DETECTION AND LOCALIZATION OF EVENTS IN COMPUTER NETWORKS USING END-TO-END MEASUREMENTS TIME SERIES

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
Outubro de 2017

DETECTION AND LOCALIZATION OF EVENTS IN COMPUTER NETWORKS USING END-TO-END MEASUREMENTS TIME SERIES

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:

Prof. Edmundo Albuquerque de Souza e Silva, Ph.D.

Prof. Antonio Jorge Gomes Abelém, D.Sc.

Prof. Daniel Sadoc Menasche, Ph.D.

Prof. Pedro Braconnot Velloso, Dr.

RIO DE JANEIRO, RJ – BRASIL
OUTUBRO DE 2017

Mendes, Diego Ximenes

Detection and Localization of Events in Computer Networks Using End-to-End Measurements Time Series/Diego Ximenes Mendes. – Rio de Janeiro: UFRJ/COPPE, 2017.

IX, 48 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. 45 – 48.

1. Time Series. 2. Change Point Detection. 3. Machine Learning. 4. Network Measurements. 5. Network Troubleshooting. I. Silva, Edmundo Albuquerque de Souza e. 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.)

DETECÇÃO E LOCALIZAÇÃO DE EVENTOS EM REDES DE COMPUTADORES UTILIZANDO SÉRIES TEMPORAIS DE MEDIÇÕES FIM-A-FIM.

Diego Ximenes Mendes

Outubro/2017

Orientador: Edmundo Albuquerque de Souza e Silva

Programa: Engenharia de Sistemas e Computação

Com o objetivo de melhor entender o desempenho de sua rede, um grande tier-3 ISP brasileiro, em parceria com a UFRJ e uma startup incubada nessa universidade, estabeleceram um projeto para, através de medidas fim-a-fim de QoS, monitorar o serviço prestado em um subconjunto dos seus clientes.

Nesse contexto, guiada pelas características da rede do ISP, e do atual processo de medição, esta dissertação se propõe a avaliar a viabilidade de apenas utilizar métricas fim-a-fim de QoS, e traceroutes, para identificar e localizar eventos de rede. Um evento pode ser interpretado como uma mudança no comportamento de um equipamento, que por sua vez afeta a qualidade de serviço percebida pelos usuários, como por exemplo, um defeito em um roteador. O procedimento de localização define um conjunto de possíveis locais onde o evento pode ter acontecido.

Com esse propósito, este trabalho propõe um framework de análise de dados, que permite explorar mudanças estatísticas nas séries temporais de QoS de diferentes clientes. Para detectar e localizar eventos, o mecanismo correlaciona esses padrões de alteração com traceroutes. A fim de aumentar o desempenho do sistema proposto, esta dissertação também indica possíveis melhorias na atual metodologia de medição.

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

DETECTION AND LOCALIZATION OF EVENTS IN COMPUTER NETWORKS USING END-TO-END MEASUREMENTS TIME SERIES

Diego Ximenes Mendes

October/2017

Advisor: Edmundo Albuquerque de Souza e Silva

Department: Systems Engineering and Computer Science

To better understand the performance of its own network, a major tier-3 Brazilian ISP, in partnership with UFRJ and a startup incubated at this university, established a project to monitor the service provided to a subset of its customers.

In this context, guided by the specific ISP's network's characteristics, and the current measurement process, this dissertation aims to check the viability of only using end-to-end QoS measures, and traceroutes, to identify and localize network events. An event can be interpreted as a behavioral change in a network equipment, that affect the quality of service perceived by the end-users, such as a router failure. The localization procedure defines a set of feasible locations where the event could have happened.

For such purpose, this work proposes a data analytics framework, which is able to track statistical changes in the QoS time series of different clients. To detect and localize events, the mechanism correlates these modification patterns with traceroutes. In order to increase the system's performance, this dissertation also indicates possible improvements in the current measurement methodology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To better understand the performance of its own network, a major tier-3 Brazilian Internet Service Provider (ISP), in partnership with the Federal University of Rio de Janeiro (UFRJ) and a startup incubated at this university, established a project to monitor the service provided to a subset of its customers.

Considering the ISP's cable-television infrastructure, which runs Data Over Cable Service Interface Specification (DOCSIS), Quality of Service (QoS) metrics are gathered through a software placed at home routers connected to cable modems. This software is responsible to perform end-to-end measurements against servers strategically located by the ISP.

In this context, guided by the specific ISP's network's characteristics, and the current measurement process, this dissertation aims to check the viability of only using end-to-end QoS measures, and traceroutes, to identify and localize network events. An event can be interpreted as a behavioral change in a network equipment, that affects the quality of service perceived by the end-users, such as a router failure. The localization procedure defines a set of feasible locations where the event could have happened.

For such purpose, this work proposes a data analytics framework, which is able to track statistical changes in the QoS time series of different clients. To detect and localize events, the mechanism correlates these modification patterns with traceroutes. In order to increase the system's performance, this dissertation also indicates possible improvements in the current measurement methodology.

1.1 Contributions

Considering the specific ISP's network topology, and the already implemented measurement process, next is listed this dissertation's main contributions.

- A data analytics procedure, that only uses the available end-to-end QoS mea-

surements, and traceroutes, to detect and localize network events.

- A list of possible improvements in the measurement methodology currently employed by the startup, in order to enhance the proposed system's performance.

1.2 Dissertation Outline

Chapter 2 consists of a literature review, that describes three previous systems that use end-to-end measurements to localize network events. To track statistical changes in the time series, the current work deploys change point detection methods. Then, Chapter 3 presents several of these algorithms, including the developed strategies to better handle the end-to-end QoS data characteristics. In Chapter 4 is detailed the measurement methodology and the proposed data analytics workflow. Chapter 5 presents several results when the proposed pipeline was applied to real data. Finally, Chapter 6 concludes the work and points future directions.

The code developed to support this dissertation is available at `https://bitbucket.org/diegoximenes/master_thesis`. Unfortunately, due to a Non-Disclosure Agreement (NDA), the data used in this project can not be open sourced.

Chapter 2

Literature Review

This chapter briefly describes three projects, Argus [1], NetNorad [2], and Crowdsourcing Event Monitoring (CEM) [3], that use end-to-end QoS to identify and localize events, or in some cases only faults, in computer networks.

These previous works share common approaches. For instance, Argus and NetNorad track the end-to-end QoS evolution over time of different end-hosts, and the anomalies identified in these time series are interpreted as events. Anomalies are defined as the time points when the data stream deviates from its standard. Besides this similarity, considering that these projects where designed to be deployed in different network architectures, they differ in several aspects, such as in the QoS data collection, anomaly detection, and event localization procedures.

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 data passively collected 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.

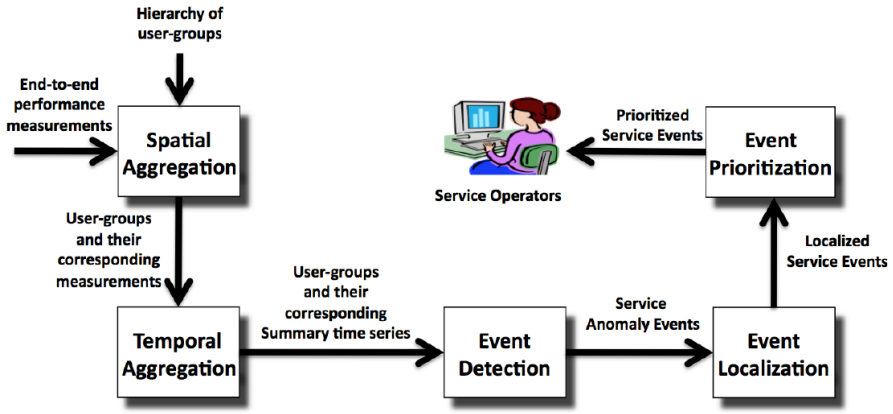


Figure 2.1: Argus' pipeline. [1]

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

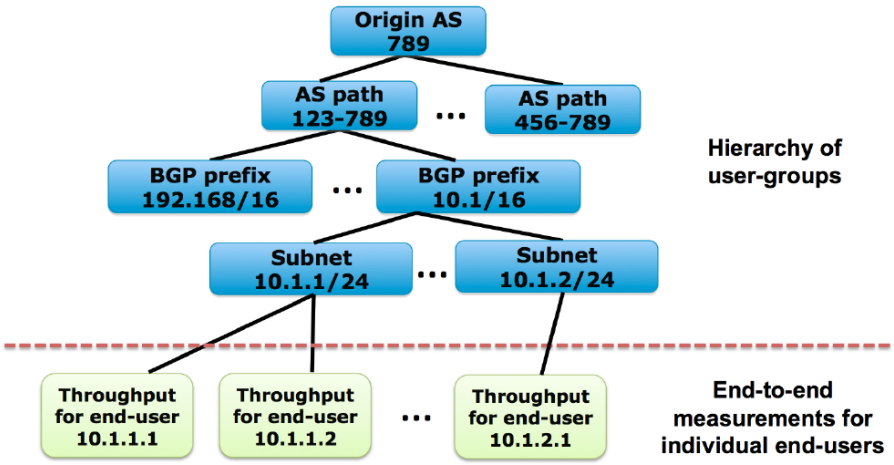


Figure 2.2: Argus' spatial aggregation. [1]

The Temporal Aggregation phase determines how data from different clients of an end-group are combined. For each user-group, the measurements samples of all end-users are grouped in time-bins, and for each time-bin, a 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 Round Trip Times (RTTs) can capture the physical propagation delay, while the average can be related with network congestion. Argus uses median by default, since it was empirically verified as an effective transformation 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 time series obtained in the Temporal Aggregation mechanism. Argus uses a Holt-Winters [5] variation.

