FAULT LOCALIZATION IN COMPUTER NETWORKS USING END-TO-END MEASUREMENTS

Diego Ximenes Mendes

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Edmundo Albuquerque de Souza e Silva

Rio de Janeiro
Janeiro de 2017

FAULT LOCALIZATION IN COMPUTER NETWORKS USING END-TO-END MEASUREMENTS

Diego Ximenes Mendes

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2017

Ximenes Mendes, Diego

Fault Localization in Computer Networks Using End-to-End Measurements/Diego Ximenes Mendes. – Rio de Janeiro: UFRJ/COPPE, 2017.

IX, 26 p.: il.; 29, 7cm.

Orientador: Edmundo Albuquerque de Souza e Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2017.

Bibliography: p. 23 – 26.

1. Network Measurements. 2. Time Series. 3. Change Point Detection. 4. Machine Learning. I. Albuquerque de Souza e Silva, Edmundo. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

FAULT LOCALIZATION IN COMPUTER NETWORKS USING END-TO-END MEASUREMENTS

Diego Ximenes Mendes

Janeiro/2017

Orientador: Edmundo Albuquerque de Souza e Silva

Programa: Engenharia de Sistemas e Computação

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

FAULT LOCALIZATION IN COMPUTER NETWORKS USING END-TO-END MEASUREMENTS

Diego Ximenes Mendes

January/2017

Advisor: Edmundo Albuquerque de Souza e Silva

Department: Systems Engineering and Computer Science

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Contributions

1.2 Dissertation Outline

Chapter 2

Literature Review

This chapter briefly describes three projects (Argus [1], NetNorad [2] and CEM [3]) that use end-to-end QoS to localize faults in computer networks.

2.1 Argus

In [1] is presented Argus, a system to detect and localize problems in ISP's networks. To achieve this goal, Argus uses network global information, and also passively collected data from the ISP's viewpoint to infer end-to-end QoS, such as traffic to/from end-users to estimate achievable download speed [4].

The system's analytics pipeline is illustrated in Figure 2.1.

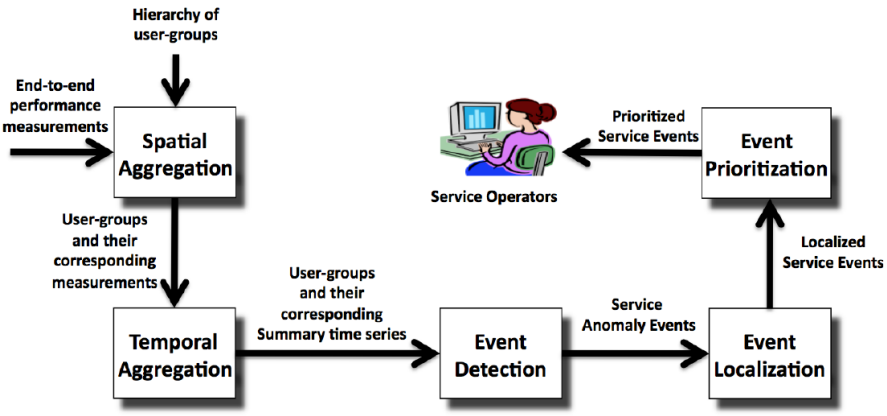


Figure 2.1: Argus pipeline. [1]

The analysis starts with the Spatial Aggregation procedure, in which end-users are clustered into user-groups. This step improves the system's scalability, since avoids keeping track the performance of all individual end-users. Each user-group is characterized by a set of end-users that share some common attributes, such as AS or BGP prefix. The used features imposes the possible fault locations to be inferred. An example of a spatial aggregation is depicted in Figure 2.2.

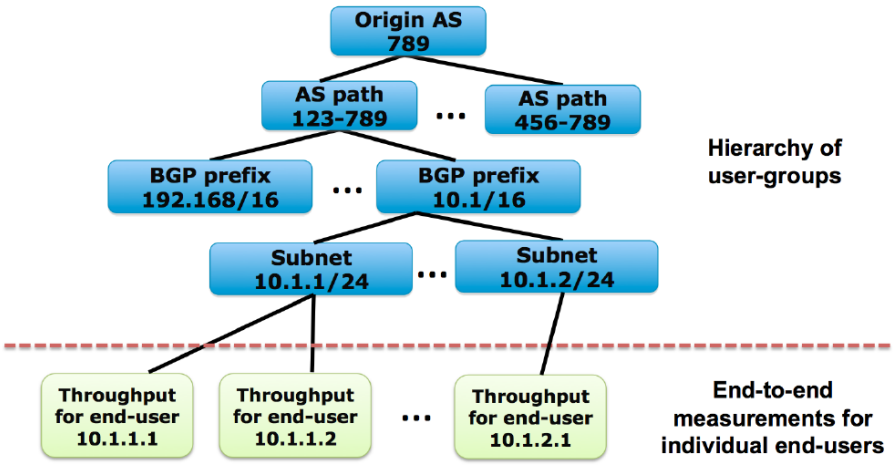


Figure 2.2: Argus spatial aggregation. [1]

The Temporal Aggregation phase determines how data from different end-users of an end-group are combined. For each user-group, the measurements of all end-users are grouped in time-bins, and for each time-bin a summary statistic, such as median or mean, is selected. Each type of fault can be better tracked by a specific statistic. As an example, the minimum of the RTTs can capture the physical propagation delay, while the average can be related with network congestion. Argus uses median as the default transformation, since it was empirically identified as effective to track network flaws, and also robust to individual end-users variability caused by their local infrastructure.

The Event Detection procedure identifies anomalies in the summary time series. Argus uses a Holt-Winters variation, which consists of an online method with low run time and memory complexities.

The responsibility of the Event Localization step is to infer fault locations using spatial and events times correlations. However, the detailed description of how this process is implemented was not published.

Finally, the detected problems are sorted according with their significance, which considers metrics obtained through the event detection algorithm, and also the number of affected end-users.

Argus was evaluated using RTT measurements of a CDN hosted in a tier-1 ISP. During an one month, 2909 anomalous events were detected. In general, lower level user-groups were more responsible for those events than the higher level groups, and only a small fraction of the end-users caused the end-group anomaly. Also, 90% of the events lasted for at most 1 hour, which was the used time-bin granularity.

Although not investigated by the Argus's authors, the fact that only a small number of end-users are responsible for the end-groups events, is an indication that fault localization can achieve higher precision with finer spatial aggregation granularity. Besides, the system accuracy was not studied.

