDoS attack targeting the ISP (Section 6.5) and a non-intentional degradation in performance (Section 7.3.3).
- *Application to assess user QoE.* As shown in one of the application examples, the basic framework can be used to assess user QoE from the data collected at home routers (Section 7.3.4).

Related work is presented in Section 2. The tensor decomposition technique is discussed in Section 3. Section 4 describes the proposed framework for anomaly detection and we explain how residuals are extracted in both offline and online scenarios in Section 5. The DDoS attack detection example application is presented in Section 6, and Section 7 describes the second application example in which network performance degradation intervals are identified. Section 8 concludes our work.

2. Related work

Most prior work on network anomaly detection is based on packet inspection in the core of the network [2,13,14,17,18], which requires processing private/sensitive information from packet headers, such as traffic volume between source and destination IPs and port number. Recent work also employs packet inspection, but at home routers [19]. Our work relies on lightweight measurements at home routers without requiring packet inspection, providing a simple, efficient and privacy-preserving strategy.

A cyber attack may be thought of as a traffic anomaly. Prior work from our group [20] focus on DDoS lightweight attack detection and solely employs byte and packet counts collected at home routers to detect an attack, similar to our present work. However, in [20] a classifier is trained using simple traffic statistics over a small time window, while we use tensor decomposition. Additionally, we show that the foundation of our approach is applicable to detecting different types of anomalies. PARAFAC produces interpretable models [15] and we are able to infer the normal daily behavior of users which is central for network managing and monitoring tasks [21] and one of the main challenges in anomaly detection [1].

Additional works use subspace extraction methods (like PARAFAC) to detect network anomalies. In one such work, Maruhashi et al. [17] identify suspicious activities in the network, such as port scanning and the spreading of worms, by searching for abnormal sub-graphs from the patterns discovered by PARAFAC. Their method heavily depends on manually choosing patterns deemed interesting. In addition, [17] uses packet inspection, considering a dataset structured as ⟨source IP × destination IP × timestamp or port number⟩. This is a typical structure used for network analysis with tensor models based on packet inspection. Instead, our work does not extract data from packet headers and considers tensors in the form: ⟨residential router × network metric × timestamp⟩.

The works of Lakhina et al. [13,14] apply PCA to define a normal subspace. In [14] anomalies that span multiple traffic features (metrics) are detected, similar to our work. However, PCA is a matrix-based model and, unlike tensor-based models like PARAFAC, it requires that multidimensional data is unfolded [22] into a single large matrix before being applied. The PARAFAC model, on the other hand, preserves correlations between different metrics in multidimensional data and, as such, it is more robust to noise. Furthermore, under mild conditions PARAFAC solution is unique, a very useful property. This differs from PCA [15] (PCA's solution is not unique).

Xie et al. [18] proposes an anomaly detection method using a modified PARAFAC model that separates normal from sparse outlier data during the optimization process. Unlike our approach, the method of [18] requires a hyperparameter which is an upper limit to the number of outliers. Subsequent work from the same authors followed similar path [23]. Kasai et al. [24] also proposes a sparse tensor to account for abnormal flows. However, all of the above works ignore the interpretability of the model and evaluate the method using only artificially generated attack data taken from arbitrary probability distributions. In contrast, we consider: (i) attack traffic generated using real malware and an actual attack traffic directed to our partner ISP; (ii) actual network performance degradation events.

We propose an online tensor decomposition approach based on the sliding window method of Sun et al. [16]. Our solution incurs a small computational cost appropriate to online applications while maintaining good performance. Kasai et al. [24] also considers a tensor-based online algorithm, makes use of the sliding window concept, and modifies PARAFAC to deal with time and space complexities required for an online approach. The work of [23] is another example targeted to online usage where an incremental tensor factorization algorithm is proposed in order to reduce storage and computational costs. The literature includes additional methods for online applications that propose

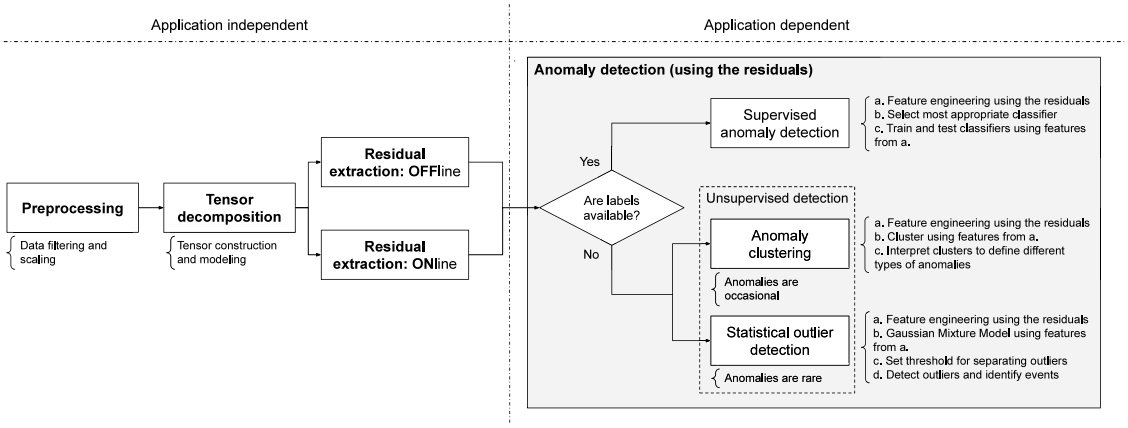


Fig. 1. Framework for anomaly detection based on tensor decomposition.

modifications to the standard PARAFAC but do not use the concept of windows [25,26].

We devise an online solution which is simpler than the above methods and avoids the use of hyperparameters such as the upper limit on the number of outliers that is necessary in [23]. In our approach, we modify the standard PARAFAC decomposition to recalculate only part of the model when the window slides, instead of updating it entirely, as done in Kasai et al. [24]. In our method, only a fraction of the entries of the time mode factor matrix are recomputed in an online fashion (see Section 5.2.2 for details).

We extended our previous work in [27] in several ways. We developed a new unsupervised approach for traffic outlier detection. The approach is assessed in detecting a real DDoS attack. We modified our previous anomaly clustering technique training a Gaussian Mixture Model using residuals of the PARAFAC model and showed that it was able to detect additional real network events including: (i) the day of improvement in the QoS resulting from interventions performed by the ISP on its network and (ii) network service degradation caused by two distinct events: a DDoS attack and the increase in residential traffic when the COVID-19 confinement was issued. In addition, we were able to correlate the results of the residual clustering using our dataset with the quality of experience (QoE) observed by ISP clients as extracted from the ISP customer ticket database. Finally, we compare our residual-based model with an approach that makes immediate use of the PARAFAC model loadings and show that the former was more efficient in detecting anomalies in our dataset.

3. Background: Tensor decomposition

In this section we briefly present tensor decomposition and describe our notation. For details we refer the reader to [22]. A tensor is a multidimensional matrix denoted by $\mathcal{X}$. We usually refer to the dimensions of $\mathcal{X}$ as *modes*. A third-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ can be represented by a sum of three-way outer products [22] as follows,

$$\mathcal{X} = \mathcal{M} + \mathcal{E}, \quad \mathbf{a}_r \in \mathbb{R}^I, \mathbf{b}_r \in \mathbb{R}^J, \mathbf{c}_r \in \mathbb{R}^K, \tag{1}$$

$$\mathcal{M}_{i,j,k} = \sum_{r=1}^{R} \mathbf{a}_{r,i}\mathbf{b}_{r,j}\mathbf{c}_{r,k}, \tag{2}$$

where $\mathcal{E}$ is the residual tensor and R is the number of factors. The *factor matrices* (or loadings) define model $\mathcal{M}$: $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_R] \in \mathbb{R}^{I\times R}$, $B = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_R] \in \mathbb{R}^{J\times R}$, $C = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_R] \in \mathbb{R}^{K\times R}$. Following standard notation, we let $\mathbf{a}_r = A_{:,r}$, for $1 \le r \le R$, and $\mathbf{a}^{(i)} = A_{i,:}$, for $1 \le i \le I$.

The PARAFAC decomposition is obtained by minimizing the sum of squares of the residuals, i.e., the difference between $\mathcal{X}$ and $\mathcal{M}$. Such a difference is a nonconvex function; however, if we fix two of the factor matrices, the problem is reduced to a linear least squares regression for the third matrix. This is the basis of the *Alternating Least Squares* (ALS) procedure [15]. ALS estimates the factor matrices one at a time, keeping the others fixed. The process iterates until a convergence criterion is satisfied or there are no changes in the estimates.

In this work we use the method of *Split-Half Validation* (SV) [28] in combination with *Tucker Congruence Coefficient* (TCC) [29] to estimate R and evaluate whether the solution is unique and generalizable.

4. Framework

The proposed framework is sketched in Fig. 1. The first three steps in the figure are executed regardless of whether labels are available or not. For these two cases, measurement data is preprocessed, tensor decomposition is applied, and residuals are extracted from the model. Note that the key to detecting anomalies is the analysis of PARAFAC residuals after they are calculated. The last step executed depends on the availability of labels. When labels for anomalies can be obtained, a supervised classifier is used to find the residual patterns associated with the anomalies under investigation. On the other hand, when labels are absent, two cases are considered, depending on whether anomalies are expected to be rare or not. In the first case a statistical outlier detection method is used and we classify as anomalies the residual samples that deviate from the expected residual distribution. When anomalies are anticipated not to be rare, unsupervised clustering methods are employed and the resulting clusters are used as a means of interpreting the anomalies. The steps in Fig. 1 are detailed below.

Preprocessing: In this step we perform data transformations needed to apply tensor decomposition, such as data scaling and filtering.

Tensor decomposition: Tensor decomposition is applied to extract the normal subspace. We use PARAFAC due to its properties (robustness and unique solution) and the ability to support multivariate data [15].

Residual extraction: Residuals are extracted from the PARAFAC model. Considering that anomalies are not well modeled by the normal subspace, it would be possible to separate the normal behavior of the data from the anomalies through residual analysis. In this work, we account for both offline and online residual extraction. In the first case, the residuals are extracted from a model parameterized using a reference dataset (Section 5.1). The online procedure employs a sliding time window that moves at regularly spaced intervals and at a speed proportional to the rate at which new data is received. The model is updated to include the samples in the current window and then new residuals are extracted. (Section 5.2).

Anomaly detection: The final steps depend on the availability of labels. When the dataset contains labeled anomalies, supervised classification is used. The application described in Section 6 provides an example that uses a traffic dataset that contains DDoS attack traffic

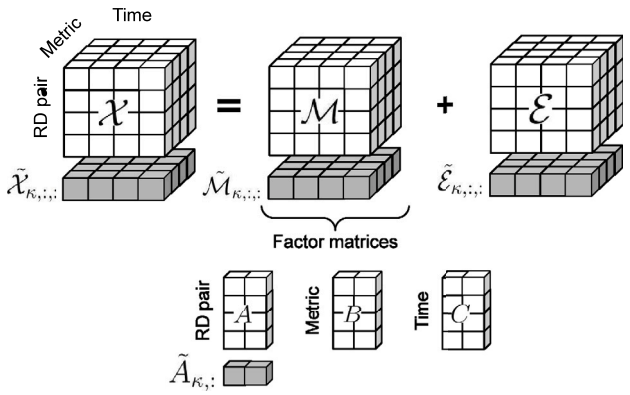


Fig. 2. Offline residual extraction.