Using spatial and events' times correlations, the Event Localization step infer fault locations. 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 customers.

Argus was evaluated using RTT measurements of a Content Delivery Network (CDN) hosted in a tier-1 ISP. During one month, 2909 anomalous events were detected. In general, lower level user-groups were more responsible for the events than the higher level groups, and only a small fraction of the customers caused the user-groups anomalies. 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 clients are responsible for the user-groups events, is an indication that fault localization can achieve higher precision with finer spatial aggregation granularity. Besides, the system's accuracy was not studied.

2.2 NetNorad

NetNorad [2] consists of a Facebook's internal project to automate its network's faults analysis. Previous deployed techniques by Facebook exhibit several disadvantages, for instance, human-driven investigation may take hours. Also, cases known as gray failures, can't be detected only collecting devices information through Simple Network Management Protocol (SNMP), or command line interface. For example, some equipments can't report its 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 then 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 them to the Facebook's global backbone. Figure 2.3 illustrates this architecture.

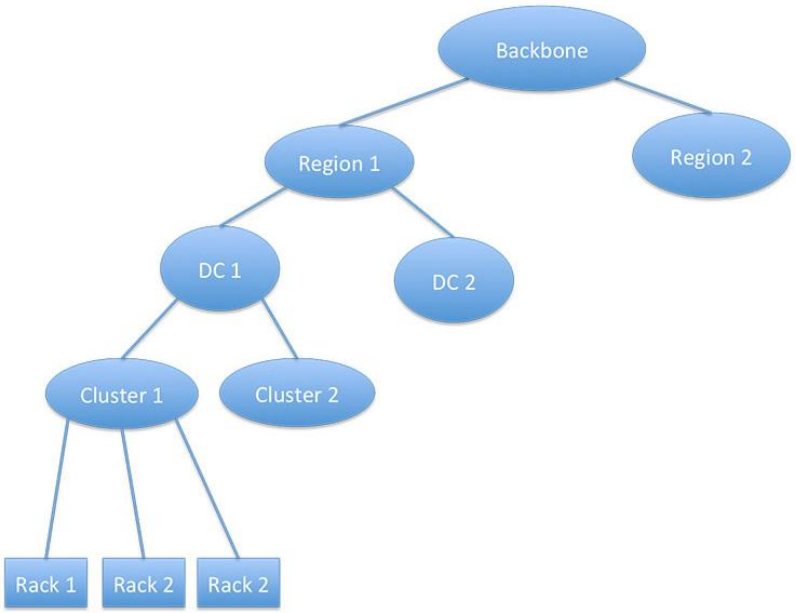


Figure 2.3: Facebook's network architecture. [2]

Unlike Argus, NetNorad uses active probing to assess loss and RTT statistics. Facebook's servers ping each other, in which a pinger sends User Datagram Protocol (UDP) packets to responders, and the latter sends 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, 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 has three time series reflecting different spatial viewpoints. To mitigate problems, these data streams are tracked through distinct percentiles over 10 minutes intervals. 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, it is defined two thresholds, one to trigger and another to 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 CEM, in which a monitoring software, that runs inside or alongside applications, is placed at 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 Video on Demand (VoD) application. To increase the system's scalability, each end-host identifies local problems by their own, and pushes these information to a distributed storage to further analysis. The framework doesn't specify 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 reasons to different hosts identify simultaneous events not caused by the network. For instance, 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, as an example, users can suffer signal interference on distinct 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 change. This problem has a broad application in different knowledge areas, and in general, an algorithm's performance is closely related with the data 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.

The choice of detecting events through change points was motivated during a visual inspection of the time series. It was noticed that, statistical changes, such as in the underlying distribution, persist for long time periods. In contrast, Argus applies an anomaly detection procedure. The difference between these two problems is subtle, and can be fuzzy in the literature. The anomaly detection assumes that a standard pattern is already known or is identified by the procedure, then, the goal is to find when the data stream deviates from its standard. The change point detection only seeks for points where the statistical properties change, and doesn't take into consideration a standard time series behavior.

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 present 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 was selected a set of techniques with a good level of theoretical formalism, and flexibility to adapt, in order to handle specificities 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 treated in an offline or online fashion. 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 is 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 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 maximize the correct classification of 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. This is an important feature since, as is going to be presented in Chapter 4, the measurement software is able to simultaneously collect more than one QoS metric.

- Unevenly spaced time series. As is going to be explained in Chapter 4, the QoS data is not regularly sampled in time. However, in general, two consecutive points are approximately 30 minutes apart.

- Unknown number of change points, since an arbitrary number of events can occur in a specific time period.

- Different number of points between change points, since events can persist for an arbitrary period of time.

- Focus on changes in the underlying mean and distribution, disregarding other kinds of changes, such as in seasonality. As stated in [6], an ongoing challenge for the change point detection problem is to handle time series with non-stationary segments. However, as it will be visually noticed through this chapter, and as it was presented in [7], even without change points, the time series used in this work are non-stationary.

- There is no latent information of the time series. Also, there isn't a change points' ground truth. Therefore, it is only possible to use unsupervised change point detection methods.
- Outliers are not considered statistical changes.
- 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 distinct for different i values. The following notation is also adopted: $\mathbf{y}_{s:t} = (y_s, \ldots, y_t)$ for $s \leq t$.

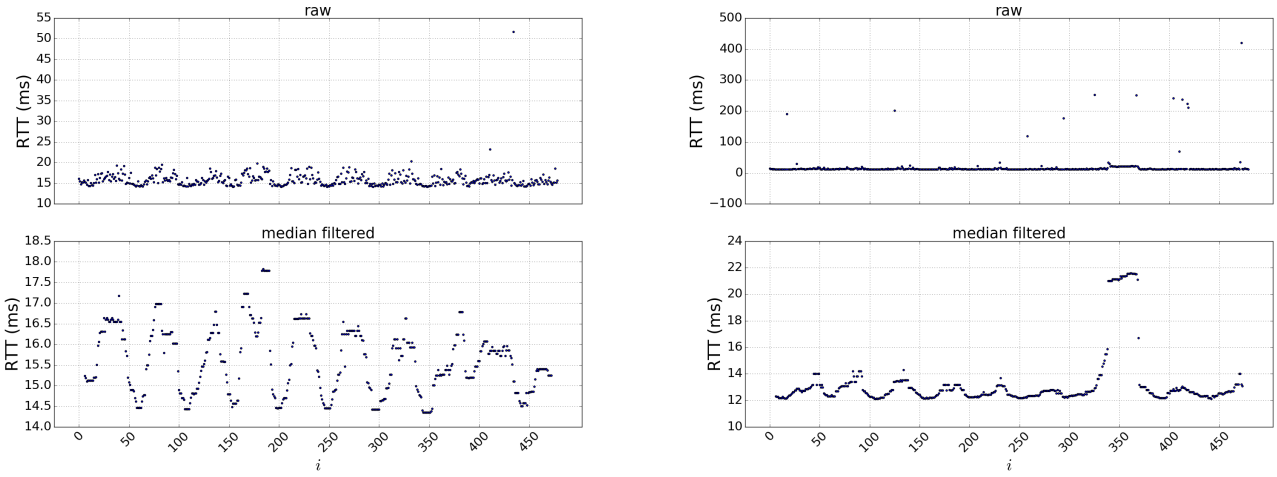
The presence of k change points implies that data is split into $k + 1$ segments, also called windows. Let τ_i indicates the i-th change point for $i = 1, \ldots, k$. Also let $\tau_0 = 0$, $\tau_{k+1} = n$, and $\boldsymbol{\tau} = (\tau_0, \ldots, \tau_{k+1})$. Then, the i-th segment is defined by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