2.2 NetNorad

NetNorad [2] consists of a Facebook's internal project to automate the analysis of faults in the Facebook's network. Previous deployed techniques by Facebook exhibit several disadvantages, for instance, human-driven investigation may take hours. Also, cases known as gray failures cannot be detected only collecting devices information through SNMP or command line interface. For example, some devices cannot report it's own malfunctioning, or some problems can be related with the global network structure.

Facebook's network is structured hierarchically. At the lowest level there are servers in racks, which are organized in clusters. A set of clusters in the same building and attached to the same network characterize a data center. Data centers are grouped through a network that interconnects them within the same region, and appends to the Facebook global backbone. Figure 2.3 presents an example of this architecture.

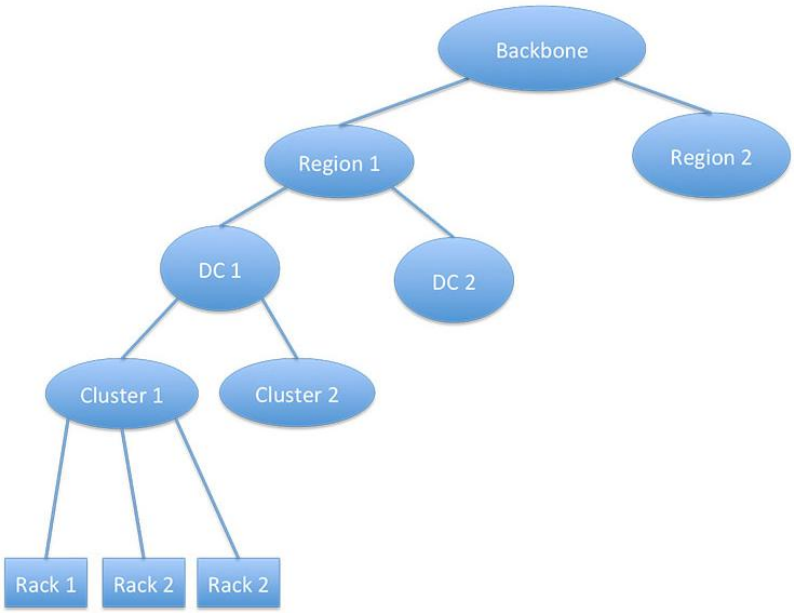


Figure 2.3: Facebook's network architecture. [2]

Unlike Argus, NetNorad uses active probing to assess loss and RTT statistics. Facebook's servers ping each other, in which a pinger sends UDP packets to responders, and the latter send the packets back. The process happens in turns, each pinger sends packets to all targets, collects the responses, and then repeats the procedure. A small number of pingers are placed in each cluster, and the responders are deployed on all machines. All pingers share a target list, which includes at least two machines of every rack.

As with Argus, NetNorad applies spatial aggregation techniques. The pingers group the responses of machines that belong to the same cluster during a round, and

tags them according with their relative location. Tags are defined by the following patterns: "DC" if the target cluster is in the pinger's data center; "Region" if the target cluster is outside the pinger's data center but within the same region; "Global" if the target cluster is outside the pinger's region.

With the tagging process, each cluster have three time series reflecting different spatial viewpoints, which are tracked through distinct percentiles over 10-minute intervals, enabling a fault mitigation reasoning. For instance, a packet loss spike at the 50th percentile means that probably there is a failure affecting the majority of machines, while a peak at the 90th and not at 50th percentile can indicate a small fraction of anomalous servers. For each combination of proximity tag and percentile is defined two thresholds, one for trigger and another for clear an alarm.

Considering the three tags, if a high loss is detected in a cluster, then the fault is probably located at the cluster's data center. Also, if all clusters in a data center identify a QoS degradation, then the fault is likely to be placed a layer above the clusters. Although these simple inferences can reduce the set of possible fault locations, they are unable to exactly isolate them. However, a Facebook tool called fbtracert can improve this analysis, exploring multiple paths between two endpoints, and checking the packet loss levels at every hop. Nonetheless, fbtracert exhibits several limitations.

When automatic investigation is unable to find the failure, then there is a human involvement to find it. A detailed accuracy analysis is not presented, however, the infrastructure allows alarms to be raised about 30 seconds far from the network events.

2.3 CEM

In [3] is proposed a framework called Crowdsourcing Event Monitoring (CEM), in which a monitoring software that runs inside or alongside applications is placed at the end-hosts, enabling the detection of service level events within seconds or minutes.

In CEM, each end-host passively collects performance metrics related with a specific service, such as a VOD application. Through these data, and to increase the system's scalability, the end-host itself identifies local problems as potential network events, and pushes them to a distributed storage to further analysis. The framework doesn't specifies how events should be detected, however, they must be associated with service level problems.

To spatially isolate network flaws, locally detected events and spatial information are centrally correlated. The first subproblem of this step is to check if concurrent events of different end-users are caused by a network fault. There are several rea-

sons to different hosts identify simultaneous events not caused by the network. For example, a high volume of requests in a web service can impact the end-hosts service performance. Also, it is possible that simultaneous events occur only by chance, for instance, users can suffer signal interference on separate wireless routers. Therefore, through service specific dependencies and the empirical rate of simultaneous events, CEM provides a statistical model to determine if concurrent problems are a coincidence. In this model, the confidence of a network fault increases with the number of hosts that detect the event, and also with the number of affected metrics. The detailed method indicating how to realize spatial and temporal correlations to localize problems is not specified.

CEM was deployed and evaluated in a P2P system, using traces collected from users of the Ono plugin in the Vuze BitTorrent client. The system's output was contrasted with ISPs public available reports. In general, CEM provides a high level system abstraction, lacking several important deployment issues.

Chapter 3

Change Point Detection

A change point detection algorithm seeks to identify points in time where the statistical properties of a time series changes. This problem has a broad application in different knowledge areas, and in general, an algorithm's performance is closely related with the time series characteristics. Further, if the latent information of the procedures that generated the time series is missing, the target statistical properties can be considered subjective, bringing difficulties not only in the detection phase but also in the problem formalization.

In this context, this chapter studies the problem and briefly discusses several change point detection algorithms. The literature of this area is extensive, and it is common to find methods that presents a poor performance due to a variety of reasons, such as being too specific to the application area, or because the mechanisms were only analyzed through theoretical aspects. Therefore, it were selected a set of techniques with a good level of theoretical formalism, and flexibility to adapt, in order to handle specifities of the problem domain. Furthermore, this chapter exposes several challenges when dealing with real network measurements data, and some adopted solutions which are not described in the literature.