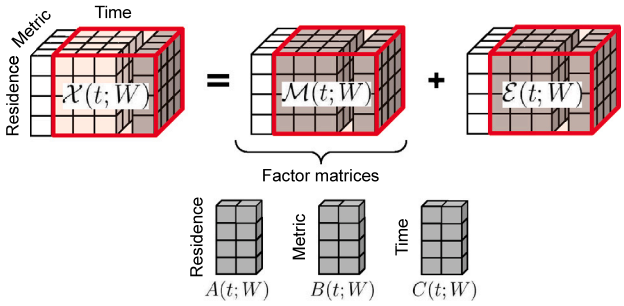


Fig. 3. Online residual extraction.

identified as such. We are also interested in applications where labels for anomalies are unavailable or hard to obtain. When labels are unknown and anomalies are believed rare, we use a statistical model to estimate the expected behavior of the residuals. Our method identifies as outliers the residuals that deviate from the expected behavior. Section 6.5 shows an example of this case. When anomalies are not expected to be rare, we use residual clustering in order to interpret the clustering results and categorize anomalies. We apply residual clustering to the example in Section 7.

5. Residual extraction

Our anomaly detection technique is based on the analysis of PARAFAC residuals [15]. The model is built from time-series of residential traffic (first application) and then from loss and latency measurements (second application) sampled continuously at every minute in both cases. Anomalies are detected from deviations of the tensor decomposition model.

5.1. Offline residual extraction

Residential traffic data exhibits strong daily patterns over time and thus we chose to work with daily residential traffic time-series. We denote each series as a *Residence–Day pair*, or *RD pair* for short.

Fig. 2 describes the offline residual extraction method. Let I denote the number of RD pairs in our dataset. We built a three-way tensor with modes: RD-pair, the measure of interest and the instant of time the measurement was taken. Modes are indexed by i, j and k, respectively. Let $\mathcal{X}_{i,:,:}$ be the ith horizontal slice of tensor $\mathcal{X}$, i.e., $\mathcal{X}_{i,:,:}$ is a two-dimensional matrix obtained by fixing the RD pair mode at value i [22]. Then, for each RD_i with sampled values $\mathcal{X}_{i,:,:} \in \mathbb{R}^{1\times J\times K}$, we obtain a model slice $\mathcal{M}_{i,:,:} \in \mathbb{R}^{1\times J\times K}$ using the PARAFAC ALS procedure. Residuals are the difference between the model estimates and the input dataset values $\mathcal{E}_{i,:,:} = \mathcal{X}_{i,:,:} - \mathcal{M}_{i,:,:}$, where $\mathcal{E}_{i,:,:} \in \mathbb{R}^{1\times J\times K}$. Measurements are performed at each residence every minute and the corresponding time-series are divided into 24-hour periods. Then, $K = 1440$.

After parameterizing the model $\mathcal{M}$, we obtain the residuals corresponding to the measures of new RD pairs not previously used to construct $\mathcal{M}$. Let $\tilde{\mathcal{X}}_{\kappa,:,:}$ denote the measures corresponding to a new RD_κ. We use factor matrices B and C from the previously trained model $\mathcal{M}$ (Eq. (2)) and the new values $\tilde{\mathcal{X}}_{\kappa,:,:}$ to obtain vector $\tilde{\mathbf{a}}^{(\kappa)} \in \mathbb{R}^{1\times R}$. Factor matrices $\tilde{A}$, B and C produce model $\tilde{\mathcal{M}}_{\kappa,:,:}$, with corresponding error $\tilde{\mathcal{E}}_{\kappa,:,:}$, where $\tilde{A}_{\kappa,:} = \tilde{\mathbf{a}}^{(\kappa)}$. Vector $\tilde{\mathbf{a}}^{(\kappa)}$ is chosen to minimize the quadratic error between model estimates and measurements. Let $\tilde{\mathcal{X}}_{\kappa,:,:(1)}$ be the matrix unfolding of tensor $\tilde{\mathcal{X}}_{\kappa,:,:}$ in its first mode [22], where $\tilde{\mathcal{X}}_{\kappa,:,:(1)} \in \mathbb{R}^{1\times JK}$. Then,

$$\tilde{\mathcal{M}}_{\kappa,:,:} = \tilde{\mathcal{X}}_{\kappa,:,:} - \tilde{\mathcal{E}}_{\kappa,:,:}$$

$$\Rightarrow \tilde{\mathbf{a}}^{(\kappa)}(C \odot B)^T = \tilde{\mathcal{X}}_{\kappa,:,:(1)} - \tilde{\mathcal{E}}_{\kappa,:,:(1)}$$

$$\Rightarrow \tilde{\mathbf{a}}^{(\kappa)} = \tilde{\mathcal{X}}_{\kappa,:,:(1)}((C \odot B)^T)^\dagger, \tag{3}$$

where $C \odot B$ denotes the Khatri–Rao product [22] between matrices C and B and $M^\dagger$ denotes the Moore–Penrose pseudo-inverse of matrix M [22]. Note that both $(C \odot B) \in \mathbb{R}^{JK\times R}$ and $((C \odot B)^T)^\dagger \in \mathbb{R}^{JK\times R}$. As vector $\tilde{\mathbf{a}}^{(\kappa)}$ minimizes the quadratic error, the corresponding error $\tilde{\mathcal{E}}_{\kappa,:,:(1)}$ is orthogonal to $((C \odot B)^T)^\dagger$ which implies (3). Thus, the residuals of RD_κ are obtained by $\tilde{\mathcal{E}}_{\kappa,:,:} = \tilde{\mathcal{X}}_{\kappa,:,:} - \tilde{\mathcal{M}}_{\kappa,:,:}$, where model $\tilde{\mathcal{M}}_{\kappa,:,:} \in \mathbb{R}^{1\times J\times K}$ contains the new factor vector $\tilde{\mathbf{a}}^{(\kappa)}$.

For the offline applications that uses daily time-series, one must wait for a 24 h period to obtain new time-series for an entire day and for all users (i.e., the measurements corresponding to RD pairs) and residuals are computed when the new daily batch of series is available. In addition, since statistics of network metrics may considerably change after a long period of time it is necessary to check, from time to time, if $\mathcal{M}$ is still a good model and, if needed, to retrain model $\mathcal{M}$. (The Split-Half validation [28] is used for the check.) In the examples we consider, no model retraining was needed even after several months.

5.2. Online residual extraction

The online method searches for anomalies in data by continuously recomputing the model using PARAFAC. Since the data used on our application is collected every minute from all residences that participated in the measurement campaign, it is natural to use one minute time slots and to process the data accordingly, to detect anomalies as new data arrives. The online decomposition considers **residences** instead of RD pairs as one of the tensor modes, and obtains a three-way tensor with residence (mode A), the metrics of interest (mode B), and time (mode C). At every minute, using the new measurement values that arrive from residences, the model is updated and residuals are extracted.

Fig. 3 shows how residuals are obtained. At every time slot t, new measurement data arrives. This new set constitutes a matrix with dimensions $I \times J$, where I is the number of residences and J is the number of different measures of interest (for instance, download and upload byte counts). This matrix is represented by the vertical tensor slice in Fig. 3. The window W (shown in red in the figure) slides to incorporate the slice with new samples to tensor $\mathcal{X}$. Finally new loadings $A(t;W)$, $B(t;W)$ and $C(t;W)$ are calculated as indicated in the following sections and the process repeats for the subsequent slots. (For conciseness of notation, we drop the parameter W from the factor matrices.)

We propose two different online residual extraction schemes: Full Window Optimization (FWO) which is based on [16] (Section 5.2.1), and Partial Window Optimization (PWO), a simpler and more efficient FWO variant (Section 5.2.2).

Fig. 4 illustrates the main difference between FWO and PWO. In the figure, each rectangle represents the metrics of one residence for a time slot. In the FWO method all window information (red/solid rectangles) is used to compute factor matrices A, B and C. On the other

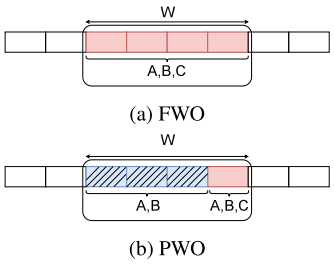


Fig. 4. Online tensor decomposition approaches ($W = 4$ time slots). Red time slots (solid rectangles) are used to compute factor matrices A, B and C. Blue time slots (hatched rectangles) are used to compute factor matrices A and B.

hand, in the PWO method, factor matrices A, B are estimated with all window information, as in FWO, but matrix C is updated using only the information from the last time slot. Note that both schemes allow expansion of modes A and B in case new residences are added or new metrics of interest are collected, respectively.

5.2.1. Full window optimization (FWO)

A simple approach to online tensor decomposition is based on a *tensor window* [16] $\mathcal{X}(t;W) \in \mathbb{R}^{I\times J\times W}$ over the time mode (mode C), where W refers to the window size. At every time slot t the window slides and a new tensor is formed by merging the preceding $W-1$ slices $\{\mathcal{X}_{:,:,t-W+1},\dots,\mathcal{X}_{:,:,t-1}\}$ with data $\mathcal{X}_{:,:,t} \in \mathbb{R}^{I\times J\times 1}$ from slot t. Since our traffic data exhibit daily patterns and we consider one minute slots, we choose $W = 1440$ minutes.

To obtain models in FWO we use the same optimization method applied in the offline scenario for each sliding window. Briefly, for each window we compute a PARAFAC model $\mathcal{M}(t;W) \in \mathbb{R}^{I\times J\times W}$ using $\mathcal{X}(t;W) \in \mathbb{R}^{I\times J\times W}$ as input for the PARAFAC ALS algorithm. Residuals are computed for a window of data $\mathcal{E}(t;W) = \mathcal{X}(t;W) - \mathcal{M}(t;W)$. Since we are interested on the most recent sample, we focus on the residuals of the last minute t, $\mathcal{E}_{:,:,t} \in \mathbb{R}^{I\times J\times 1}$.

FWO requires the computation of a new PARAFAC model at every window. As such, it may not be suitable for online applications, often requiring a large and variable number of iterations [25]. We evaluated this method (Section 6) and the results indicate it was computationally expensive, enough to preclude its use for our online application (see Fig. 8). This motivates the variation we propose below aimed at reducing the computational cost while providing good performance.

5.2.2. Partial window optimization (PWO)

In order to reduce the run time of the online residual extraction we modify FWO. Consider the factor matrix related to the time mode $C(t) \in \mathbb{R}^{W\times R}$ used to model the tensor window $\mathcal{X}(t;W)$. To obtain $C(t)$ we keep the previous $W-1$ loadings calculated from $t-W+1$ to $t-1$, $\{\mathbf{c}(t-W+1),\dots,\mathbf{c}(t-1)\}$ and compute $\mathbf{c}(t)$, the time mode loadings of the last sample. The other factor matrices $A(t)$ and $B(t)$ are fully recomputed based on the tensor window $\mathcal{X}(t;W)$. We use $W = 1440$, as in FWO.

Let $\mathcal{X}_{:,:,t}$ denote the measures at time t. The model at t is obtained by updating matrices $A(t)$, $B(t)$ and vector $\mathbf{c}(t)$ (see Fig. 4(b)) alternately and iteratively, until a convergence criterion is satisfied (Algorithm 1). As in the ALS algorithm, matrices $A(t)$ and $B(t)$ and vector $\mathbf{c}(t)$ are calculated by minimizing the quadratic error between model estimates and measurements. The sequence of updates is given by lines 4–7 in Algorithm 1.

In Algorithm 1, $\mathcal{X}_{(1)}$ and $\mathcal{X}_{(2)}$ are the tensor unfoldings of $\mathcal{X}$ in its first and second modes, respectively, $\mathcal{X}_{(1)} \in \mathbb{R}^{I\times JK}$, $\mathcal{X}_{(2)} \in \mathbb{R}^{J\times IK}$. Note that $\mathbf{c}(t) \in \mathbb{R}^{1\times R}$, $A(t) \in \mathbb{R}^{I\times R}$ and $B(t) \in \mathbb{R}^{J\times R}$. The factor matrices of model $\mathcal{M}(t;W)$ are initialized with the model estimates $\mathcal{M}(t-1;W)$ obtained for the previous window. The residuals at t are obtained by:

$$\mathcal{E}(t;W)_{:,:,t} = \mathcal{X}(t;W)_{:,:,t} - \mathcal{M}(t;W)_{:,:,t}.$$

6. Application I: DDoS attack detection

The first example application of the anomaly detection framework uses actual residential traffic to assess the efficacy of detecting DDoS attacks (see Fig. 1). We first focus on the setting where we have labeled data, and describe in detail every step of our approach. Then we consider a setting where the data is unlabeled and show that an anomaly we detected using our method was also identified by the partner ISP as an external attack on routers in the ISP's network.