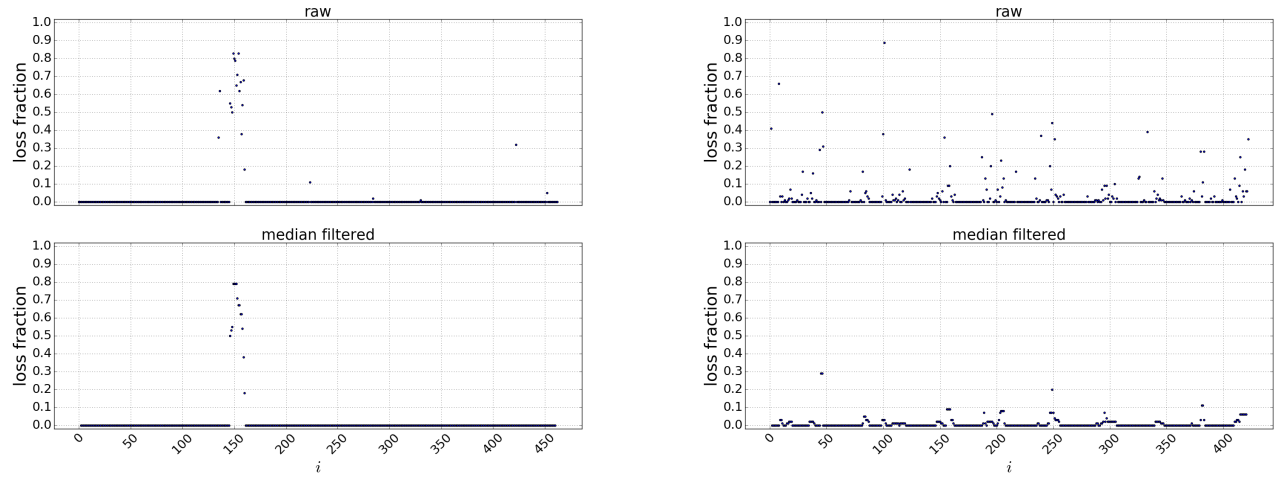
To reduce noise and remove outliers, the time series are preprocessed before being presented to a change point detection algorithm. Several filters were tested, such as moving averages, sliding windows percentiles, z-score, wavelet thresholding [8], optimal 1D clustering [9], and Savitzky-Golay filter [10]. Also, since several time series exhibit a seasonal pattern, it was also considered the possibility of only using the trend, or the trend + residual transformation, resulted from the STL decomposition [11].

This chapter uses 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 $[i - w, i + w]$. The choice of this method among the others was guided by two reasons. First, it was empirically verified its effectiveness in removing outliers. Second, its simplicity allows to easily interpret the impact of its parameter, which is an important feature to manually fine tune the procedure for the different QoS metrics. Figures 3.1 and 3.2 show some examples of this preprocessing.



(a) (b)

Figure 3.1: Median filter RTT. $w = 6$. Plots with different scales.



(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 $d(\mathbf{a}, \mathbf{b})$ to be the distance between two empirical distributions defined by the windows $\mathbf{a}$ and $\mathbf{b}$, and considering windows of length m, Algorithm 1 presents a simple online sliding windows method [12].

Algorithm 1 Online Sliding Windows

$i \leftarrow 1$
while $i + 2m - 1 \leq n$ **do**
    if $d(\mathbf{y}_{i:i+m-1}, \mathbf{y}_{i+m:i+2m-1}) > \alpha$ **then**
        Report $i + m - 1$ as a change point
        $i \leftarrow i + m$
    else
        $i \leftarrow i + 1$
    end if
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 [13], Kolmogorov-Smirnov, and Earth Mover's Distance [14].

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 α.

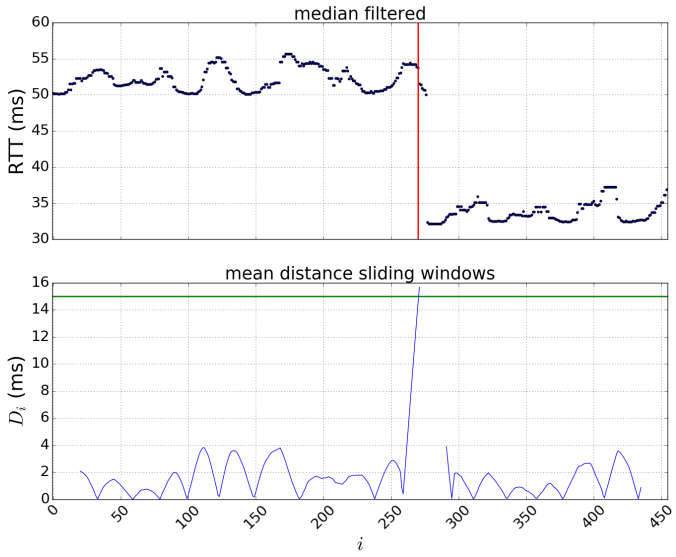


Figure 3.3: Online Sliding Windows.

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.

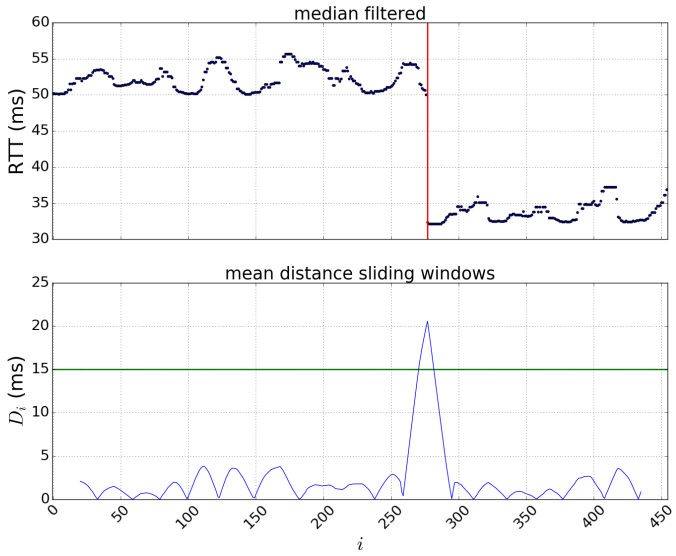


Figure 3.4: Offline Sliding Windows.

As stated in [12], a performance improvement can be achieved concurrently executing the same sliding windows algorithm with different windows lengths. This adaptation facilitates the detection of segments with distinct number of points.

3.5 Optimization Model

Given a fixed number of windows (segments), 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 Mean Squared Error (MSE), 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 [15]:

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

This problem can be solved using dynamic programming with $O(kn^2 f(n))$ time complexity, where $f(n)$ is a function of the cost to evaluate C. Several segment cost functions can be evaluated in $O(1)$ after a $O(n)$ preprocessing phase, implying 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. Although this issue has not been addressed 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 [15], is:

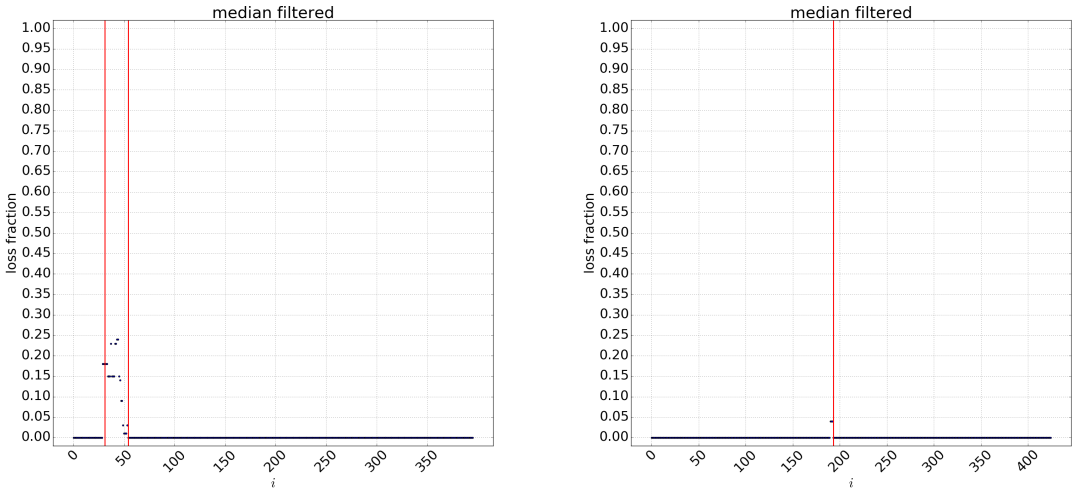
$$\min_{k,\boldsymbol{\tau}_{1:k}} \sum_{i=1}^{k+1} C(\mathbf{y}_{\tau_{i-1}+1:\tau_i}) + g(k) \tag{3.2}$$

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 [15–17], 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 outliers. 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

The idea that each segment is associated with a specific latent configuration has a direct interpretation to a Hidden Markov Model (HMM) model [18–20]. 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 occurs.