3.1 Problem Definition

The problem can be offline or online. In the offline version, to decide if a specific point at time t is a change point, the solver has available the whole time series, including past and future information w.r.t. t. On the other hand, in the online version, the information is available up to time t. The choice between these options is defined by the application domain. In some cases data are processed in real time, and change points should be detected as soon as possible. But in other applications changes are identified by historical purposes, and offline algorithms can be used.

It is intuitive that the offline case is more robust, since there is more information to analyze. In practice, to increase the statistical confidence of a decision, the online

definition is relaxed, and to decide if a point in time is a change point it is possible to use data up to a small window in the future, which in real time processing means that the application should wait until additional data is available. Hence, there is a trade-off between minimizing the time to detect a change and correctly classify a point. Therefore, in some cases, the online version can be transformed in offline by minor modifications.

In this work it is considered the following input and change points attributes, which were defined considering the final application scenario:

- Univariate time series. However, it is possible to extend several methods presented here to deal with multivariate data.
- Unevenly spaced time series, that is, data is not regularly sampled in time.
- Time series with different lengths.
- Unknown number of change points.
- Different number of points between change points.
- Focus on changes in the underlying mean and distribution, disregarding other kinds of changes, such as in periodicity.
- Outliers are not considered statistical changes.
- There is no latent information of the time series.
- It is considered the online and offline options.

3.2 Notation

An univariate time series composed of n points is defined by two vectors, $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$. The value y_i indicates the i-th sampled value, and x_i indicates the associated sample time. It is assumed that the points are sorted by time, that is, $x_{i-1} < x_i$ for $i = 2, \ldots, n$. Since unevenly spaced time series is considered, $x_i - x_{i-1}$ can be different for different i values. For $s \leq t$ the following notation is adopted: $\mathbf{y}_{s:t} = (y_s, \ldots, y_t)$.

The presence of k change points implies that data is split into $k + 1$ segments, also called windows. Let τ_i indicates the i-th change point for $i = 1, \ldots, k$. Also let $\tau_0 = 0$, $\tau_{k+1} = n$ and $\boldsymbol{\tau} = (\tau_0, \ldots, \tau_{k+1})$. Then, the i-th segment is defined by $\mathbf{y}_{\tau_{i-1}+1:\tau_i}$, assuming that $\tau_{i-1} < \tau_i$ for $i = 1, \ldots, k + 1$.

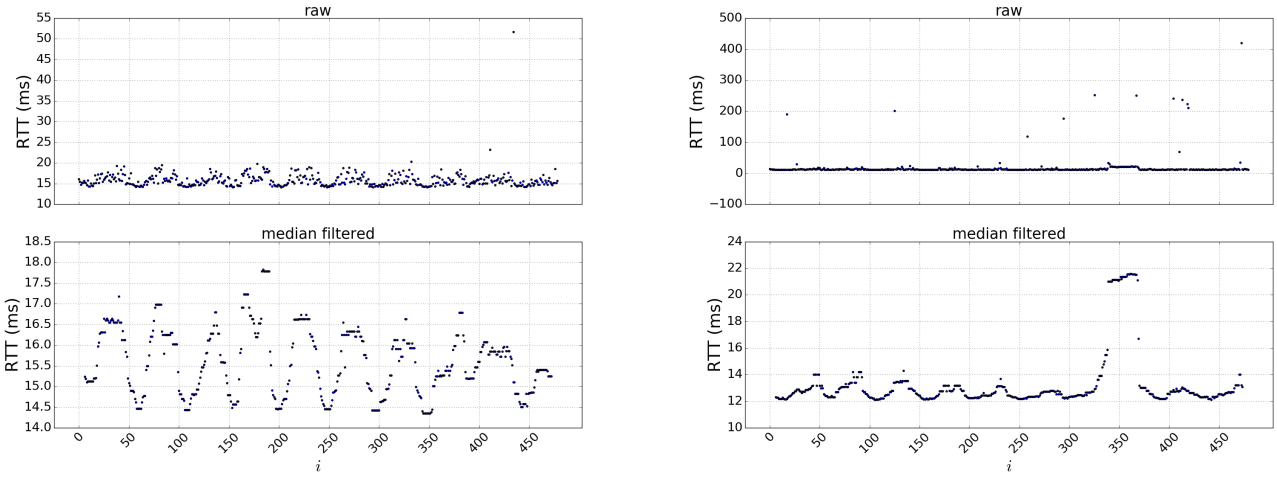
Through the previous definitions, change point detection algorithms mainly aim to find both k and $\boldsymbol{\tau}$.

3.3 Preprocessing

To reduce noise and remove outliers, the time series are preprocessed before being presented to a change point detection algorithm. Several filters were tested, such as moving averages, sliding windows percentiles, z-score, wavelet thresholding [5], optimal 1D clustering [6], and Savitzky-Golay filter [7].

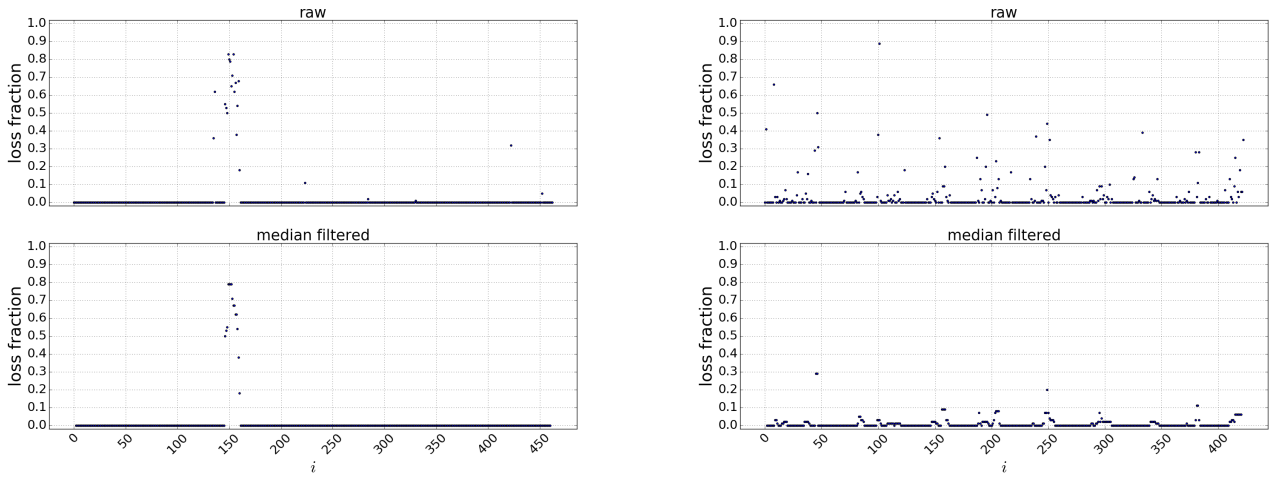
Since several time series are periodic, it was also considered the possibility of only using the trend, or the trend + residual transformation, resulted from the STL decomposition [22].