6.1. Labeled dataset construction and preprocessing

Our original dataset consists of upload and download byte and packet counts from nearly a thousand residences, collected every minute organized into daily multivariate time-series. (Time is divided into minute-slots.) Recall that the time-series are anonymously identified with a residence and with a specific day, referred to as a RD pair, and constitute the input to the tensor decomposition method. Data was collected during five weeks, 19-August-2019 to 22-September-2019, and from 812 residences producing a total of 18,494 time-series.

In order to obtain a labeled dataset with DDoS attack traffic, we followed the methodology described in [20]. Briefly, three steps are executed: (a) residences are randomly chosen to form a botnet; (b) attacks initiate at randomly chosen instants of time; (c) the total traffic generated during the attack is the actual regular residential traffic during that period added to the attack traffic generated from a real malware.

The attack traffic we used was generated and logged by [20] after running, on a Raspberry-Pi in our laboratory, the actual malware code from two common IoT botnets: Mirai and BASHLITE. Table 1 lists the different types of attacks considered in our work. Since, according to [30], the majority of attacks lasts for a few minutes, the duration of each attack was chosen to follow a Gaussian distribution with mean $\mu = 120$ seconds and standard deviation $\sigma = 10$ seconds.

A fraction $q = 0.05$ of all residences are randomly selected to participate in the synchronized botnet, before attacks are launched. The choice for the value of q was based on attack statistics reported in [31].

There are more than 50,000 minute-slots in our dataset. Attacks start in slots chosen uniformly at random from the total, averaging one attack per day. (Similar results were obtained when other attack rates were considered.) In addition, the specific type of attack for a slot is randomly selected from the list in Table 1.

The five-week labeled dataset with DDoS attacks was partitioned into three non-overlapping subsets: $\mathrm{Tr}_1, \mathrm{Tr}_2$ and Te, respectively used to: extract the normal subspace; train the classifier to detect traffic anomalies (supervised training) and; test the performance of the classifier (see Fig. 5). The first week (dataset Tr_1) is used to obtain the PARAFAC model. The subsequent four weeks are left for extracting the residuals used for training and testing the classifier (Tr_2 and Te, respectively). We did not apply standard k-fold cross validation to train and test the classifier, since traffic time-series possess temporal dependencies that would be affected by standard cross validation. Instead, we used forward-chaining cross-validation for training the classifier [32].

1 $A(t) \leftarrow A(t-1)$, $B(t) \leftarrow B(t-1)$
2 $C(t) \leftarrow [\mathbf{c}(t-W+1)^T,\dots,\mathbf{c}(t-1)^T,\mathbf{c}(t-W)^T]^T$
3 **while** *not converged* **do**
4 $\quad \mathbf{c}(t) \leftarrow \mathcal{X}_{:,:,t(3)}((B(t)\odot A(t))^T)^\dagger$
5 $\quad C(t)_{W,:} \leftarrow \mathbf{c}(t)$
6 $\quad A(t) \leftarrow \mathcal{X}_{(1)}((C(t)\odot B(t))^T)^\dagger$
7 $\quad B(t) \leftarrow \mathcal{X}_{(2)}((C(t)\odot A(t))^T)^\dagger$
8 **end**
9 **return** $A(t), B(t), \mathbf{c}(t)$

Algorithm 1: PWO Online algorithm

Table 1
Types of DDoS attacks evaluated.

Malware	Attack type (payload size)
Mirai	UDP flood (1400B)
Mirai	TCP SYN flood (0B)
Mirai	TCP ACK flood (0B)
Mirai	UDP PLAIN flood (1400B)
BASHLITE	UDP flood (1400B)
BASHLITE	TCP SYN flood (0B)
BASHLITE	TCP ACK flood (0B)

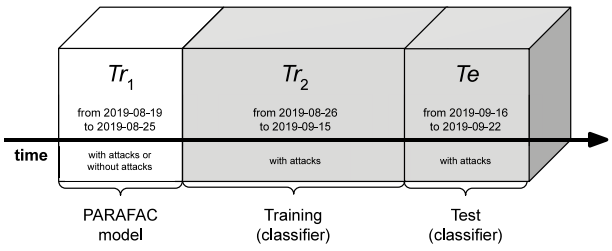


Fig. 5. Application I dataset.

Briefly, Tr_2 begins with a week of data and then expands at each step of the procedure to include more data, while Te contains the data for the week that immediately follows Tr_2. Fig. 5 shows the data intervals for the last "fold".

Remarks: The number of residences in the original dataset vary slightly from one day to the other because not all home routers keep collecting data continuously over the entire four week period. In the last fold, Tr_2 has sightly over one fourth of the RD-pairs in the four week period (76%) while Te has 24% of the RDs. We applied log-transformation to the dataset values and then Min-Max normalization.

In order to extract the normal subspace, intuitively one should use a dataset free of attacks, since anomalies can contaminate the normal subspace. Ringberg et al. [33] showed that sufficiently large anomalies can inadvertently pollute the normal PCA subspace, resulting in increased false positive rates. Therefore, detection using a model trained from a dataset that contains anomalies is more difficult than when training is done on a dataset known to be anomaly free.

However, it is not feasible to find out whether any real and unlabeled traffic dataset contains anomalies, in particular malicious anomalies [14]. Therefore, to stress our model, we use the Tr_1 dataset that contains attacks to extract the normal subspace. For comparison purposes, we also computed an additional PARAFAC model using the original dataset in which attacks were not added to the measured residential traffic. In the latter case, no significant accuracy gain in the testing phase was observed. In the remainder of this section, we present our results using Tr_1 (with attacks included) to emphasize the robustness of the procedure in realistic scenarios.

6.2. PARAFAC model

6.2.1. Tensor decomposition

Different tensor structures are used for offline and online scenarios. In the first scenario, our tensor is composed of three modes: (RD × traffic metric × minute). We obtain model $\mathcal{M} \in \mathbb{R}^{3412\times4\times1440}$ from the training set Tr_1 containing 3412 RDs. The application of *Split-Half Validation* validates up to $R = 6$ factors. (Refer to Section 3. Except as otherwise noted, we use $R = 6$.) On the other hand, the modes for the online approach are (residence × traffic metric × minute), and models $\mathcal{M}(t; W) \in \mathbb{R}^{812\times4\times1440}$ are obtained for a window of size $W = 1440$. Recall that the dataset contains information from 812 residential routers and only four metrics, bit and packet counts of upload and download traffic collected at every minute are used in our models.

We analyze the factors obtained in the offline scenario aiming to interpret the model. Fig. 6 shows the results of (column-wise) Khatri–Rao product $B \odot C$, for each of the six factors and the four measurement loadings, where we recall that C is the time mode and mode B is associated with download and upload bit and packet rate measurements. The PARAFAC model reveals that there is a factor which is nearly constant throughout the day. We chose to call this factor the *Base Factor*. The remaining factors can be associated with high network usage during different periods of the day as shown in the figure using different colors and letters. For each factor, when we compare the download values in the figure with the corresponding upload values, we note that there is a significant difference for the loadings corresponding to bits, while there is almost no difference for the loadings corresponding to packets. This is expected, since TCP should be the predominant protocol and download packets are acknowledged in the upward direction, but download packets carry the load.

6.2.2. Residual extraction

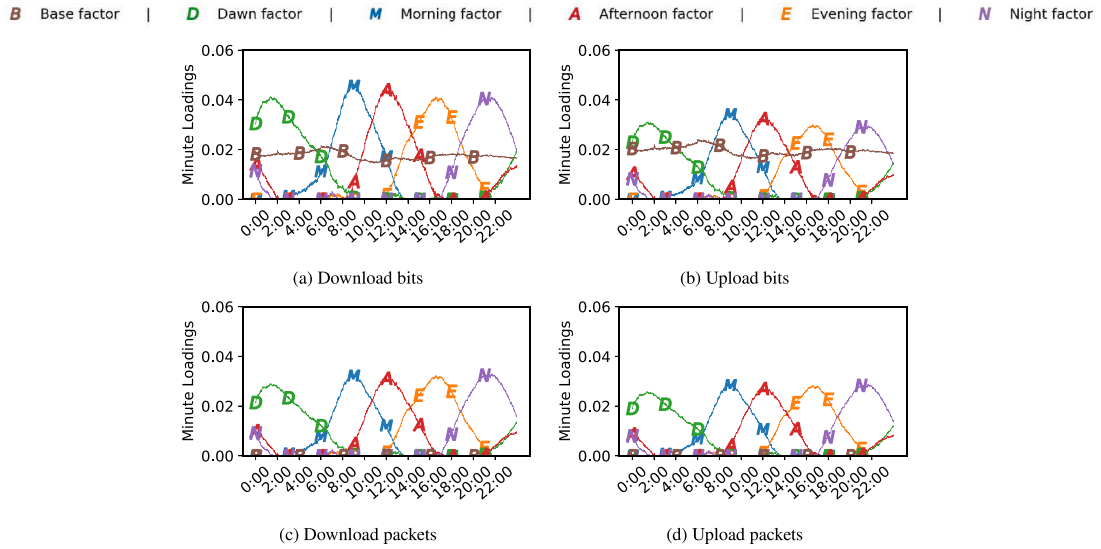
We extract residuals for sets Tr_2 and Te using the residual extraction techniques described in Section 5. These residuals are extracted from all traffic metrics for each minute of each RD/residence (offline/online) and are used as input to the classifier. Since the time correlation between upload and download traffic is an important feature to detect attacks [20], we make use of two additional features, bringing the total to the following six features: (i) *download bit residuals*, (ii) *upload bit residuals*, (iii) *download packet residuals*, (iv) *upload packet residuals*, (v) *difference between upload and download bit residuals* and (vi) *difference between upload and download packet residuals*. Figs. 7(a) and 7(b) show histograms of features (iv) and (vi) above, obtained by the offline method. These are the two features that are best ranked by the classifier. The red/crosshatched (green/unhatched) bars of the histogram represent the feature values for minutes and residences that either do or do not contain attacks, respectively. Note that the red/crosshatched and green/unhatched histograms differ significantly and, intuitively, this observation provides an indication that traffic containing attacks can be distinguished from normal traffic using the PARAFAC residuals.

For the online methods, the computation time needed to compute the PARAFAC residuals should be assessed, as the residuals must be obtained at every minute and hence they must be computed in less than one minute. PARAFAC validates for models between two to six factors. We use $R = 2$ for the online methods to reduce extraction time even though the computational cost is not significantly affected in this range of R values. Fig. 8 shows the number of iterations and run times for the PWO and FWO online methods, for each minute during a specific day which included an attack. The red dotted line indicates when a DDoS attack occurred in the chosen day. We compute both models using a PowerEdge R230 server with a Intel Xeon E3-1220 v6 of 3Ghz with 4 cores and 64 GB of RAM. FWO recomputes all loadings of factor matrix $C(t)$ (time) at every minute, that is, whenever the window slides. From the figure we notice that the computation time required to compute $\mathcal{M}(t; W)$ using FWO varies depending on the data values $\mathcal{X}(t; W)$ and, in this example, it is too high for online use. On the other hand, PWO consistently requires much less computation time than FWO and is the method of choice.