There are several approaches in the detection and training phases. For example, given a trained HMM, the most likely 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, such as those 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 train for each data stream 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 [20] 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 may not reflect real data. To overcome this problem, [20] 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 hyperparameters for this strategy. Instead, to surpass this issue, when using the best hidden state path, this dissertation used a HMM 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, which was used to train the HMM. The middle plot is the best hidden state path of the previous time series, in which the vertical axis indicates the distribution of each hidden state. The bottom plot is the discrete Hellinger distance of consecutive segments resulted from applying the sliding windows method to the best hidden state path.

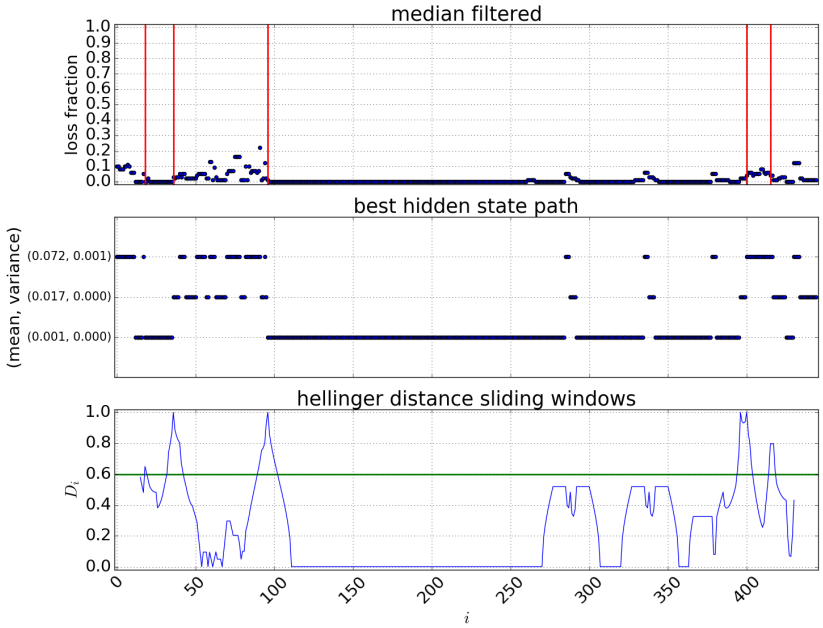


Figure 3.6: HMM filter.

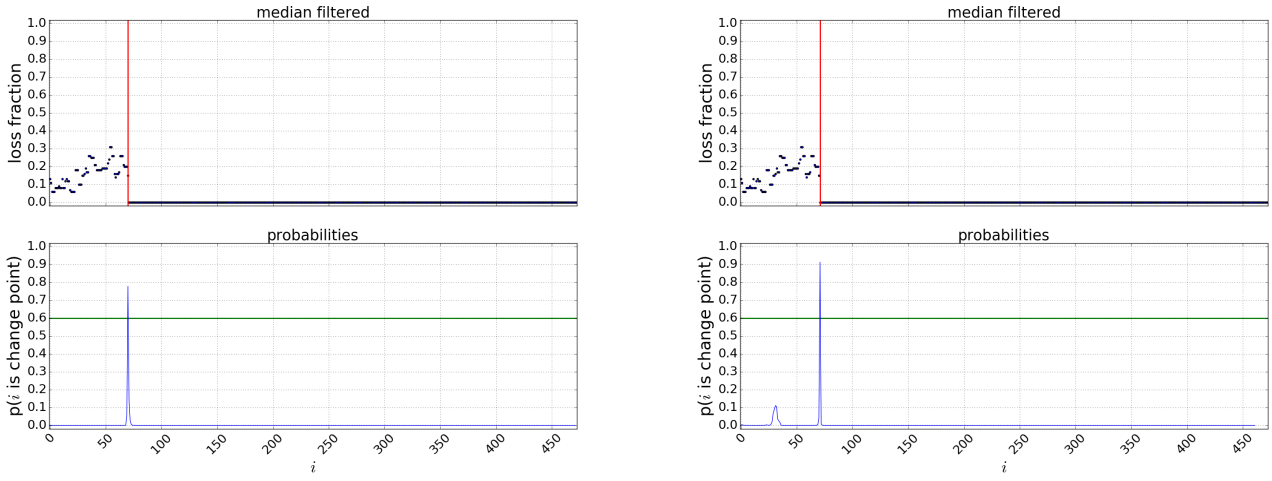
3.7 Bayesian Inference

There are several Bayesian methods which aim to assess the probability that a point is a change point. Following an offline fashion, the work of [21] 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. The overall complexity of this method is quadratic in n.

In [22] 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. In addition, the absence of a change points dataset ground truth, turn the selection of the most appropriate set of algorithms and hyperparameters a manual procedure.

In this context, the HMM and bayesian inference methods face a major disadvantage, since it was empirically verified that it is a challenging task to proper

choose their hyperparameters. The optimization model was the easier to manually fine tune, in order to establish a single set of hyperparameters that could generalize to different time series characteristics.

In Chapter 4 is discussed the issue relative to how to construct an events dataset.

Chapter 4

Methodology

This chapter describes the proposed data analytics workflow. The QoS data gathering procedures are briefly presented, and the proposed methodology is compared against those used in previous projects.

4.1 Measurement Environment

The time series used in this work result from end-to-end measures of a cable-television infrastructure, which runs DOCSIS with asymmetric download and upload bandwidths. A home gateway, connected to a cable modem, communicates with a server strategically located by the ISP. Each server is responsible for measurements of several end-users. Measurements are triggered from each home router every half hour and, by the end of every day, the results are transferred to a database. The measurement software was developed by the startup, and is spread over approximately two thousand customers of a major Brazilian tier-3 ISP.

To measure the round trip packet loss fraction, and the RTT, between the end-user and the associated server, the home router sends a train of short UDP packets, and the server bounces them back. The final RTT is the average of the individual packets' RTTs. The data here presented considers a train of 100 UDP packets of 32 bytes, separated by 1 millisecond. Maximum achievable upstream throughput is measured by overflowing the upload links with parallel Transmission Control Protocol (TCP) data transfers from the home router to the server. Traceroutes from the end-users to the respective servers are collected at 30 minutes intervals. The ping packets used in this process is also used to infer RTT per hop. Traceroutes from servers to end-users are not gathered. In [7] is presented a preliminary descriptive investigation of these metrics.

The resulting time series are unevenly spaced due to several reasons. First, measurements are initiated only if the residential link is not under use by the customer. Also, the client may have no Internet connection to start a measurement, or the

home gateway may be turned off.

Other details about the measurement software are presented during the text, as needed.

4.2 Proposed Workflow

This section describes, in a high level abstraction, the proposed workflow used to detect network events and localize their cause. When necessary, some steps are better explored in later sections.

The mechanism was motivated by an empirical visual investigation of the QoS time series. Besides the statistical changes that persist for long time periods, it was identified that clients with the same change point pattern, including the occurrence time, usually share physical network equipments.

The analysis seeks for events during a specific time interval, in a single measurement server, and considers a particular metric (round trip loss fraction, RTT, or maximum achievable upstream throughput), which are specified as parameters. A complete analysis can be accomplished through different executions with distinct parameters. Figure 4.1 illustrates the process.

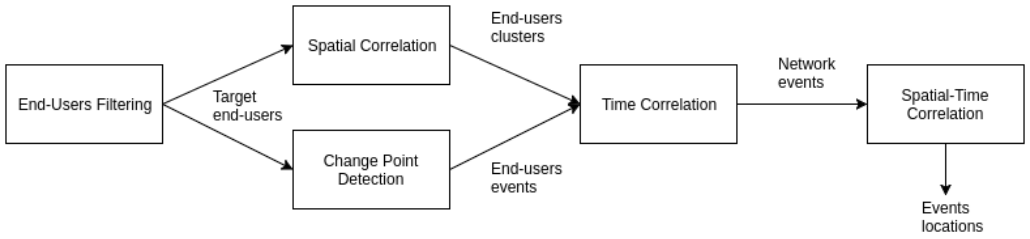


Figure 4.1: Pipeline.