Due to it's simplicity, this chapter use time series only filtered by a centered median sliding windows, in which given a parameter w, y_i is set to be the median of the raw samples with indexes in $[i - w, i + w]$. Figures 3.1 and 3.2 show some examples of this preprocessing.



(a) (b)

Figure 3.1: Median filter RTT. $w = 6$.



(a) (b)

Figure 3.2: Median filter loss fraction. $w = 2$.

3.4 Sliding Windows

Sliding windows techniques use two sliding windows over the time series, and reduce the problem of detecting change points to the problem of testing whether data from the segments were generated by different distributions. One approach is to consider a distance metric between two empirical distributions as the base to infer the change points. Letting $d(\mathbf{a}, \mathbf{b})$ be the distance between two empirical distributions defined by the windows $\mathbf{a}$ and $\mathbf{b}$, and considering windows of length m, the Algorithm 1 presents a simple online sliding windows method.

Algorithm 1 Online Sliding Windows

```
1: i ← 1
2: while i + 2m − 1 ≤ n do
3:     if d(y_{i:i+m−1}, y_{i+m:i+2m−1}) > α then
4:         Report i + m − 1 as a change point
5:         i ← i + m
6:     else
7:         i ← i + 1
8:     end if
9: end while
```

The distance function has a direct impact on the classification accuracy. Therefore, several distance measures where tested, such as mean difference, relative mean difference, Hellinger [8], Kolmogorov-Smirnov, and Earth Mover's Distance [9].

Figure 3.3 is an example of the online sliding windows execution. The top plot

shows the RTT over time, and in the bottom plot, the (i, D_i) point represents the distance between $\mathbf{y}_{i-m:i-1}$ and $\mathbf{y}_{i:i+m-1}$. The red vertical line indicates a detected change point, and the green horizontal line illustrates the used threshold.

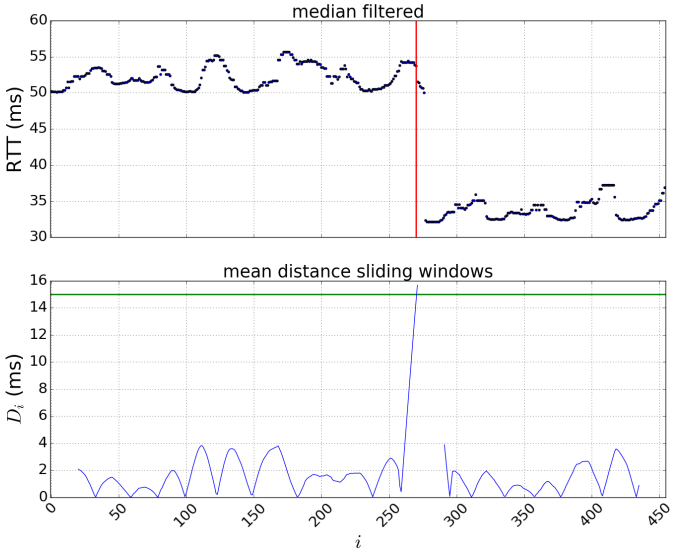


Figure 3.3: Online Sliding Windows.

It is possible to note that the detected change point is on the left of the correct location. This occurred since the distance between the windows was still increasing when reached the threshold. Therefore, for the offline version, it was defined that a peak detection method is applied on the sliding windows distance time series. An execution example is found in Figure 3.4.

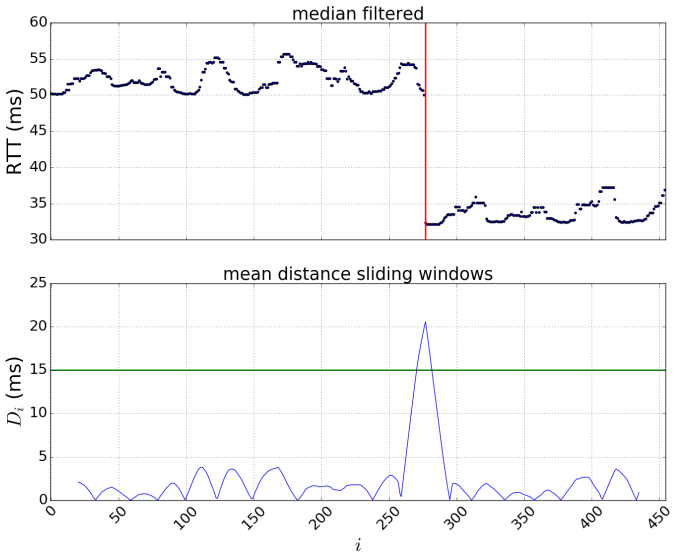


Figure 3.4: Offline Sliding Windows.

As stated in [10], a performance improvement can be achieved concurrently ex-

ecuting the same sliding windows algorithm with different windows lengths. This change facilitates the detection of segments with distinct number of points.

3.5 Optimization Model

Given a fixed value of k, one approach is to define a cost function that measures the homogeneity of a window, and therefore, choose the change points that globally optimize this homogeneity. Let the cost of the i-th segment be defined as $C(\mathbf{y}_{\tau_{i-1}+1:\tau_i})$, then the cost of a segmentation is the sum of all segments costs.

A common choice for the function C is the MSE (Mean Squared Error), which can capture changes in the mean. Another usual approach is to consider distribution changes through negative maximum log-likelihood functions, considering that data within a window is iid.