6.3. Supervised anomaly detection

6.3.1. Performance

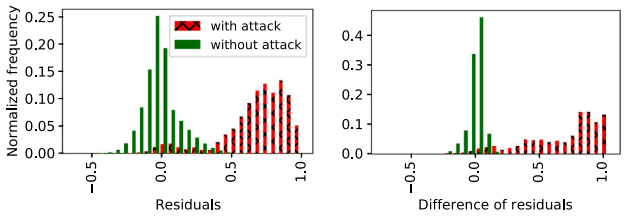
In this example, a classifier must be trained to detect attacks leveraging features extracted using PARAFAC. We compare results from five different classifiers, *Logistic Regression*, *Decision Tree*, *Random Forest*, *Gaussian Naive Bayes* and *Multi-layer Perceptron*. We also use PCA as an alternative method to PARAFAC for comparison purposes, after unfolding the three dimensional data to a two-dimensional array. For all five classifiers, PARAFAC outperforms PCA, and classification using



(a) Download bits (b) Upload bits

(c) Download packets (d) Upload packets

Fig. 6. Factors obtained by PARAFAC.



(a) Upload packet residuals (b) Difference between upload and download packet residuals

Fig. 7. Histograms of residuals for minutes and residences with and without attacks.

Table 2
Performance measures with random forest classifier.

Model	Precision	Detection accuracy	False positive rate
PARAFAC (Offline)	0.9893	0.9618	9.77E−07
PCA	0.9725	0.949	2.57E−06
PWO (Online)	0.9672	0.9401	2.89E−06
Enhanced PWO (Online)	0.9864	0.9908	1.14E−06

Table 3
Feature importance of random forest.

Residual Feature	Gini index
Difference between up and down packet residuals	0.5006
Up packet residuals	0.1601
Difference between up and down bit residuals	0.1445
Down packet residuals	0.1024
Down bit residuals	0.0496
Up bit residuals	0.0427

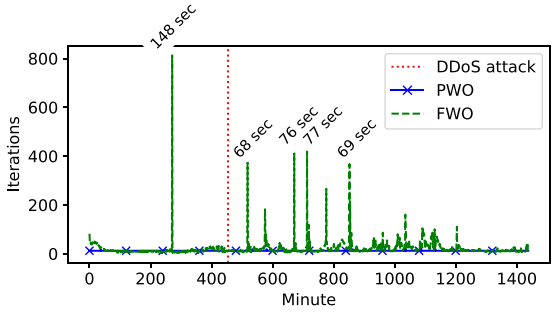


Fig. 8. Number of iterations for FWO and PWO online methods.

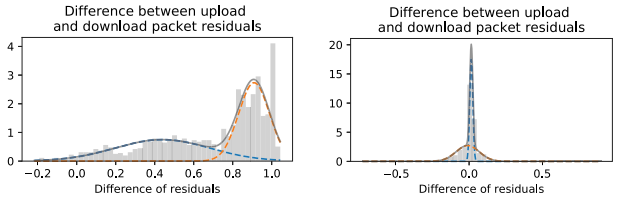
Random Forest produced the best results both for PARAFAC based methods (online and offline) and PCA.

Table 2 presents the results. (In this table two factors are used for comparison between the online and offline algorithms, but the values do not vary much with R.) Precision and false positive rate have the usual definition from the literature. We call *Detection Accuracy* the ratio between the number of attacks that are correctly detected and the total number of attacks in dataset Te. Note that an attack can last for one or more time slots and we consider an attack to be correctly detected when the classifier identifies at least one of the slots with attack traffic. The table includes an additional method, referred to as *Enhanced PWO*. Briefly, the only difference between PWO and Enhanced PWO is that the latter includes two additional features for classification, as described in Section 6.3.2.

From Table 2, the PARAFAC-based methods perform better than PCA for almost all metrics, although the difference if not significant in this dataset. We also applied the methodology to a different dataset with traffic collected during 2019 (full results omitted for the sake of conciseness) and, in that case, while PARAFAC consistently produced good results as the number of factors varies, the precision and accuracy of PCA are significantly affected as the number of components varies. This is consistent with results from the literature (e.g. [33]) that concludes that PCA is sensitive to its parameters. In addition to robustness to the number of parameters, PARAFAC results are interpretable (Section 6.2.1) and its solution is unique.

It is important to highlight the False Positive Rate (FPR) metrics. From the FPR one can immediately obtain the mean number of false alarms per day (MNFA). The MNFA for the offline PARAFAC-based method is 1.1 considering all 812 residences. The online Enhanced PWO has an MNFA comparable to that of offline PARAFAC. Section 6.4.2 shows how the MNFA and other performance metrics can be improved by taking into account spatio-temporal correlations.

For the online algorithms, an important performance metric is the expected time to detect an attack, as a fast detection time is essential for adopting countermeasures and mitigating the impact of an attack. In the case of dataset Te and Enhanced PWO, the expected detection time was one minute in 89.19% of the detected attacks, while all the attack detections occur within two minutes.



(a) GMM trained using residuals of minutes and residences with attacks (b) GMM trained using residuals of minutes and residences without attacks

Fig. 9. One feature of the GMMs trained using residuals of minutes and residences with and without attacks.

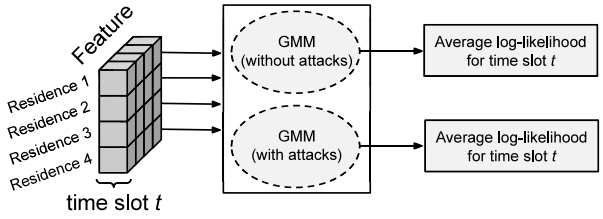


Fig. 10. Average log-likelihood estimation for each time slot.

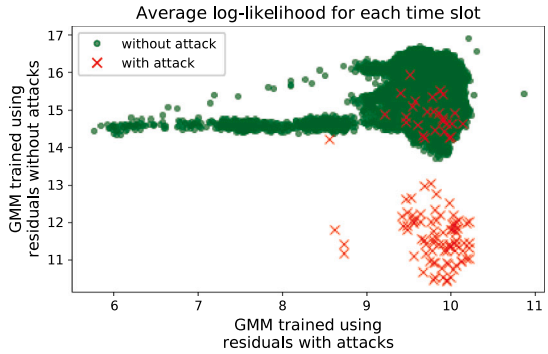
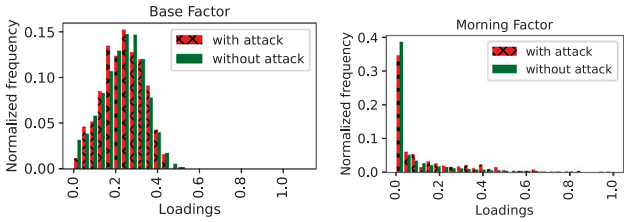


Fig. 11. Average log-likelihood for each time slot computed for GMMs with and without attacks.



(a) Loadings for base factor (b) Loadings for morning factor

Fig. 12. Histograms of loadings for minutes and residences with and without attacks.

6.3.2. *Feature importance and enhanced PWO*

To evaluate the relevance of the six residual features obtained from PARAFAC, we use the Gini index-based importance metric from the *Random Forest* classifier (Table 3). The most important residual feature is the *difference between upload and download packet residuals* (Gini 0.5006) followed by *upload packet residuals* (Gini 0.1601) and *difference between upload and download bit residuals* (Gini 0.1445). The two most important features are based on packet residuals, so we evaluate the classifier using only packet-based features and compare the results with those including all six features. Using PARAFAC with two factors, results show that the average number of false alarms per day triples when only packet-based features are used, while Precision decreases from 0.9618 to 0.9581. (Similar conclusions hold when we increase the number of factors.)

The histograms shown in Fig. 7 suggest that the probability distributions of the residual features can be approximated by a Gaussian Mixture Model (GMM). Therefore, we fit two GMMs with two components each and six dimensions associated with the six features in Table 3. Each of the two GMMs are trained using residual features associated to the minutes for residences that do and do not contain attacks, respectively, as samples. Figs. 9(a) and 9(b) illustrate both Gaussian mixtures for the feature with the highest Gini index (Table 3): *difference between upload and download packet residuals*. Note that the two GMMs are quite different. Based on this observation, we chose to use two additional features for the online PWO classifier.

Consider a set of features sampled for a residence r at minute-slot t. Let $\mathcal{L}_r^a(t)$ and $\mathcal{L}_r^w(t)$ be the log-likelihood of that sample given that it is generated by the GMM previously trained with samples that do and do not contain attacks, respectively. Calculate $\mathbb{E}^{(r)}[\mathcal{L}_r^a(t)]$ (respectively, $\mathbb{E}^{(r)}[\mathcal{L}_r^w(t)]$) defined as the mean value of $\mathcal{L}_r^a(t)$ (respectively, $\mathcal{L}_r^w(t)$) averaged over all residences at slot t. Fig. 10 illustrates how these two new features are obtained.

Fig. 11 shows the values of $\mathbb{E}^{(r)}[\mathcal{L}_r^a(t)]$ and $\mathbb{E}^{(r)}[\mathcal{L}_r^w(t)]$ computed for both datasets Tr_2 and Te. Each point on the scatterplot represents a time slot (refer also to Fig. 10). Red crosses corresponds to time slots in which at least one residence contains attack traffic during that minute, while green dots correspond to slots in which no residence contains attack traffic during that minute. Note the clear separation between red crosses and green dots. Therefore, a new classifier is trained with all the six previously considered features and the two expected values $\mathbb{E}^{(r)}[\mathcal{L}_r^a(t)]$ and $\mathbb{E}^{(r)}[\mathcal{L}_r^w(t)]$ for each t, totaling eight features. The results reported in the last line of Table 2 indicate that these two additional features increase the classifier performance.

6.4. *Additional remarks*

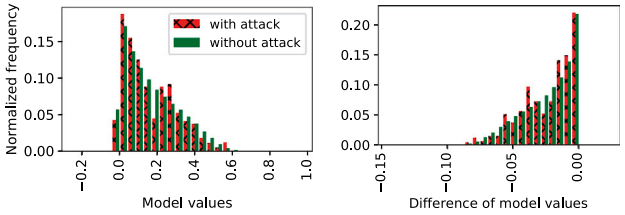
6.4.1. *Loadings versus residuals for detection*

Previous work in the literature consider the detection of patterns using PARAFAC loadings instead of residuals [17,34]. Inspired by those works, we investigate whether it is possible to detect DDoS attacks using the model loadings.

Fig. 12 plots the histograms of the loadings obtained from the Khatri–Rao product of RD mode A and time mode C ($A \odot C$) for the Base factor (Fig. 12(a)) and Morning factor (Fig. 12(b)). The red/hatched and green/unhatched bars in the figures are associated to residences that do and do not contain attack traffic in a minute slot, respectively. It should be noted that there is no clear separation of the red/hatched and green/unhatched histograms as in Fig. 7. The same conclusions can be drawn when the other factors are used.

We also evaluated whether it is possible to detect slots containing attack traffic using model $\tilde{\mathcal{M}}$. Fig. 13 shows the histograms of the metrics (a) *upload packets* and (b) *difference between upload and download packets* obtained by model $\tilde{\mathcal{M}}$ (offline method). The red/hatched (green/unhatched) bars of the histograms represent the values of metrics (a) and (b) obtained from slices $\tilde{\mathcal{M}}_{:,j,:}$ of the model for the minutes and residences with (without) attack. Note that, unlike Fig. 7 where residuals are used, there is a large overlap between the red and green bars in the histograms of Figs. 12 and 13 when loadings and model $\tilde{\mathcal{M}}$ are used, respectively.

We may also try to distinguish slots with attacks based on the $\mathbf{c}(t)$ loadings values, using the PWO online method. Fig. 14 shows the loadings values of the two factors of $\mathbf{c}(t)$ for each time slot t. Each point on the scatterplot represents a time slot. Red crosses corresponds to time slots in which at least one residence contains attack traffic during that minute, while green dots correspond to slots in which no residence



(a) Upload packet model values (b) Difference between upload and download packet model values

Fig. 13. Histograms of model values for minutes and residences with and without attacks.