The workflow starts with the End-Users Filtering step, which aims at removing clients that can negatively affect the analysis. As an example, this work eliminates clients with a small number of measurement samples.

The Change Point Detection phase preprocess the end-users' time series and identify change points for each time series. Also, the change points are classified in three types of events: failure, improvement, or inconclusive. For the RTT and round trip loss fraction, a change point defines a failure event if the average of the window after this point is greater than the average of the preceding window. The improvement event is analogous, however it is characterized by a mean decrease. The inconclusive event means that the segments averages are within a given tolerance. The same reasoning is applied to maximum achievable upstream throughput, however, a decrease means a failure, and a increase is related to an improvement.

The Spatial Correlation procedure clusterizes the end-users in user-groups according with their position in the network. This clustering can then be used in further steps to isolate possible events locations. This stage must produce a specific grouping structure, which is detailed in Section 4.3.

The Time Correlation step aims at combining similar events of different end-users. For example, if three clients detect a failure at approximately the same time, then this information is identified at this step. The grouped end-users events of an user-group are named as network events. The details of this method are enlightened in Section 4.4.

Finally, to localize events' causes, the Spatial-Time Correlation method matches network events with the user-groups structure. This is better explained in Section 4.5.

4.3 Spatial Correlation

This method provides the necessary information to determine which network equipments are shared by end-users that perceive the same network event. The developed strategies were guided by the specific ISP's topology.

The server position relative to the client can be distinguished in two scenarios. In the OFF-NET case, the traffic between them goes through a tier-2 ISP. In the ON-NET case, the traffic doesn't leave the tier-3 ISP's infrastructure. The latter situation represents a simpler Internet Protocol (IP) topology, since the tier-3 ISP uses static routing and doesn't applies load balancing nor Multi Protocol Label Switching (MPLS), techniques that are usual in the tier-2 ISP's network.

Then, considering measurements against a single server and the ON-NET case, paths from end-users to the server form a hierarchically structured topology, which is a required design for the Spatial-Time Correlation procedure 4.5. This work doesn't have access to the ISP's routing tables, therefore, this structure is reconstructed through traceroutes. In this context, each equipment that responds to traceroute pings is used as an user-group. An user-group is formed by all end-users in which traceroutes contain the associated network equipment.

The first hop of the traceroutes, which corresponds to the home gateway, is removed from the analysis. Additionally, Cable Modem Termination Systems (CMTSs) configured to answer traceroute pings are reported with private IP addresses, since they are located in the same subnetwork of the home router. Besides, due to multiple network interfaces, these equipments can be related to different IP addresses. Therefore, considering the lack of a private IP address to name mapping, the CMTSs were also discarded.

In the OFF-NET case, due to the load balancing applied by the tier-2 ISP,

the traceroute reconstruction doesn't lead in a hierarchical topology. Routers can implement three types of load balancing policies [23]. The per destination load balancing can be disregarded in the current analysis, since the target is always the same. The per flow load balancing attempts to maintain packets from the same flow in the same path. A flow is identified by header's fields of IP packets. For instance, a router can employ that packets with equal source/destination addresses, protocol, and source/destination ports, belong to the same flow. The per packet load balancing ensures that the load is equally distributed over the feasible output links, deploying, for example, a round-robin strategy. However, the latter method doesn't make attempts to keep packets from the same flow in the same path.

For an end-user and server pair, the measurement software uses the same ports over time to execute a specific measurement type, e.g., round trip loss fraction. Hence, if routers only apply per flow load balancing, and considering a time period without routing tables updates, and depending of which IP header's fields are used to define a flow, it is possible that all packets generated by a specific measurement type traverse the same path. However, these are strong and unlikely assumptions. Additionally, since the implemented traceroute also uses the same ports over time, and considering that different measurements types use distinct ports, the path followed by traceroute packets can be different from the path followed by the QoS measurements packets. Then, in this scenario, even with static paths in the tier-2 ISP's network, the correlation between traceroutes results with end-to-end metrics can lead to wrong conclusions.

Therefore, this work assumes that the tier-2 ISP only applies per packet load balancing, which is modeled here as a random strategy. With this hypothesis, if two end-users are geographically near, then it is likely that their packets traverse the same set of routers in the tier-2 network. This implies that if an event occurs in a tier-2 router, it is not possible that only one of these clients perceives the event. This is an important consequence explored in Section 4.5.

To handle the OFF-NET case, sequential hops related to the tier-2 ISP are compressed to a single hop. As an example, the path (tier-3 Router 1, tier-2 Router 1, tier-2 Router 2, tier-3 Router 2, Server) is transformed to (tier-3 Router 1, tier-2, tier-3 Router 2, Server). In general, this process conducts to a hierarchical topology. The exception occurs when the tier-2 ISP has output connections to different routers of the tier-3 ISP in the same network region. However, this is an unusual situation in the current dataset, and therefore, was removed from the analysis. The distinction between tier-3 and tier-2 hops is made through the investigation of hop names.

Figure 4.2 presents an example of the reconstructed topology. The equipments were anonymised, and the edges were oriented from the end-users to the server.

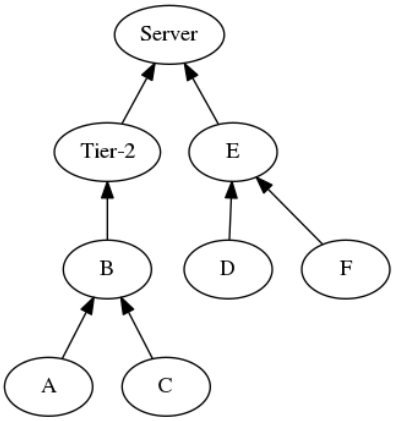


Figure 4.2: User-groups structure.

In this oriented tree, the zero indegree vertices indicate equipments that only appear as the first hop, however, an internal vertex can be a first hop in a traceroute. Considering an arc (A, B), all users who belong to A also belong to B, but the inverse is not necessarily true.

During the End-Users Filtering step, some clients are removed due to traceroute inconsistencies [23]. As an example, there are cases in which the same equipment appears in different hops of the same end-user traceroute. Also, the measurement software limits the maximum number of hops, then, scenarios in which the traceroute doesn't reach the server are discarded. Since routing tables can be updated, after the tier-2 vertices compression, it is only considered clients with static traceroutes during the specified time period. Situations characterized by tier-2 routers that not appear as consecutive elements in the traceroute are also deleted.

4.4 Time Correlation

The motivation of this mechanism is to infer network events that can explain end-users events of an user-group.

There are two main reasons to erroneous detect the same network event, such as an equipment failure, at distinct times in different clients. The first is related to the fact that the time series are not regularly sampled. The second is due to the change point detection algorithm behavior. Therefore, a procedure to group end-users events based on their type and time, should be flexible enough to take into consideration time delay effects. Also, it must be robust to deal with multiple change points per time series.

In order to relax a change location, a detected change point is transformed into an interval according to a time tolerance. Hence, given a parameter δ, a change point identified at time t implies that there is a change point in the interval $[t - \delta, t + \delta]$. To be consistent, change point algorithms must report points in time separated by

more than 2δ time units. In general, the algorithms presented in Chapter 3 can be adapted to respect this restriction. However, this can also be achieved by the following post processing step: considering that the points are sorted by time, and sweeping points from left to right, if two points are at most 2δ time units apart, then the right one is removed.

Consequently, the problem can be defined as selecting a set of network events, that explain all end-users events. It is important to note that events are defined only by their time and type. A client event is explained by a network event if they have the same type, and the end-user event time is inside the network event time interval. This problem has several possible solutions, and the comparison between them is subjective.

To this goal, it was developed a heuristic inspired in the Inexact Voting in Totally Ordered Space problem [24]. In this problem, people vote in a single point in the real line. Also considering a δ parameter, the objective is to select the interval with the greatest number of voters.

The created greedy procedure, called here as Multiple Inexact Voting in Totally Ordered Space, is specified in Algorithm 2

Algorithm 2 Multiple Inexact Voting in Totally Ordered Space

1: Let l be the end-users events of a specific type.
2: **while** l is not empty **do**
3: Select the interval d with the greatest number of end-users events that belong to l, and that all these events are at most 2δ apart. In case of ties, choose the left most one.
4: Report the average time of the d extremes as the network event time.
5: Remove all end-users events from l that are in d.
6: **end while**

It is possible to note that, in each iteration the procedure solves an instance of the Inexact Voting in Totally Ordered Space problem. The proposed algorithm is executed three times in every user-group, one for each event type.

A more straightforward solution, that was not used in this work, is to create regular time-bins, and interpret all end-users events that occur in the same bin as a common network event. However, this strategy can introduce discontinuities. If a network event occurs in the end/beginning of a time-bin, then it is likely that different clients events, associated with this network event, are going to be located in different bins.