Therefore, given a fixed k, the optimal segmentation is obtained through the following optimization problem, which is called the constrained case [11]:

$$\min_{\boldsymbol{\tau}_{1:k}} \sum_{i=1}^{k+1} C(\mathbf{y}_{\tau_{i-1}+1:\tau_i}) \tag{3.1}$$

This problem can be solved using dynamic programming with $O(kn^2 f(n))$ time complexity, where $f(n)$ is related with the cost function evaluation. Several segment cost functions can be evaluated in $O(1)$ after a $O(n)$ preprocessing phase, implying in an overall $O(kn^2)$ complexity. It is possible to prove that MSE, negative maximum log-likelihood functions of normal, exponential, poisson and binomial distributions have this characteristic. Also, the formulation can consider a minimum value of a window length.

Modeling segments with distributions can lead to practical difficulties. One of them is the fact that segments can form degenerate distributions, that is, the data of a window can have zero variance, which is always the case of unitary length windows. In these scenarios the negative maximum log-likelihood can be undefined. Despite this issue has not been described in the literature, this work followed two approaches to overcome this situation. The first one tries to avoid degenerate segments adding a white noise with small variance to the data stream. The second one considers that the cost of any degenerate distribution is equal to a constant.

When the number of change points is unknown, an usual way is to introduce a non decreasing penalty function $g(k)$. Then, the new optimization problem, called penalized case [11], is:

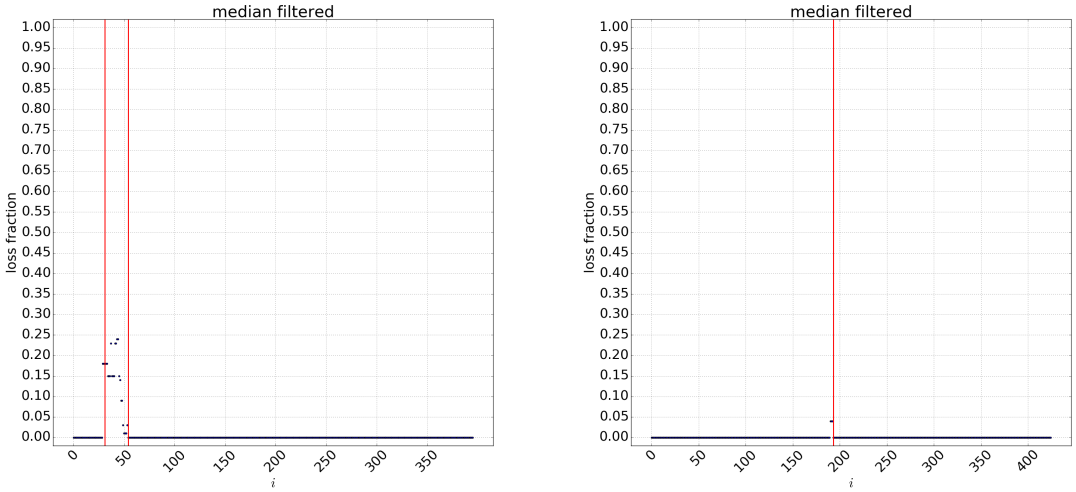
$$\min_{k,\boldsymbol{\tau}_{1:k}} \sum_{i=1}^{k+1} C(\mathbf{y}_{\tau_{i-1}+1:\tau_i}) + g(k) \tag{3.2}$$

This problem can be solved in $O(Kn^2 f(n))$. However, if the penalty function is linear in k, the problem can be formulated more efficiently and solved in $O(n^2 f(n))$.

Also, there are several pruning algorithms to speedup the computation [11–13], in general trying to reduce the $\boldsymbol{\tau}$ search space but maintaining optimality.

When a negative maximum log-likelihood is used, the cost of a segment with an outlier can be orders of magnitude greater than the cost of a window without anomalies. Therefore, in this case, the method is sensible to outliers.

Figure 3.5 presents two output examples.



(a) (b)

Figure 3.5: Optimization model.

3.6 HMM (Hidden Markov Model)

The idea that each segment is associated with a specific latent configuration has a direct interpretation to a HMM model [14–16]. In this context, each window is related to a hidden state of a HMM, and the observation distribution of this state represents the distribution of that segment. Therefore, the mechanism models the time series using a HMM, and through the hidden state path, assesses the times when a transition between different hidden states occur.

There are several approaches in the detection and training phases. For example, given a trained HMM, the most probable hidden state path can be checked through the Viterbi algorithm. Also, it is possible to evaluate the probability of a transition between different hidden states at time t, and then apply a threshold and peak detection methods, as well as in sliding windows techniques. For the training step, it is possible to use several time series to train a single HMM, and then use this model to detect change points in all time series. Another way is to, for each data

stream, train a single model using only the target time series.

It is important to note that the structure of the hidden state graph has a large impact on the performance. Using a fully connected graph, the number of states defines the maximum number of distribution configurations. Employing a left to right structure, the number of hidden states will impact the maximum number of segments.

In [16] is stated that when using a fully connected structure, the time interval that a time series stays in the same hidden state is low, which can not reflect real data. To overcome this problem, [16] suggests to increase the time that a time series stands in the same hidden state using a dirichlet prior regularization. However, it was empirically verified that it is difficult to choose good parameters for this strategy. Instead, to surpass this issue, when using the best hidden state path, this dissertation used a HMM only as a filter, which acts as a dimensional reduction that takes into consideration temporal patterns. In this scenario, the best hidden state sequence is the input to a sliding windows method with a discrete Hellinger distance.

Figure 3.6 presents an execution using a fully connected HMM with observations following a Normal distribution. The top plot is the median filtered time series. The middle plot is the best hidden state path, in which the vertical axis indicates the distribution of each hidden state. The bottom plot is the sliding windows distance.