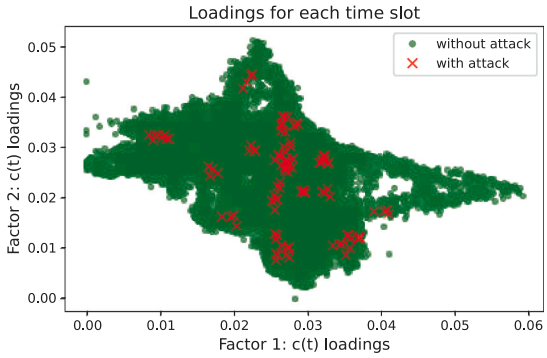


Fig. 14. $\mathbf{c}(t)$ loadings for time slots with and without residences attacking.

contains attack traffic during that minute. From the figure it is easy to see that one cannot distinguish slots containing attacks from those that do not using $\mathbf{c}(t)$ loadings. We conclude that, while features based on residuals are adequate to detect slots with attacks in our dataset, features based on PARAFAC loadings cannot differentiate these two cases.

6.4.2. Spatio-temporal correlation

It is possible to further improve the performance of the attack detector by correlating the detection at individual residences, since DDoS attacks are synchronized by nature. Mendonça et al. [20] proposed a Bayesian decision problem using MAP criterion to detect synchronized attacks with high probability. The decision problem is parameterized with the number of home routers N_R, the prior probability of attack P_D, the classifier's false positive rate p_{fp}, the classifier's recall p_{rc} and the prior probability of infection q. The model of [20] generates a threshold m_0 for the minimum number of residences simultaneously reporting an attack needed to identify a synchronized attack with very high accuracy.

Using the results of the Enhanced PWO online method ($N_R = 812, P_D \approx 0.0014, p_{fp} \approx 1.14 \times 10^{-6}, p_{rc} \approx 0.8864, q = 0.05$) to parameterize the model we obtain $m_0 \approx 4.09$. From this, a synchronized attack is reported if at least 5 users report an attack. In this case, detection is significantly improved since the probability of false alarms (Type I error) is 5.59×10^{-18} and the probability of missing a synchronized attack (Type II error) is 9.57×10^{-12}.

6.5. Unsupervised anomaly detection

In the previous sections we applied our proposed methods based on PARAFAC residuals to a scenario where we have labels for anomalies, enabling the use of supervised classifiers to detect DDoS attacks. However, in many situations, it may be hard to obtain labels. Furthermore, network anomalies are a moving target and new anomalies continue to arise over time [14]. Therefore, it is essential to devise methods capable

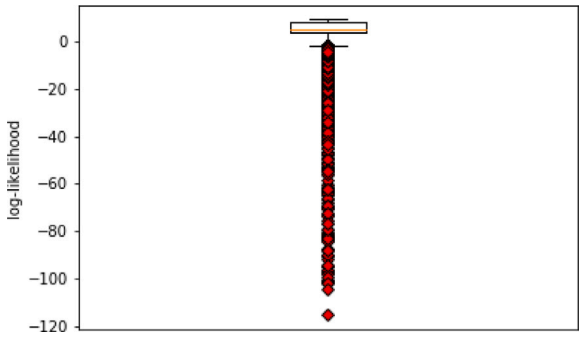


Fig. 15. Boxplot of the log-likelihood of each minute t of each RD i.

of detecting anomalies in the absence of ground truth labels. In this section, the framework of Section 4 is used to detect traffic anomalies (outliers) in an unsupervised manner. Since labels are not available we cannot train a classifier. Instead, we identify outliers based on the residuals extracted from a residential–day time-series (RD) that deviate from expected behavior using a simple statistical model.

The method consists of the following steps: (1) Extract the residuals of the training set; (2) Fit a multivariate GMM using the residual features of the training set; (3) Compute the likelihood that the residuals for slot t and RD i in the training set are explained by the GMM model; (4) Build a boxplot with the log-likelihoods obtained in (3); (5) Set a threshold α equal to the bottom whisker of the boxplot and use it to identify anomalies for a new set of data.

In the first step, from the trained tensor model, we extract the residuals of the traffic metrics for each minute and each RD, and obtain the following four features: *download/upload bit residuals* and *download/upload packet residuals*. These residual features are used to fit a multivariate Gaussian mixture (GMM) with two components and four dimensions. Then, for each RD i and time slot t, we calculate the likelihood that the corresponding residuals features are explained by the GMM model and build a boxplot with the log of the results (log-likelihoods). The bottom whisker of the boxplot is used as a threshold (α). In order to detect anomalous traffic in residences in a given period of time (not included in the trained dataset), we extract the residual features of all RD pairs during the considered period, using the previously trained tensor model. Then, for each time slot and RD pair in that period, we compute the likelihood of the new residuals, given the GMM fitted with the dataset used for training. The logs of these values are compared with the threshold α to identify anomalous traffic for each RD and each slot.

Motivated by the DDoS application, in what follows we assume that the root cause of an anomaly affects traffic from several residences simultaneously. Recall that, in our application example, traffic measurements from home routers are collected every minute and the number of residences collecting measures can vary with time. Then, for every time slot t in the considered period, we calculate the fraction of residences with detected anomalous traffic and use this fraction as a measure of importance of an anomaly at t as shown in the example below based on a real dataset.

Example. The following example uses data collected from residential routers from 1-March-2020 to 31-March-2020. This period was selected because our partner ISP notified us of a DDoS attack on the ISP's network from external bots between days 26 and 27 of March 2020. Since we stored the measurement data in our laboratory during that period, this information could then be used to assess the ability of our *unsupervised* method to identify a real DDoS attack using only packets and byte counts collected from residential routers. The dataset includes a total of 149,341 time-series from 5,490 residential routers.

Following the steps outlined above, we used the previous PARAFAC model trained with dataset Tr_1 and then fit a two-component GMM

using the residual tensor $\mathcal{E}$ and defined the threshold α from the boxplot (Fig. 15). Next, we computed new residuals for the RDs from the month of March 2020 using the trained PARAFAC model, calculated the likelihood using the GMM and finally estimated the fraction of residences with outlier traffic for every time slot t.

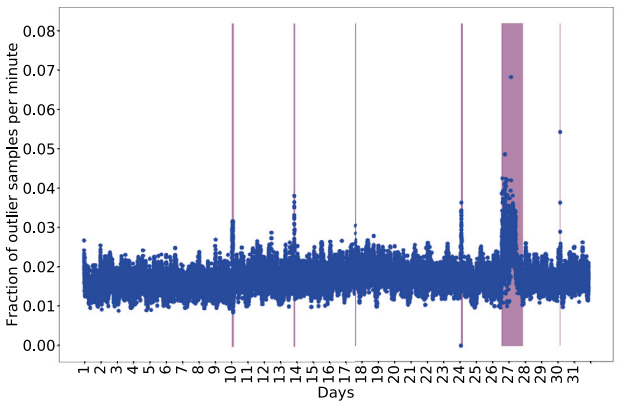
Fig. 16(a) shows the fraction of residences with outlier traffic (FRO(t)) during each minute during March-2020. Visually, from the figure, FRO(t) nearly doubles on many minutes during March 26–27.

Instead of using visual identification from the FRO(t) time-series of Fig. 16(a) we applied the CuSum (Cumulative Sum) [35] algorithm, which is commonly used to identify changes in data samples as compared to their mean. We assume that samples at each minute follow a Gaussian distribution and CuSum was parameterized using results from the training data. CuSum parameters were set to detect changes greater than three standard deviations and, in addition, the CuSum detection threshold was set to achieve a mean time between false detections equal to 31 days. The vertical purple lines in Fig. 16(a) identify the outlier samples, after applying CuSum. Note that, during the month, there are six periods containing outlier samples with a clear concentration of outliers during the days of March 26th to 28th. Since our data is not labeled we have no way of knowing what caused the anomalies except during the attack days.

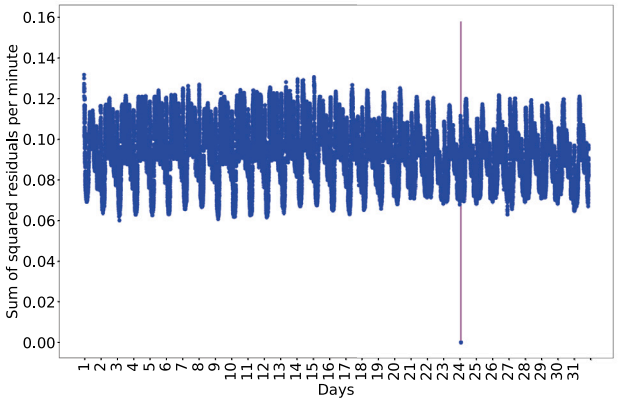
Comparison. To assess the effectiveness of our method compared with others in the literature we first use the *Sum of squared residuals per minute*, a commonly used metric for outlier detection using residuals extracted from factor models [36–38]. Second, we checked whether outliers can be detected using the *Median of download traffic per minute* directly. CuSum was applied in both cases. However, since both the *Sum of squared residuals* and *Median of download traffic* time-series exhibit strong seasonality as shown in Figs. 16(b) and 16(c), before using CuSum, both series are smoothed using an Exponentially Weighted Moving Average fitted using the training set, as done in previous works [39,40], to remove seasonality.

From Fig. 16(b) it is evident that the reported DDoS attack was not identified and only a single outlier event was marked in day 24. In contrast, when the median of the download traffic was used, too many outliers were marked after day 16 as shown in Fig. 16(c). The increase in traffic that is evident after March 16 can be explained by confinement measures adopted due to COVID-19 in the state in which the data was collected.

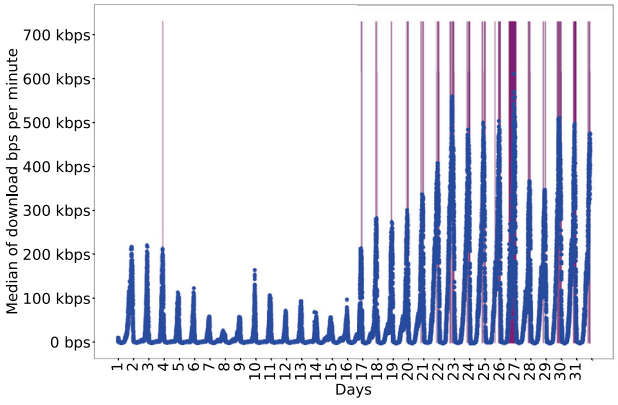
Using PARAFAC residuals we were able to detect a significant number of outliers on March 26–27 which corresponds to the single event for which we have knowledge about an anomaly in ground-truth (Fig. 16(a)). The other two considered approaches either missed such an event (Fig. 16(b)) or detected the event but also raised a significantly large number of additional alarms (Fig. 16(c)).



(a) Fraction of outlier samples



(b) Sum of squared residuals



(c) Median of download traffic

Fig. 16. Comparison of different metrics for detecting the DDoS attack originated by bots outside the ISP's network, between days 26 and 27 of March-2020. Periods in purple represent minutes detected as anomalous by the CuSum algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Application II: Detecting network degradation intervals

The second example application aims to employ our framework to detect when and where performance degradation occurs in the ISP's network. In the absence of reliable labels to identify most anomalies and evaluate the results quantitatively, we rely on *unsupervised clustering* over residuals extracted by the offline residual extraction procedure of Section 5.1 to group events with similar behavior. Our primary goal is to automatically detect potential network issues affecting multiple residences and to locate regions with poor performance. We show that our method can be effective in the real world by presenting examples of network events that were detected during the year of 2020. Finally, we correlate our results with customer tickets in order to understand the relationship between the obtained clusters and users' quality of experience (QoE).