If co-occurrent events of the same type affect the same end-users set, it is not possible to distinguish them. Nonetheless, the chances of detecting different events at the same time can be reduced by increasing the time series sampling frequency.

4.5 Spatial-Time Correlation

This section describes the steps that leads to matching network events from different user-groups in order to define a set of feasible events locations.

It is not possible to determine if changes in round trip metrics, such as RTT, are caused in the path from the end-user to the server or in the reverse path. Therefore, correlate these metrics with the available traceroutes can lead to erroneous or inconclusive conclusions. Also, the maximum achievable upstream throughput can be affected by a service degradation on the path from the server to the end-user, since losses of TCP ACKs can influence the measurement performance. Hence, this work restricts to report possible events locations considering that they are caused by equipments in the path from the end-user to the server.

With exception of the tier-2 user-group, the proposed mechanism assumes that if an event occurs in an user-group, all traffic that goes through the corresponding cluster is affected by the event. Since there is a direct connection between an user-group and an unique network equipment, this is a reasonable assumption. It is also assumed a hierarchical user-groups structure, and that there is at most one link between two network equipments.

In the first reasoning part, cases with no tier-2 user-groups will be processed. Also, traceroutes paths that do not start in a zero indegree vertex will be removed. Additionally, co-occurrent events with the same type will be disregarded. Moreover, it is assumed that the change point detection algorithm setup is able to detect all end-users affected by a network event, or no end-user is identified. Later, the mechanism will be expanded, and these restrictions will be removed. Unless stated otherwise, in the rest of this section an unique network event during the period of study is considered.

In order to better explain the proposed procedure, Figure 4.3 presents several topologies and different network events. The gray vertices indicate a network event that occurred in this location.

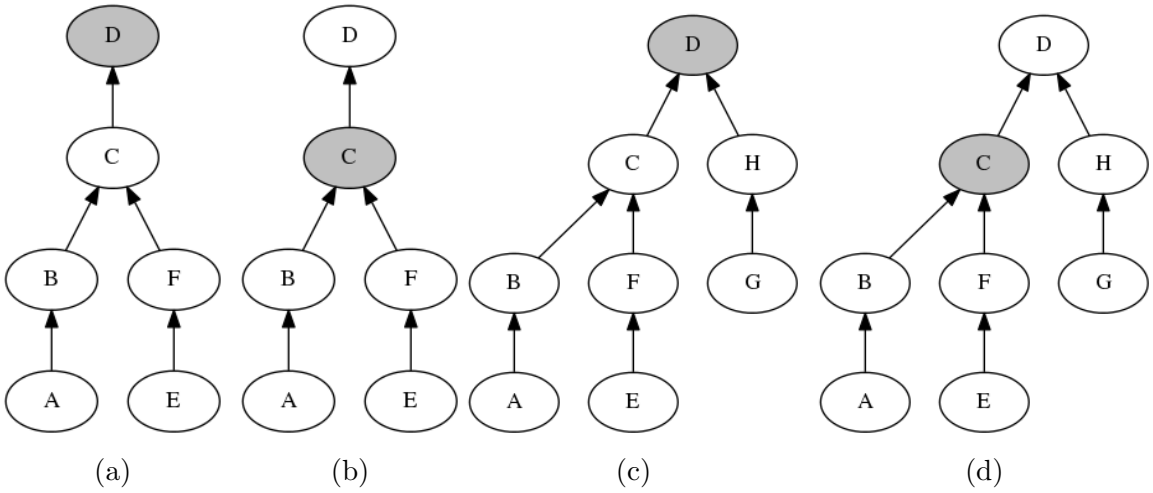


Figure 4.3: Network event location examples.

The procedure starts analyzing zero indegree vertices. Suppose that only a proper subset of the clients that belong to a zero indegree node perceive the event. In this situation, it is possible to affirm that this vertex is not the cause of the event, since not all clients detected it. For the same reason, the cause is not located in posterior nodes in the path to server. Then, if this set's size is larger than one, these end-users must share a network equipment, which is the cause, that is located before the first hop. As an example, this equipment can be a fiber node.

However, if all clients of the zero indegree vertex detect the event, then there are three options: this vertex can be the cause, or these end-users share an equipment before the first hop that explains the event, or the cause is located after the first hop. Due to the lack of information about the topology below the IP layer, the first two cases can't be distinguished. Nonetheless, to check the latter scenario, the same type of analysis is applied to the next vertex in the path to the server. If not all clients of the second hop detected this event, then it is surely located in the first hop or before. Otherwise, the procedure continues to the next hop.

When applied to different zero indegree vertices, this method results in possible events locations that are prefixes of the paths to the server. Also, these prefixes can be correlated. If different paths detect the same network event, then they must share at least one equipment. The ones that are not common can be discarded from the possible cause list, since there are end-users that don't belong to them and still perceive the event.

Additionally, it is possible to note that a failure in the directed link from a router A to a router B is identified as a fault in A.

In Figure 4.3a, the analysis starting at vertex A results in the following feasible event locations, {A, B, C, D}. The same reasoning using node E leads to {E, F, C, D}. Correlating both results, the possible locations are {C, D}. Figure 4.3b

presents a scenario with the same conclusion.

In Figure 4.3c, the analysis made using the zero indegree vertices results in the paths from them to the server. Matching the outcomes, the D vertex is the only possible event location.

In Figure 4.3d, the reasoning through node G doesn't detect the event. The analysis in A finds {A, B, C}. Through node E, {E, F, C} is the result. Correlating the outcomes, C is the only feasible event location.

The previous analysis can't be fully applied in situations with the tier-2 user-group. As an example, suppose two geographically distant end-users in an OFF-NET case. The packets from these two users can go through completely different routers in the tier-2 ISP. Then, it is possible that an event in the tier-2 ISP only affects one of the users, which contradicts one of the assumptions. To overcome this issue, when analyzing a path, if an event can be located in the previous hop of the tier-2 user-group, then the event can also be located at the tier-2 vertex, even considering that not all end-users of this user-group perceive the event.

Now, consider paths that does not start in zero indegree vertices. After the previous analysis, if all clients of the first hop user-group detect the event, then this path was already treated. Otherwise, the end-users that detected the event share an equipment before the first hop that is the cause.

The hypothesis that a change point detection algorithm setup is able to identify all, or none, end-users affected by an event, is too strict. As an example, consider two clients, one closer to the server than the other. If a failure in a router impacts the RTT of both users, probably, the RTT of the client that is closest to this router will suffer a larger relative variation than the other's RTT. Therefore, instead of checking if all users perceive the event, it is defined a threshold, and is verified if the fraction of clients that detect the event is greater than this value.

If there are co-occurrent events with the same type, the complete procedure can result in a wrong conclusion. Figure 4.4 illustrates this case.

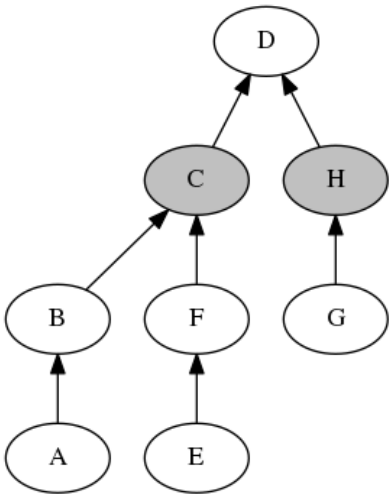


Figure 4.4: Co-occurrent events.

Through node A, the feasible locations are {A, B, C, D}. Checking E, the result is {E, F, C, D}. In vertex G, the outcome is {G, H, D}. The paths results correlation leads to the location D, which is a mistake. Therefore, when considering co-occurrent events with the same type, the step of matching the conclusions of different paths must not be applied, and the union of these results should be the final answer.

4.6 Change Point Detection Issues

As stated in Chapter 3, one of the main issues to be dealt with in this work is to select the proper algorithms and parameters. In general, this process requires a dataset to allow evaluation of the algorithm chosen and its related parameters.

There are several approaches to construct a change points dataset in the literature. Some works create simulated time series, in which distinct segments are sampled by the same generative model with different parameters [25]. 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 with different characteristics [20]. However, this can introduce unrealistic change points scenarios. Since one of the goals of this work is to deal directly with real data, this approach was discarded.

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, without even analyzing the time series. However, in the application domain of the present work, this approach is impractical. First, it would be necessary to know the network topology, routers congestion, physical equipment problems, among other features, and how they 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.

Another way is to use visual annotations, as it was done in [26]. Manual labeling has been used for anomaly identification in Internet traffic traces [27]. In this strategy, an application domain expert is exposed to a time series, and visually indicates his opinion about the change points locations.