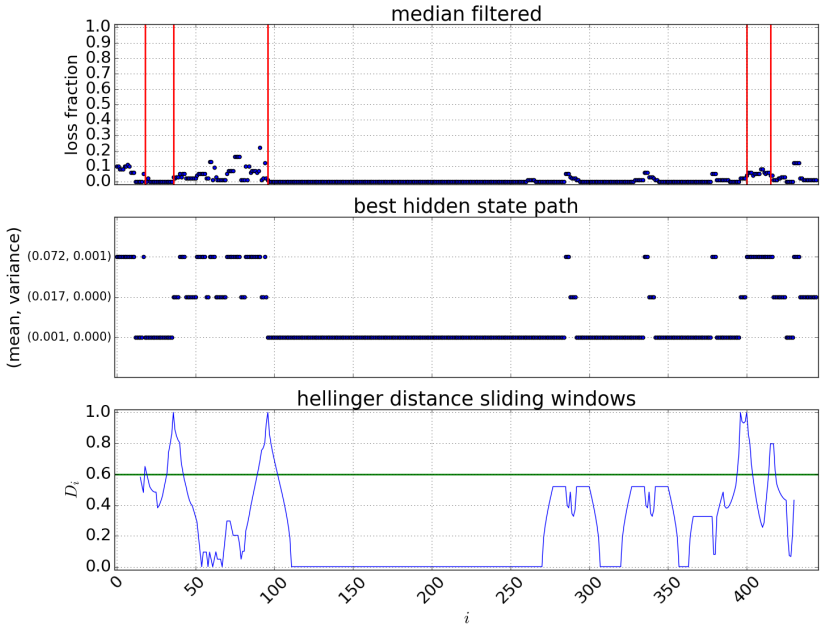


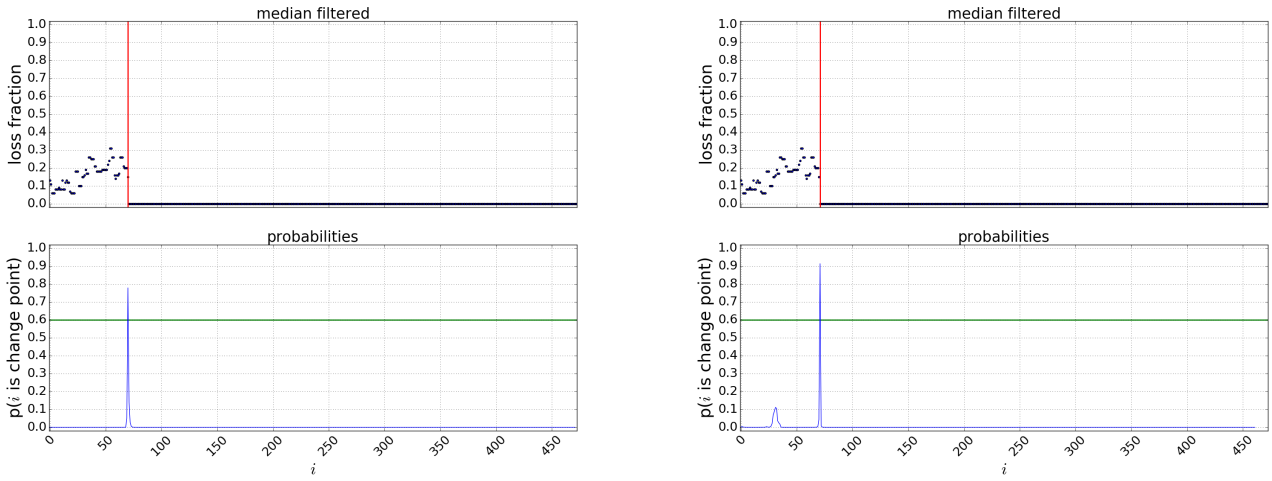
Figure 3.6: HMM filter.

3.7 Bayesian Inference

There are several Bayesian methods which aims to assess the probability that a point is a change point. Following an offline fashion, the work of [17] recursively calculates, for each i, the probability of $\mathbf{y}_{i:n}$ given a change point at i. With these probabilities is possible to simulate the time of the first change point, and then, compute the conditional distribution of the time of the second change given the first, and so on. To achieve this, the mechanism assumes that observations are independents, and that each segment is modeled by conjugate priors. Also, the procedure considers priors to model the number of changes and the time between two consecutive change points. Depending of the priors choices, the overall complexity of this method can be $O(n^2)$.

In [18] it is also considered that parameters of different segments are independents, and that data within a window is iid. However, through an online mode, the procedure is concerned with the estimation of the distribution of the length of the current time since the last change point, called run length, given the data so far observed. To achieve this, the method assumes the probability of current run length given the last run length as a prior. Assuming exponential-family likelihoods to model a segment, the time complexity to process a point is linear in the number of points already observed.

As with the previous procedures, in both cases is applied a peak detection algorithm in the probability time series. Also, these methods can be sensible to outliers, specially the online version. Furthermore, it was empirically noticed that, in general, the probabilities are only non zero around the probability peaks. Figure 3.7 illustrates the bayesian inference algorithms executions.



(a) Offline.

(b) Online.

Figure 3.7: Bayesian Inference.

3.8 Final Remarks

Despite not being explicitly approached, it is possible to observe through this chapter that network measurements data has different patterns. Therefore, the choice of the most appropriate set of algorithms and parameters is one of the main difficulties of this project. In Chapter 4 this issue is discussed with more details.

The description of the parameters used in this chapter can be found in appendix **??**.

Chapter 4

Methodology

4.1 Measurement Process

The exposed time series of this work represent network end-to-end measures of a cable-television infrastructure, which runs DOCSIS with asymmetric download and upload bandwidths. Home routers connected to the cable modem communicate with one or more servers strategically located by the ISP. Measurements results from each home router are consolidated every half hour and, by the end of every day, are transferred to a database. The software responsible for these procedures was developed by TGR (a startup at COPPE/UFRJ) in collaboration with the university (UFRJ), and is spread over a customers subset of a major Brazilian ISP.

In [19], a preliminary investigation of several metrics is presented. To measure the round trip packet loss fraction and RTT between the home router and the associated server, the home router sends a train of short UDP packets, and then the server bounces back them. The data here presented considers a train of 100 UDP packets of 32 bytes, separated by 1 millisecond. This dissertation only deals with these two metrics.

The resulted time series are unevenly spaced due to a range of reasons. First, measurements are initiated only if the residential link is not under use by the ISP customer. Also, the client may have no Internet connection to start a measurement, or even be without electrical energy.

4.2 Supervised Learning Try

As stated in Chapter 3, one of the main issues of this work is to select the algorithms and parameters. In this context, it is essential to be able to evaluate the performance of a change point detection algorithm set up, which can be a challenging task when handling real data. In literature are several approaches to construct a change points

dataset.

It is important to note that the algorithms presented in Chapter 3 doesn't use a previous point labels for a training phase. Once a change point dataset is constructed, in which a train set each labels each point of a time series as a change point or not, can open new possibilities to a supervised learning procedure, in which are not much explored in the literature of change pointe detection.

Some works create simulated time series, in which distinct segments are sampled by the same generative model with different parameters [24]. In general, this type of data is more easily handled by change point detection algorithms, since some methods assume the same models used in the dataset building process. Also, real data can have complex characteristics that are difficult to be reproduced by generative models. Another strategy is to join segments from different real time series [16]. However, this can introduce unreal change points scenarios, and also it must be guaranted that the joined time series are statistically different.