7.1. Dataset construction and preprocessing

Our dataset consists of loss and latency metrics measured from thousands of residential routers to a server located in the ISP network. At every minute during data collection, a train of 100 ICMP probe packets is sent by the routers at 10 millisecond intervals, and both the fraction of packets lost in the train as well as the average round trip time of the probes in the train are computed and stored. We used two different datasets. The first, for fitting the model and clustering, contains 50,282 multivariate daily time-series collected from 2,964 residences between 19-August-2019 and 22-September-2019. A much larger dataset, is

used to evaluate the model. It contains 1,282,140 multivariate time-series collected from 6,307 residences between 16-January-2020 and 31-December-2020.

Latency and loss active measurements collected from the routers can be affected by the residential traffic (which we call cross-traffic) [41]. Therefore, we do not consider samples when cross-traffic exceeds a threshold θ. In other words, during every minute, when the upload or download residential traffic is above θ, latency and loss measurements are discarded for that minute. We set $\theta = 2.5$ Mbps after studying the impact of θ on the sample values and based on traffic from residences with the lowest nominal capacity. In addition to discarding samples with cross-traffic, we also discarded daily time-series with less than 1,000 valid measurements during a day.

For some network failures, such as link failures, communication between the client and the measurement server can be interrupted and no measurement samples are recorded. Therefore, it is possible to infer periods of network unavailability from the lack of measurement samples, especially when multiple residences do not report results during the same measurement slot. Hence, we mark slots lacking measurements to indicate 100% packet loss and considered latency a missing value.

7.2. Tensor decomposition and residual extraction

We model the measurement data as a tensor with three modes: (RD × network metric × minute). Model $\mathcal{X} \in \mathbb{R}^{50282\times2\times1440}$ is obtained from the daily time-series of latency/loss measurements (first dataset). We use the Split-Half Validation to choose the number of factors ($R = 4$).

From the model residuals, we used time-series corresponding to latency residuals and loss residuals. The latter time-series is further divided into two: one that includes residuals from all time slots and another where residuals corresponding to slots with 100% losses are marked as missing samples. We calculate three statistics for each of the three time-series of residuals: mean, standard deviation and 95th percentile, totaling nine features that will be used for clustering.

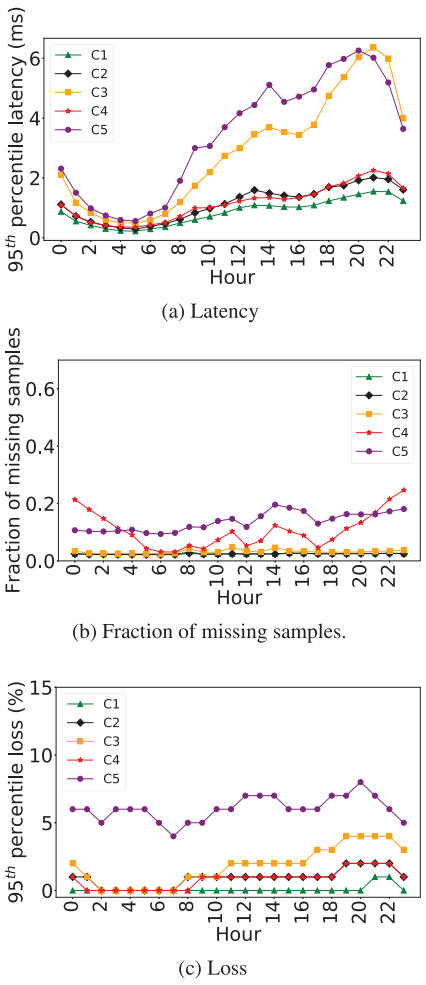
7.3. Anomaly clustering

The clustering algorithm uses the nine features extracted from the residuals of each RD pair. In Section 7.3.1 we show that each cluster can be associated with a distinct level of degradation in network performance. Section 7.3.2 describes how we correlate residences spatially in order to evaluate the quality of service provided in different network regions and detect performance degradation events affecting multiple residences. From a real-world example, we present evidence in Section 7.3.3 that our unsupervised algorithm based on PARAFAC residuals was able to detect performance related events. Finally, in Section 7.3.4, we correlate the clustering results with customer tickets to understand the relationship between different clusters and the users' quality of experience and also describe how our method can be used to improve ISP troubleshooting.

7.3.1. Clustering results

A popular approach for unsupervised clustering is the K-means algorithm. It can be shown that K-means is a variant of the Expectation–Maximization algorithm for Gaussian Mixtures where equal spherical covariance matrices for each cluster are considered and a hard cluster assignment is performed [42]. Since, the Gaussian Mixture Model (GMM) is a more flexible and general alternative for clustering, we choose GMM to cluster the model residuals. To select the number of clusters, we use the Elbow Method [14] and chose five clusters. Before clustering the data, we apply z-score normalization to prevent any feature dominating the analysis due to scaling.

To investigate the meaning of the clusters we compute the 95th percentile latency, the 95th percentile loss and the fraction of missing samples for each cluster. These statistics are calculated for every hour of the day using the loss and latency minute-samples of the RDs associated with a given cluster. To infer packet queueing times during congestion periods, we subtracted from each latency time-series its lowest sample value.



(a) Latency

(b) Fraction of missing samples.

(c) Loss

Fig. 17. Statistics computed for each cluster.

Figs. 17(a)–17(c) show the 95th percentile latency, the fraction of missing samples and the 95th percentile loss, respectively. Residences of cluster C1 experience good quality of service (low latency, low loss, low network unavailability). Cluster C2 contains time-series with moderate losses but low network unavailability and low latency. Cluster C3 contains time-series with high latency and moderate losses. The time-series of cluster C4 are from residences that experience high network unavailability periods, while cluster C5 contains time-series with high network unavailability, latency and loss. Note that we cluster RD pairs, i.e., daily measurement time-series of different residences. Therefore, a residence can be assigned to different clusters on different days.

7.3.2. Spatial correlation

To identify periods experiencing performance degradation that affect multiple residences in the same "region" of the network, we spatially correlate the clustering results and ISP topology information. The spatio-temporal correlation algorithm assumes that the routes between residential routers and the measurement server are static during each measurement interval. Consequently, the network topology can be represented by a tree structure at each measurement interval. We expect clients that share the same ISP network paths to exhibit similar performance inside the ISP network.

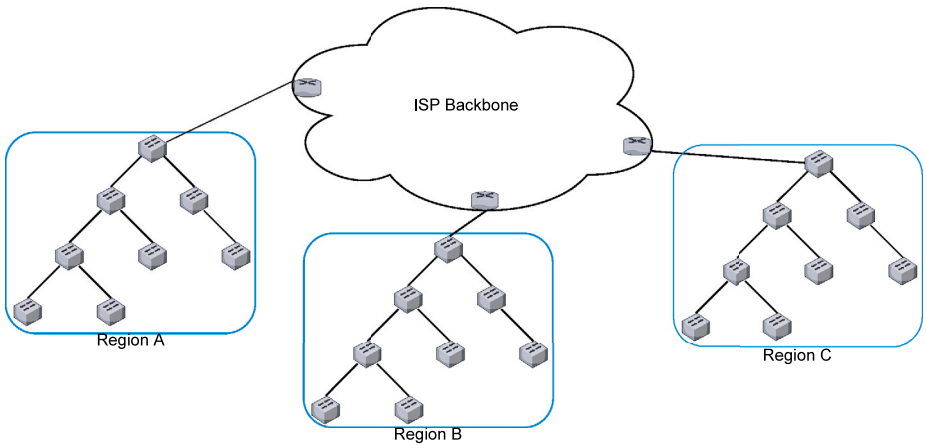


Fig. 18. Partition of network into network regions.

In order to carry out the spatial correlation procedure, we partition the network into non-overlapping subsets denominated *network regions*. Different partition criteria could be considered to obtain network regions. In our dataset, network equipment is divided by the ISP based on a geographical partition of the city. We take this into account and define a network region in our dataset as a subset of network equipment connecting a neighborhood to the ISP's backbone, as illustrated in Fig. 18. Consequently, each network region contains multiple network equipment that are geographically close and are shared by multiple clients. Based on this partition, we analyze the fraction of residences assigned to each cluster at each day of the dataset for each network region.

We illustrate the results of the spatial correlation for a particular network region. Similar results are obtained for other regions. Fig. 19(d) shows the daily fraction of RD pairs per cluster for a given network region. From the figure, we observe that the majority of residences are allocated to cluster C1 (good quality of service) on most days. Fig. 19(a) illustrates the daily time-series of packet losses for a single residence that was assigned to cluster C1 on day 6. Note that the loss fraction is close to zero for most minutes of that day. On the other hand, a large number of residences are allocated to clusters C4 and C5 on day 10. (Recall that clusters C4 and C5 are associated with periods of high network unavailability.) In this day, there are multiple time-series of packet losses with missing samples (red squares in the figure) between hours 13:00 and 17:00, as illustrated for one of the RDs in Fig. 19(b). Another type of event detected by the spatial correlation occurs on day 17. On that day, several time-series associated to losses have missing samples between 6:00 and 8:00 and high losses between 19:00 and 21:00. These were assigned to cluster C5 on day 17, and Fig. 19(c) shows the measurement results of one of these residences.

The ISP can benefit from the spatial correlation results to prioritize efforts to improve the quality of service based on the network performance of each region. To illustrate this idea, in Fig. 20 we present clustering results for two different network regions with distinct performance results. It is clear that region A consistently presents a high fraction of residences associated with good performance (Fig. 20(a)), and only a few days (9, 10, and 33) suffer from performance degradation. On the other hand, only a few residences were assigned to cluster C1 (good performance) in region B (Fig. 20(b)).

7.3.3. Events which were detected

The analysis in Section 7.3.2 was based on *unsupervised clustering* using PARAFAC residuals. We showed that it is possible to identify network regions and time intervals with different levels of performance, from good to bad, and that affect multiple residences. In what follows, we present a few network events detected by our approach for which root causes were later identified by our partner ISP. The events were detected using the dataset collected during 2020. Note that we were unable either to categorize all anomalies in our dataset or to automatically classify the type of anomaly, since labels are scarce and hard to obtain.

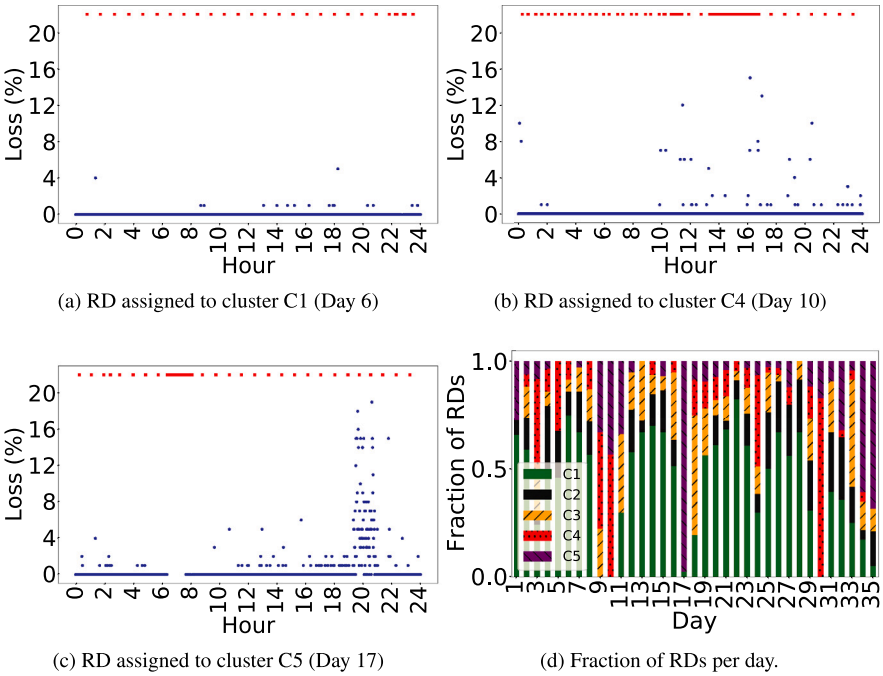
Fig. 21(a) presents the clustering results of all RDs during the month of March 2020. It shows that all residences were assigned to cluster C5 (purple) on days 26 and 27. A DDoS attack targeting the ISP was performed during these two days. The attack affected the entire network and residences experienced network outages during this period. Our method was able to capture this behavior automatically, assigning all the clients to a cluster related to periods of network unavailability. Note that this DDoS attack was also detected by the method of Section 6.5.