It is known that visual inspection methods can bring erroneous conclusions [28], and also amplify subjectivity. However, to better understand the problem, this approach was experimented in this work.

Through a web system a 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. The user had visual access to the loss fraction time series with 10 days of data. Also, the selected series had at least 85% of the maximum possible number of points during the specified period, considering that data is sampled at most twice per hour. Change points can be interpreted as rare events in this dataset, and several data streams have almost all measures with zero losses. Therefore, to increase the entropy, it was selected time series that had at least one window of length 48 with more than 5 measures with loss fraction larger than 0.01.

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.

Figure 4.5 presents a system snapshot. The vertical red line means that the user marked a change point in that position.

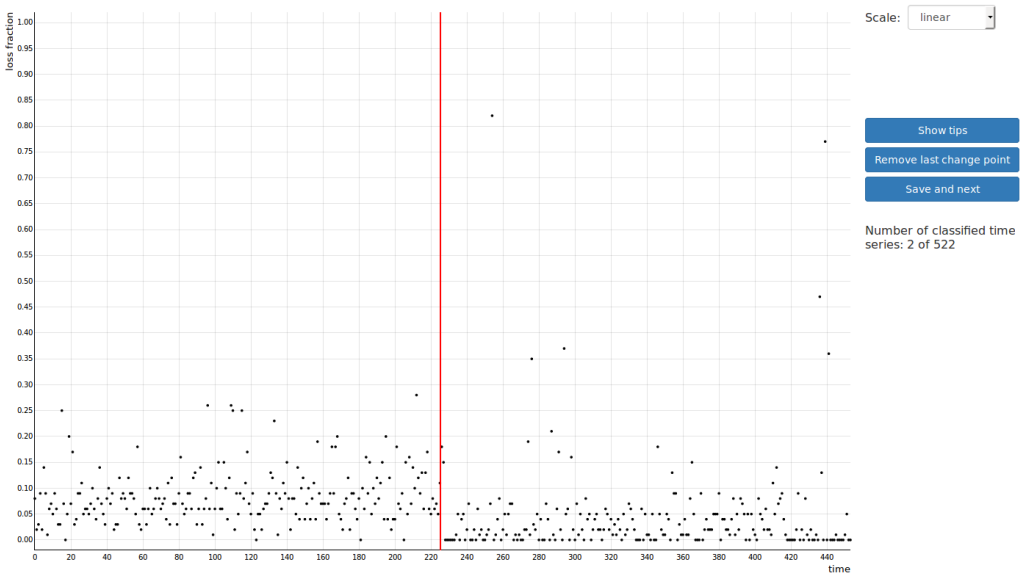


Figure 4.5: Survey system snapshot.

Six specialists with experience in network measurements and statistical modeling, but without background in change point detection, classified 71 time series. To analyze the agreement between different users classifications, for each time series, was applied the Time Correlation procedure 4.4. Therefore, it was considered that each specialist voted to a set of change points positions. The δ parameter described in Section 4.4 was set to 7.

Then, for each change point voted at least once, it was counted the number of specialists that voted in that location. Figure 4.6 shows the histogram of this counting.

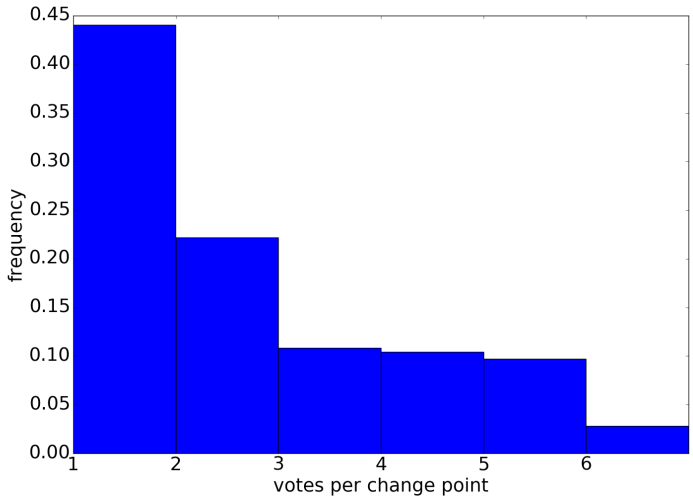
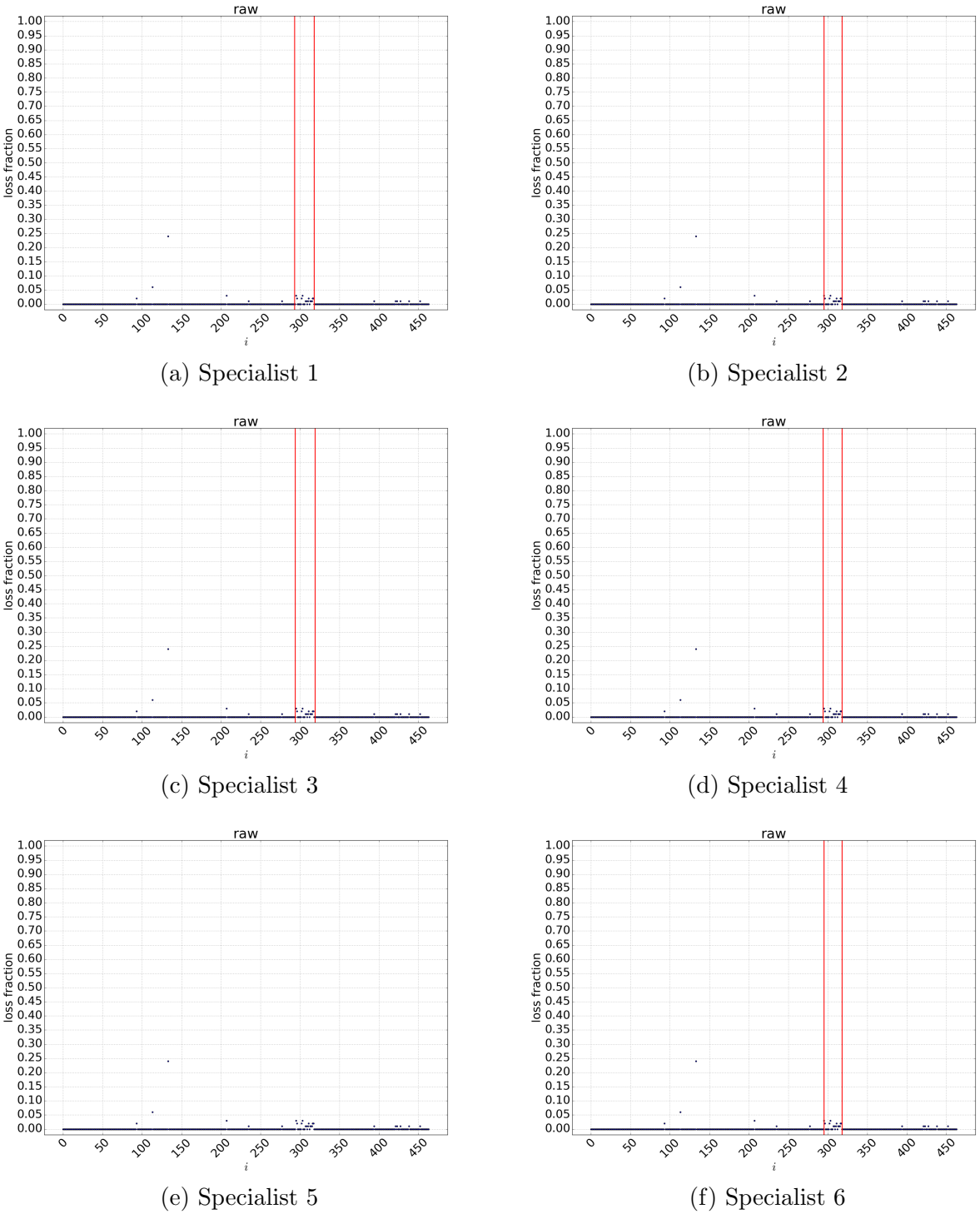


Figure 4.6: Number of votes per change point histogram.

It is possible to note that, 44% of the change points were only voted by a single

user, and only 23% were voted by the majority (more than 3 votes). Therefore, in general, the consensus of change point locations is low.

Figure 4.7 shows the classifications obtained from this experiment in one of the time series that had a high level of agreement on the change points locations.