When the latent information of the time series are available, and if there is a complete knowledge of what configurations changes in the latent state impact data, it is possible to check the change points only analyzing this underlying information. As an example, consider a time series that represents the cardiac frequency of a soccer player during a match. Also, consider that in this controlled environment, the only causes of changes in the cardiac frequency are the variations of physical activities, such as starting or stopping to run. Therefore, it is possible to use the times in which a player changed his movement behavior as the change points, whithout even analyzing the time series. However, in the application domain of the present work, this approach would be impractical. First, this would need the expertise of how the configurations of network topology, routers congestion, physical equipment problems, among other features, affect the different end-to-end QoS metrics. Second, this kind of information is absent in the dataset, and would be too complex to collect it.

The approach followed in this work was to use visual classifications, as it was done in [25]. An application domain expert was exposed to a set of time series, and visually indicated his opinion about the change points locations. It is known that visual inspection methods can bring erroneous conclusions [23], and also amplify subjectivity, however, it was the best alternative considering the data availability and the objective of working with real data.

Through a web system the user freely marked the change points with a mouse. The fact that data is not regularly sampled in time could bring an unwanted visual change perception. Therefore, the X axis of the displayed time series represented only the temporal order of the measures.

A single specialist classified all time series. This person has experience with

network measurements and statistical modeling, however, without background in change point detection. The user could take any time to make a classification, and it was able to access the system in different days. Additionally, it was provided a set of tips to the specialist:

- In the case of packet loss fraction, mean changes between 0 and 0.1 are more sensible to the end users.
- The time axis only represents the temporal order of the measurements. However, in general, consecutive points in time axis are separated by 30 minutes.
- Outlier is not a statistical change. An outlier is an observation that lies outside the overall pattern of a distribution.

To provide a clear visualization, the change points dataset was composed by time series with 10 days of data. Therefore, each time series of the previous dataset was split in two, one from 01/may/2016 to 10/may/2016, and other from 11/may/2016 to 20/may/2016. Also, it was selected only the ones that have at least 85% of the maximum possible number of points during the specified period, considering that data is sampled at most two times in a hour. Change points can be interpreted as rare events in this dataset, and most data streams have almost all measures with zero losses. Therefore, to increase the dataset entropy, it was only selected time series that have at least one window of length 48 with more than 5 measures with loss fraction larger than 0.01. These filters resulted in 522 time series.

Figure 4.1 presents a system snapshot. The vertical red line means that the user marked a change point in that position.

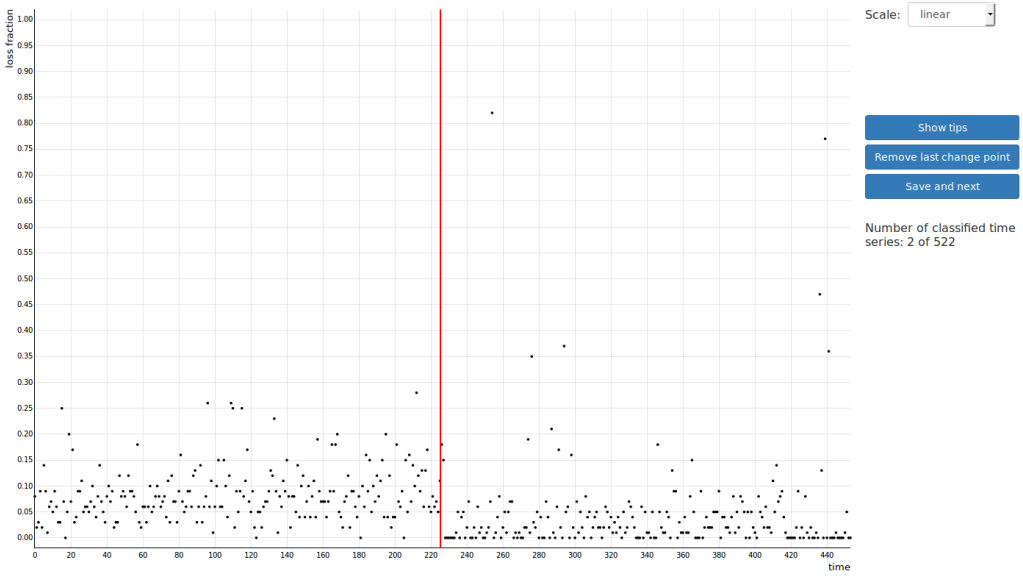


Figure 4.1: Survey system snapshot.

4.3 Pipeline

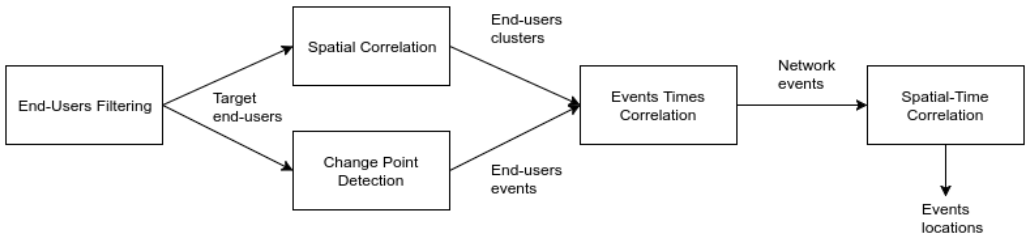


Figure 4.2: Pipeline.

4.4 Spatial Aggregation

4.5 Change Point Detection Parameters Tuning

4.6 Time Aggregation

4.7 Spatial/Time Correlation

4.8 Differences to Previous Systems

Chapter 5

Results

Chapter 6

Conclusions

Bibliography

[1] YAN, H., FLAVEL, A., GE, Z., et al. "Argus: End-to-end service anomaly detection and localization from an ISP's point of view". In: *2012 Proceedings IEEE INFOCOM*, pp. 2756–2760, March 2012. doi: 10.1109/INFCOM.2012.6195694.

[2] ADAMS, A., LAPUKHOV, P., ZENG, J. H. "NetNORAD: Troubleshooting networks via end-to-end probing". 2016. https://code.facebook.com/posts/1534350660228025/netnorad-troubleshooting-networks-via-end-to-end-probing/.