Figs. 21(b) and 21(c) show two examples of quality of service improvement caused by ISP interventions executed on May 2020. In the first, an upgrade of network equipment by the ISP in region C on day 14, resulted in improved quality of service for many residences. For instance, if we calculate the daily fraction of residences in each cluster and average the results over the days that precede and succeed day 14, we discover that more than 20% of residences migrate to cluster C1 after day 14, from clusters associated with inferior performance. In the second example, a reconfiguration of the network topology started on day 23 in region D, clearly increasing the fraction of residences that belong to C1 in subsequent days.

Next, we investigate whether it is possible to observe changes in network performance directly from the model loadings, instead of from the residuals. For illustrative purposes, we compare both approaches (loadings versus residuals) using the equipment upgrade event that took place in Region C on May 14, 2020 (Fig. 21(b)). Fig. 22 shows the median of the RD loadings for one of the factors and the median of the residual feature that correspond to the measured losses. We observe that, unlike Fig. 22(a) the median of the residual feature (Fig. 22(b)) shows a clear change after day 14. The same conclusion was drawn when other factors were used.

Fig. 21(d) shows an example of an event (equipment failure) that we detected on August 2020 in day 2, that caused some customers to be disconnected for a few hours. The failed equipment was replaced on day 5 causing further disruption during service hours. Our method assigned the vast majority of residences in the affected region to a cluster associated with unavailable intervals (cluster C5) on days 2 and 5.

In addition to events that occur during a day, we were also able to detect performance trends over longer periods. We analyzed our dataset during 5 months, between March 1st and July 31st 2020. Fig. 23 shows clearly that performance degraded around mid March, since the fraction of residences in the cluster with the best performance was approximately halved. Many days after, performance slowly improved during the following months as residences return to cluster C1.



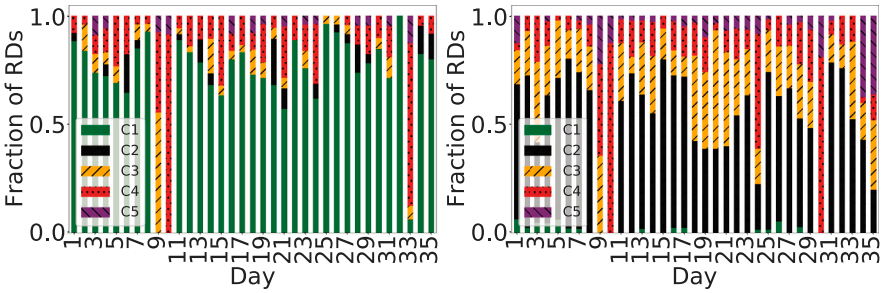
(a) RD assigned to cluster C1 (Day 6)

(b) RD assigned to cluster C4 (Day 10)

(c) RD assigned to cluster C5 (Day 17)

(d) Fraction of RDs per day.

Fig. 19. Spatial correlation example (missing samples are represented using red squares).



(a) Region A

(b) Region B

Fig. 20. Network performance of two different network regions obtained from residual clustering.

A plausible explanation for the observed trend is the considerable change in user behavior during confinement resulting in increased network traffic and reduced network performance. The confinement period in the region where data was collected started on March 16th and restriction rules were slowly relaxed during the subsequent weeks.

7.3.4. Clusters and quality of experience

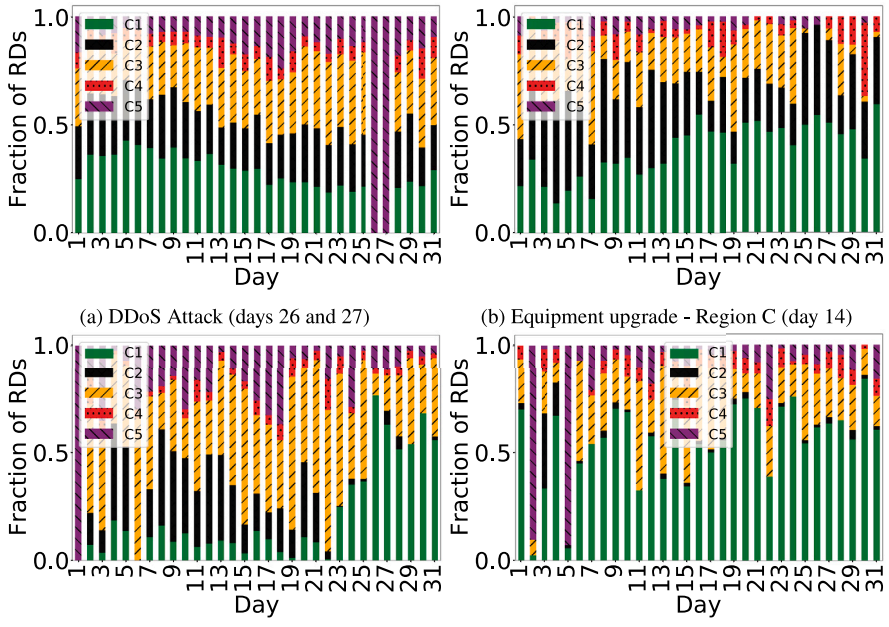
User feedback obtained from complaints to the call center of an ISP could, at a first glance, be interpreted as a good metric for assessing the user's quality of experience (QoE) [43]. However, user feedback is very noisy since, for instance, customers may complain about poor network performance while the problem is in the home network rather than an ISP issue and, in addition, many customers may not complain even if the network is performing poorly [44]. Therefore, correlating customer complaints with QoS can be an important tool to facilitate QoE assessment.

We correlate the clustering results obtained using our second dataset (from January 16th to December 31st 2020) with a database of customer's technical complaints provided by the partner ISP, in order to infer the relationship between clusters and the customers' quality of experience. Data from customers is properly anonymized to preserve privacy. Fig. 24(a) shows, for each cluster, the ratio of the number of RDs from a cluster with customer tickets by the total number of RDs in that cluster (times 100). Clearly, clusters C4 and C5, which are associated with poor performance, have the highest percentage of RDs with customer tickets. Clusters C2 and C3, associated with moderate losses and latency, have smaller fractions of RDs with customer tickets than clusters C4 and C5. These results indicate that customers in clusters associated with poor performance are more likely to complain to the call center.

To understand how customer tickets are distributed among clusters we plot in Fig. 24(b) the percentage of tickets from customers associated with each of the five clusters. Approximately 60% of all complaints are from customers in C4 and C5, the clusters with the worst QoS, while only approximately 20% of the total number of tickets comes from customers in C1 and C2.

8. Conclusion

In this work, we propose a framework based on tensor decomposition to detect network anomalies. We apply the PARAFAC method and extract the residuals obtained by the model in order to detect abnormal behavior. We also propose a new online tensor decomposition method that efficiently extracts the normal subspace and detects anomalies with good performance. We show the flexibility of our method, using different applications as examples. First, we considered DDoS attack detection using supervised and unsupervised techniques. Our results show that detection has high accuracy and very low false positive rates when



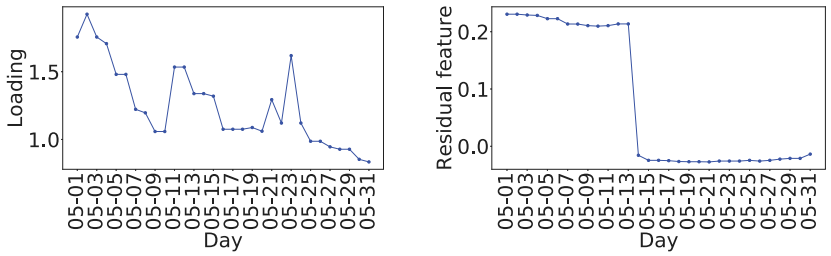
(a) DDoS Attack (days 26 and 27)

(b) Equipment upgrade - Region C (day 14)

(c) Network reconfiguration - Region D (day 23)

(d) Network maintenance - Region E (day 2 and day 5)

Fig. 21. Events detected by our unsupervised approach.



(a) Median of the model loadings (Region C)

(b) Median of the residual feature (Region C)

Fig. 22. Loadings vs Residuals for event detection (smoothed with a median filter with window size of 3).

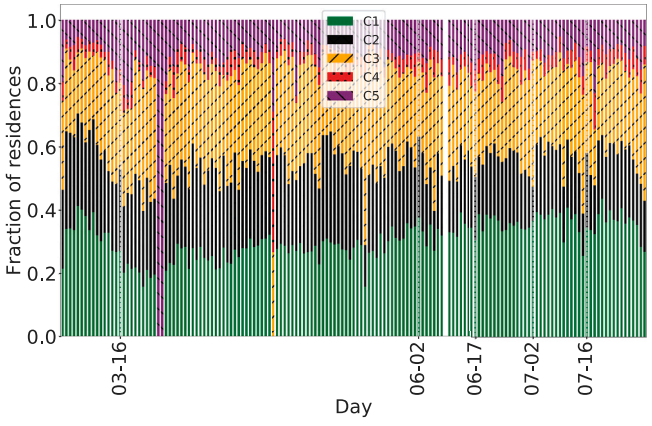
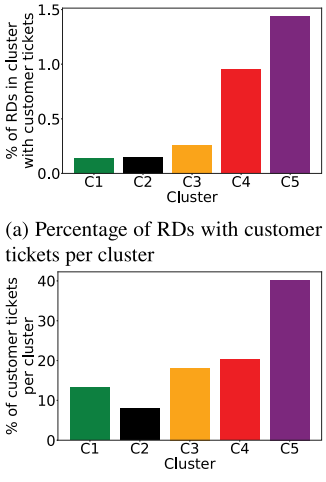


Fig. 23. Clustering results from March 1st to July 31th. The first labeled day represents the start of lockdown while the following ones represent the starting days of different phases of confinement relaxation.



(a) Percentage of RDs with customer tickets per cluster

(b) Percentage of customer tickets per cluster

Fig. 24. Relationship between cluster results and customer tickets.

applied to real data collected from thousands of residential routers. In addition, we were able to detect an actual DDoS attack event using the unsupervised technique. Second, we applied the proposed framework to identify periods within which network performance deteriorates. We show how real network events could be detected in different regions of our partner ISP's topology. In addition, we correlate the results

of our unsupervised method with a database of customer's technical complaints provided by the ISP.

CRediT authorship contribution statement

Ananda Streit: Methodology, Formal analysis, Investigation, Writing - original draft. **Gustavo H.A. Santos:** Methodology, Formal analysis, Investigation, Writing - original draft. **Rosa M.M. Leão:** Conceptualization, Methodology, Resources, Supervision, Writing - review & editing. **Edmundo de Souza e Silva:** Conceptualization, Methodology, Resources, Supervision, Writing - review & editing. **Daniel Menasché:** Writing - review & editing. **Don Towsley:** Conceptualization, Methodology, Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.comnet.2021.108503.

References

[1] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15.

[2] F. Silveira, C. Diot, N. Taft, R. Govindan, Astute: Detecting a different class of traffic anomalies, ACM SIGCOMM CCR 41 (4) (2011) 267–278.

[3] S. Fadilpašić, AWS hit by DDoS attack, 2019, URL https://www.itproportal.com/news/aws-hit-by-ddos-attack/.

[4] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez, et al., The Lockdown Effect: Implications of the COVID-19 Pandemic on Internet Traffic, in: Proceedings of the ACM Internet Measurement Conference, 2020, pp. 1–18.

[5] J. Spataro, Our commitment to customers during COVID-19, 2020, URL https://www.microsoft.com/en-us/microsoft-365/blog/2020/03/05/our-commitment-to-customers-during-covid-19/.