[3] CHOFFNES, D. R., BUSTAMANTE, F. E., GE, Z. "Crowdsourcing Service-level Network Event Monitoring", *SIGCOMM Comput. Commun. Rev.*, v. 40, n. 4, pp. 387–398, ago. 2010. ISSN: 0146-4833. doi: 10.1145/1851275.1851228. Disponível em: <http://doi.acm.org/10.1145/1851275.1851228>.

[4] GERBER, A., PANG, J., SPATSCHECK, O., et al. "Speed Testing Without Speed Tests: Estimating Achievable Download Speed from Passive Measurements". In: *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pp. 424–430, New York, NY, USA, 2010. ACM. ISBN: 978-1-4503-0483-2. doi: 10.1145/1879141.1879196. Disponível em: <http://doi.acm.org/10.1145/1879141.1879196>.

[5] GRAPS, A. "An introduction to wavelets", *IEEE computational science and engineering*, v. 2, n. 2, pp. 50–61, 1995.

[6] WANG, H., SONG, M. "Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming", *The R Journal*, v. 3, n. 2, pp. 29–33, 2011.

[7] WIKIPEDIA. "SavitzkyGolay filter". 2016. https://en.wikipedia.org/wiki/Savitzky-Golay_filter.

[8] WIKIPEDIA. "Hellinger distance". 2016. https://en.wikipedia.org/wiki/Hellinger_distance/.

[9] RUBNER, Y., TOMASI, C., GUIBAS, L. J. "The earth mover's distance as a metric for image retrieval", *International journal of computer vision*, v. 40, n. 2, pp. 99–121, 2000.

[10] KIFER, D., BEN-DAVID, S., GEHRKE, J. "Detecting Change in Data Streams". In: *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pp. 180–191. VLDB Endowment, 2004. ISBN: 0-12-088469-0. Disponível em: <http://dl.acm.org/citation.cfm?id=1316689.1316707>.

[11] MAIDSTONE, R., HOCKING, T., RIGAILL, G., et al. "On optimal multiple changepoint algorithms for large data", *Statistics and Computing*, pp. 1–15, 2016.

[12] KILLICK, R., FEARNHEAD, P., ECKLEY, I. "Optimal detection of changepoints with a linear computational cost", *Journal of the American Statistical Association*, v. 107, n. 500, pp. 1590–1598, 2012.

[13] HAYNES, K., ECKLEY, I. A., FEARNHEAD, P. "Computationally Efficient Changepoint Detection for a Range of Penalties", *Journal of Computational and Graphical Statistics*, v. 0, n. ja, pp. 1–28, 0. doi: 10.1080/10618600.2015.1116445. Disponível em: <http://dx.doi.org/10.1080/10618600.2015.1116445>.

[14] KEHAGIAS, A. "A hidden Markov model segmentation procedure for hydrological and environmental time series", *Stochastic Environmental Research and Risk Assessment*, v. 18, n. 2, pp. 117–130, 2004. ISSN: 1436-3259. doi: 10.1007/s00477-003-0145-5. Disponível em: <http://dx.doi.org/10.1007/s00477-003-0145-5>.

[15] LUONG, T. M., ROZENHOLC, Y., NUEL, G. "Fast estimation of posterior probabilities in change-point analysis through a constrained hidden Markov model", *Computational Statistics and Data Analysis*, v. 68, pp. 129 – 140, 2013. ISSN: 0167-9473. doi: http://dx.doi.org/10.1016/j.csda.2013.06.020. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167947313002326>.

[16] MONTAÑEZ, G. D., AMIZADEH, S., LAPTEV, N. "Inertial Hidden Markov Models: Modeling Change in Multivariate Time Series". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 1819–1825. AAAI Press, 2015. ISBN: 0-262-51129-0. Disponível em: <http://dl.acm.org/citation.cfm?id=2886521.2886573>.

[17] FEARNHEAD, P. "Exact and efficient Bayesian inference for multiple changepoint problems", *Statistics and Computing*, v. 16, n. 2, pp. 203–213, 2006. doi: 10.1007/s11222-006-8450-8. Disponível em: <http://dx.doi.org/10.1007/s11222-006-8450-8>.

[18] ADAMS, R. P., MACKAY, D. J. "Bayesian online changepoint detection", *arXiv preprint arXiv:0710.3742*, 2007.

[19] MENDES, D. X., SENGES, G. D. S., SANTOS, G. H. A. D., et al. "A Preliminary Performance Measurement Study of Residential Broadband Services in Brazil". In: *Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks*, LANCOMM '16, pp. 16–18, New York, NY, USA, 2016. ACM. ISBN: 978-1-4503-4426-5. doi: 2940116.2940135. Disponível em: <http://doi.acm.org/2940116.2940135>.

[20] MAHIMKAR, A., YATES, J., ZHANG, Y., et al. "Troubleshooting Chronic Conditions in Large IP Networks". In: *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pp. 2:1–2:12, New York, NY, USA, 2008. ACM. ISBN: 978-1-60558-210-8. doi: 10.1145/1544012.1544014. Disponível em: <http://doi.acm.org/10.1145/1544012.1544014>.

[21] AUGER, I. E., LAWRENCE, C. E. "Algorithms for the optimal identification of segment neighborhoods", *Bulletin of Mathematical Biology*, v. 51, n. 1, pp. 39–54, 1989. ISSN: 1522-9602. doi: 10.1007/BF02458835. Disponível em: <http://dx.doi.org/10.1007/BF02458835>.

[22] CLEVELAND, R. B., CLEVELAND, W. S., MCRAE, J. E., et al. "STL: A seasonal-trend decomposition procedure based on loess", *Journal of Official Statistics*, v. 6, n. 1, pp. 3–73, 1990.

[23] JAMES, N. A., KEJARIWAL, A., MATTESON, D. S. "Leveraging Cloud Data to Mitigate User Experience from" Breaking Bad"", *arXiv preprint arXiv:1411.7955*, 2014.

[24] LIU, S., YAMADA, M., COLLIER, N., et al. "Change-point detection in time-series data by relative density-ratio estimation", *Neural Networks*, v. 43, pp. 72 – 83, 2013. ISSN: 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2013.01.012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608013000270>.

[25] HOCKING, T., RIGAILL, G., PHILIPPE VERT, J., et al. "Learning Sparse Penalties for Change-point Detection using Max Margin Interval Regres-

sion". In: Dasgupta, S., Mcallester, D. (Eds.), *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, v. 28, pp. 172–180. JMLR Workshop and Conference Proceedings, maio 2013. Disponível em: <http://jmlr.org/proceedings/papers/v28/hocking13.pdf>.