[6] Sandvine, The global internet phenomena report: Covid-19 spotlight, 2020, URL https://www.sandvine.com/covid-internet-spotlight-report.

[7] H.S. Lallie, L.A. Shepherd, J.R.C. Nurse, A. Erola, G. Epiphaniou, C. Maple, X. Bellekens, Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic, Comput. Secur. 105 (2021) 102248.

[8] ENISA Report, Distributed Denial of Service: ENISA Threat Landscape from January 2019 To April 2020, ENISA (European Union Agency for Cybersecurity), Athens, Greece, 2020.

[9] A10 Networks, Q4 2019 - The State of DDoS Weapons Report, San Jose, California, 2019, URL https://www.a10networks.com/marketing-comms/reports/state-ddos-weapons/.

[10] S. Stein, J. Jacobs, Cyber-Attack Hits U.S. Health Agency Amid Covid-19 Outbreak, 2020, URL https://tinyurl.com/tbgq6as.

[11] M. Candela, V. Luconi, A. Vecchio, Impact of the COVID-19 pandemic on the internet latency: A large-scale study, Comput. Netw. 182 (2020) 107495.

[12] M. Trevisan, D. Giordano, I. Drago, M.M. Munafò, M. Mellia, Five years at the edge: Watching internet from the ISP network, IEEE/ACM Trans. Netw. 28 (2) (2020) 561–574, http://dx.doi.org/10.1109/TNET.2020.2967588.

[13] A. Lakhina, M. Crovella, C. Diot, Diagnosing network-wide traffic anomalies, in: Comp. Comm. Review, 34, (4) 2004, pp. 219–230.

[14] A. Lakhina, M. Crovella, C. Diot, Mining anomalies using traffic feature distributions, in: ACM Computer Communication Review, vol. 35, no. 4, 2005, pp. 217–228.

[15] R. Bro, Parafac. Tutorial and applications, Chemometr. Intell. Lab. Syst. 38 (2) (1997) 149–171.

[16] J. Sun, D. Tao, S. Papadimitriou, P.S. Yu, C. Faloutsos, Incremental tensor analysis: Theory and applications, ACM Trans. Knowl. Discov. Data (TKDD) 2 (3) (2008) 11.

[17] K. Maruhashi, F. Guo, C. Faloutsos, Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis, in: Advances in Social Networks Analysis & Mining, 2011, pp. 203–210.

[18] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, D. Zhang, Graph based tensor recovery for accurate Internet anomaly detection, in: IEEE INFOCOM 2018, 2018, pp. 1502–1510.

[19] R. Doshi, N. Apthorpe, N. Feamster, Machine Learning DDoS Detection for Consumer IoT Devices, in: IEEE Security and Privacy Workshops, 2018, pp. 29–35.

[20] G. Mendonça, G.H. Santos, E. de Souza e Silva, R.M. Leão, D. Menasché, D. Towsley, An extremely lightweight approach for DDoS detection at home gateways, in: 2019 IEEE International Conference on Big Data (Big Data), IEEE, 2019, pp. 5012–5021.

[21] A. D'Alconzo, I. Drago, A. Morichetta, M. Mellia, P. Casas, A survey on big data for network traffic monitoring and analysis, IEEE Trans. Net. and Service Manag. 16 (3) (2019) 800–813.

[22] N.D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E.E. Papalexakis, C. Faloutsos, Tensor decomposition for signal processing and machine learning, IEEE Trans. Signal Process. 65 (13) (2017) 3551–3582.

[23] X. Li, K. Xie, X. Wang, G. Xie, J. Wen, G. Zhang, Z. Qin, Online internet anomaly detection with high accuracy: A fast tensor factorization solution, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 1900–1908.

[24] H. Kasai, W. Kellerer, M. Kleinsteuber, Network volume anomaly detection and identification in large-scale networks based on online time-structured traffic tensor tracking, IEEE Trans. Netw. Serv. Manag. 13 (3) (2016) 636–650.

[25] D. Nion, N.D. Sidiropoulos, Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor, IEEE Trans. Signal Process. 57 (6) (2009) 2299–2310.

[26] S. Zhou, N.X. Vinh, J. Bailey, Y. Jia, I. Davidson, Accelerating online cp decompositions for higher order tensors, in: ACM SIGKDD Knowledge Discovery and Data Mining, 2016, pp. 1375–1384.

[27] A. Streit, G. Santos, R.M. Leão, E. de Souza e Silva, D. Menasché, D. Towsley, Network anomaly detection based on tensor decomposition, in: 2020 Mediterranean Communication and Computer Networking Conference (MedComNet), IEEE, 2020, pp. 1–8.

[28] R.A. Harshman, "how can I know if it's real?" a catalogue of diagnostics for use with three-mode factor analysis, Res. Methods Multimode Data Anal. (1984) 566–591.

[29] U. Lorenzo-Seva, J.M. Ten Berge, Tucker's congruence coefficient as a meaningful index of factor similarity, Methodology 2 (2) (2006) 57–64.

[30] N. Blenn, V. Ghiëtte, C. Doerr, Quantifying the Spectrum of Denial-of-Service Attacks through Internet Backscatter, in: Conference on Availability, Reliability and Security, 2017, p. 21.

[31] E. Auchard, German Internet outage was failed botnet attempt: report, Reuters, URL https://tinyurl.com/reutersoutage.

[32] C. Bergmeir, J.M. Benítez, On the use of cross-validation for time series predictor evaluation, Inform. Sci. 191 (2012) 192–213, http://dx.doi.org/10.1016/j.ins.2011.12.028, Data Mining for Software Trustworthiness.

[33] H. Ringberg, A. Soule, J. Rexford, C. Diot, Sensitivity of PCA for traffic anomaly detection, in: Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2007, pp. 109–120.

[34] E. Acar, R. Bro, B. Schmidt, New exploratory clustering tool, J. Chemometrics: A J. Chemom. Soc. 22 (1) (2008) 91–100.

[35] E.S. Page, Continuous inspection schemes, Biometrika 41 (1/2) (1954) 100–115.

[36] A. Smilde, R. Bro, P. Geladi, Multi-Way Analysis: Applications in the Chemical Sciences, John Wiley & Sons, 2005.

[37] P.M. Kroonenberg, Applied Multiway Data Analysis, vol. 702, John Wiley & Sons, 2008.

[38] J. Riu, R. Bro, Jack-knife technique for outlier detection and estimation of standard errors in PARAFAC models, Chemometr. Intell. Lab. Syst. 65 (1) (2003) 35–49.

[39] D.C. Montgomery, Statistical Quality Control, Wiley Global Education, 2012.

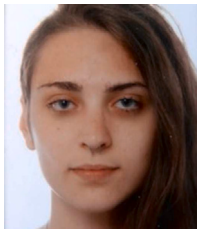
[40] V.A. Siris, F. Papagalou, Application of anomaly detection algorithms for detecting SYN flooding attacks, Comput. Commun. 29 (9) (2006) 1433–1442.

[41] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, A. Pescapè, Broadband Internet Performance: A View From the Gateway, in: ACM SIGCOMM 2011, 2011.

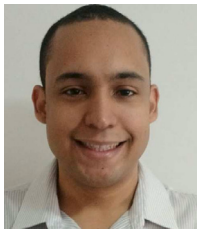
[42] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.

[43] H.H. Song, Z. Ge, A. Mahimkar, J. Wang, J. Yates, Y. Zhang, A. Basso, M. Chen, Q-score: Proactive service quality assessment in a large IPTV system, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, 2011, pp. 195–208.

[44] J. Hu, Z. Zhou, X. Yang, J. Malone, J.W. Williams, CableMon: Improving the reliability of cable broadband networks via proactive network maintenance, in: 17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20), 2020, pp. 619–632.



Ananda Streit is a D.Sc. student in Computer Engineering at the Federal University of Rio de Janeiro. She holds a M.Sc. in Computer Engineering from the Federal University of Rio de Janeiro, and a B.Sc. in Computer Science from the University Center of Brasilia. Her research interests include multivariate analysis, machine learning and network measurements.



Gustavo H.A. Santos is a D.Sc. student in Computer Engineering at the Federal University of Rio de Janeiro. He holds a M.Sc. in Computer Engineering and a B.Sc. in Computer Science from the Federal University of Rio de Janeiro. His research interests include machine learning, network measurements and performance evaluation.

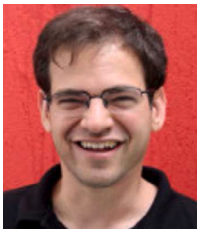
Rosa M.M. Leão received the B.Sc. degree in Computer Science from the Federal University of Rio de Janeiro in 1983, the M.Sc. degree in Computer Science from PUC-Rio in 1990, and the Ph.D. degree in Computer Science from the Paul Sabatier University (LAAS) in 1994. Currently she is an Associate Professor in the Systems Engineering and Computer Science Department at the Federal University of Rio de Janeiro. Her research interests include performance evaluation, machine learning, networking modeling and analysis, cache networks, network measurements, p2p systems.

Edmundo de Souza e Silva received the B.Sc. and M.Sc. degrees in electrical engineering, both from Pontifical Catholic University of Rio de Janeiro (PUC/RJ), and the Ph.D. degree in computer science from the University of California, Los Angeles in 1984.

Edmundo was a visiting professor/researcher at renowned universities and research centers including the IBM T.J. Watson research Center, IBM Tokyo Research Laboratory, UCLA Department of Computer Science, Computer Science Department at USC, Politecnico di Torino, Chinese University of Hong Kong, IRISA/INRIA-Rennes, University of Massachusetts at Amherst.

He has served as Technical Program Committee co-Chair of major international conferences including IEEE/Globecom'1999, ITC'2001, ACM/Sigmetrics'2002, IEEE/Infocom 2009. Edmundo was elected for the ACM/SIGMETRICS Board of Directors for 2 terms (2001-2005), and was Chair of the IFIP WG 7.3 for the 2008-2014 terms. He serves on the Editorial Board of the Journal of Internet Services and Applications (Springer) and the ACM Transactions on Modeling and Performance Evaluation of Computing Systems. He is a member of the Brazilian Academy of Sciences and the National Academy of Engineering. In 2008 he received the medal of the National Order of Scientific Merit.

Currently he is a professor of the Systems Engineering and Computer Science at the Federal University of Rio de Janeiro, COPPE. His areas of interest include the modeling and analysis of computer systems and computer communication multimedia networks.



Daniel Sadoc Menasché received the Ph.D. degree in computer science from the University of Massachusetts, Amherst, in 2011. Currently, he is an Assistant Professor with the Computer Science Department, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil. His interests are in modeling, analysis, and performance evaluation of computer systems. Dr. Menasché received Best Paper Award at GLOBECOM 2007, CoNEXT 2009, INFOCOM 2013 and ICGSE 2015.

Don Towsley's research spans a wide range of activities from stochastic analyzes of queueing models of computer and telecommunications to the design and conduct of measurement studies. He got his Ph.D. in Computer Science from the University of Texas (1975) and BA Physics, University of Texas (1971). Professor Towsley is a University Distinguished Professor at the University of Massachusetts, Amherst in the College of Information and Computer Sciences. He has been an editor of the IEEE Transactions on Communications, IEEE/ACM Transactions on Networking, and Journal of Dynamic Discrete Event Systems. He is currently on the Editorial boards of Networks and Performance Evaluation. He is a two-time recipient of the Best paper Award of the ACM Sigmetrics Conference. He is a Fellow of the IEEE and of the ACM. He is also a member of ORSA and is active in the IFIP Working Groups 6.3 on Performance Modeling of Networks and 7.3 on Performance Modeling. Towsley is the recipient of one of the IEEE's most prestigious honors, the 2007 IEEE Koji Kobayashi Computers and Communications Award. He also received a UMass Amherst Distinguished Faculty Lecturer award in 2002 and a UMass Amherst College of Natural Sciences and Mathematics Faculty Research Award in 2003.