(a) Specialist 1

(b) Specialist 2

(c) Specialist 3

(d) Specialist 4

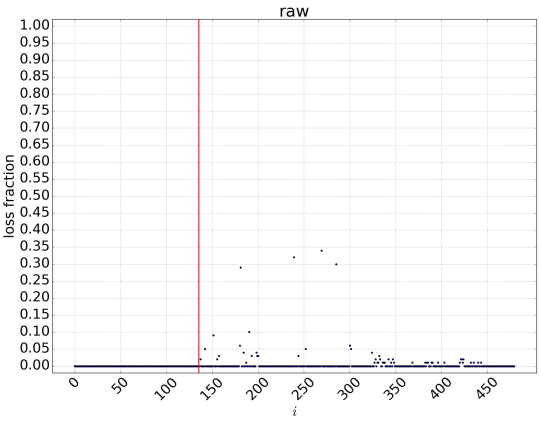
(e) Specialist 5

(f) Specialist 6

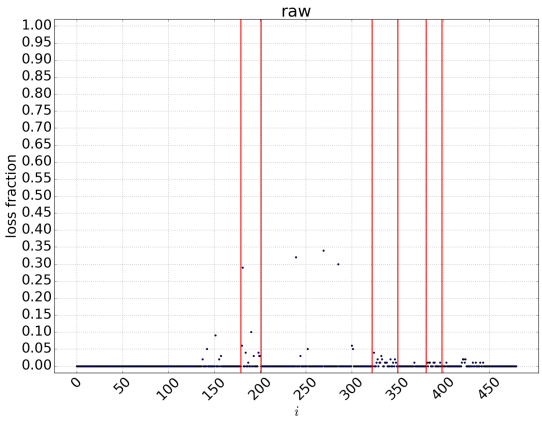
Figure 4.7: Classifications agreements.

Figure 4.8 shows a time series with several disagreements. It was verified that

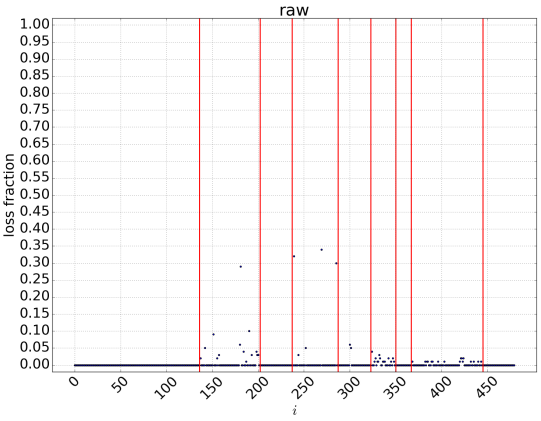
this case is the most representative in the constructed dataset, fact that corroborates with the problem subjectiveness.



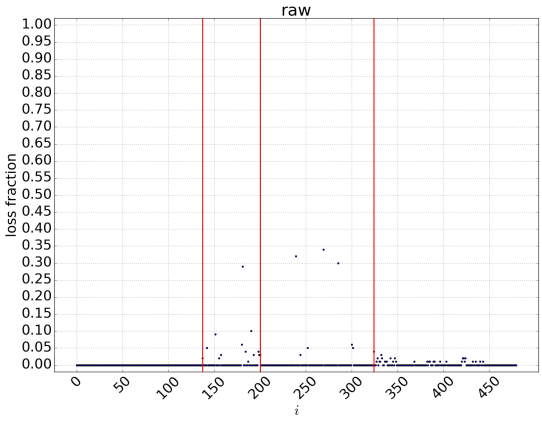
(a) Specialist 1



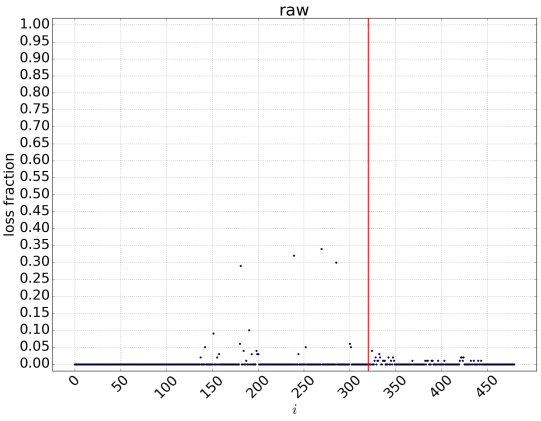
(b) Specialist 2



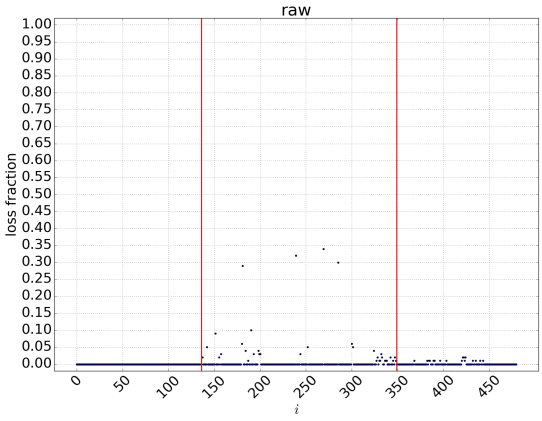
(c) Specialist 3



(d) Specialist 4



(e) Specialist 5

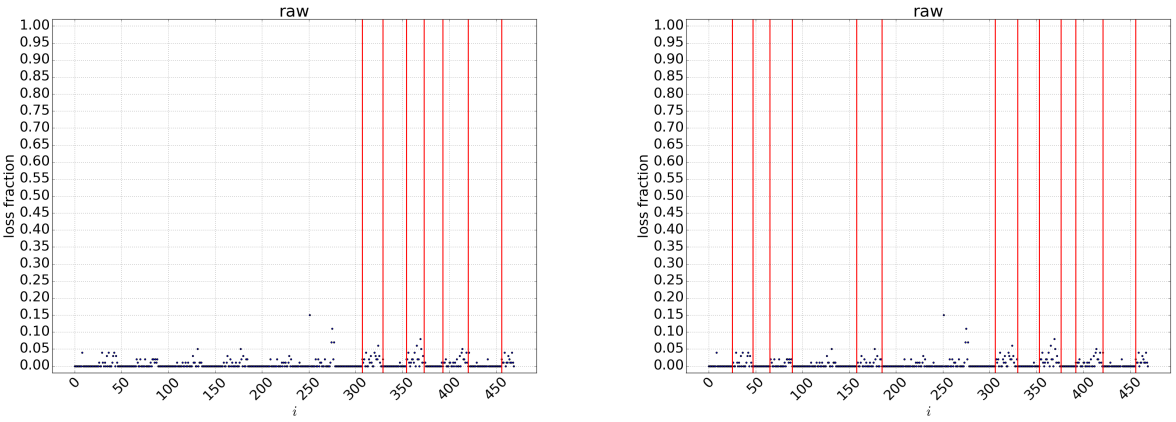


(f) Specialist 6

Figure 4.8: Classifications disagreements.

Also, it is possible to note that some users apparently changed their classification pattern in the same time series. As an example, Figure 4.9 presents two specialists that fits this description. Also, in general, users changed their classification pattern

in different time series.



(a) Specialist 1 (b) Specialist 6

Figure 4.9: Different classification pattern in the same time series.

Since this initial experiment resulted in a noisy dataset, in which change points probably don't reflect real network events, this strategy was aborted. Also, it would be difficult to scale the study to include more specialists and time series.

It is important to note that the algorithms described in Chapter 3 are unsupervised methods. Once a change point dataset is constructed, it is possible to apply supervised learning procedures, which are not much explored in the change point detection literature.

4.7 Differences from Previous Work

The proposed data analytics architecture has several similarities with the projects described in Chapter 2.

As with Argus and NetNorad, this work clusters end-users in user-groups, however with finer topology granularity. Also, to increase the system's scalability, NetNorad and Argus use this grouping to reduce the number of tracked time series. This strategy was not applied in this work, since the finer granularity requires more time series to be spread in different network locations, and the number of tracked end-users tracked by the measurement software is low.

Additionally, beyond the network edge point of view, Argus and NetNorad uses some internal network information, which is absent in the present work. Besides, Argus uses anomaly detection, while this work deploys change point detection.

Chapter 5

Results

As stated in Chapter 4, due to the absence of an events dataset, it is not possible to extensively study the accuracy of the proposed procedure. However, this chapter presents illustrative examples when the proposed workflow was applied to real data.

It was considered 7 months of measurements, from May to November 2016. This data was then split in batches of 10 days, and for each batch, a complete offline analysis was executed. During this time period the measurement software was deployed in 35 servers, and the mean number of client-server pairs, such that at least one measurement between them occurred during a batch, was 2246. After the End-Users Filtering step 4.2, this average reduced to 741. In general, one client measured against a single server through the batch.

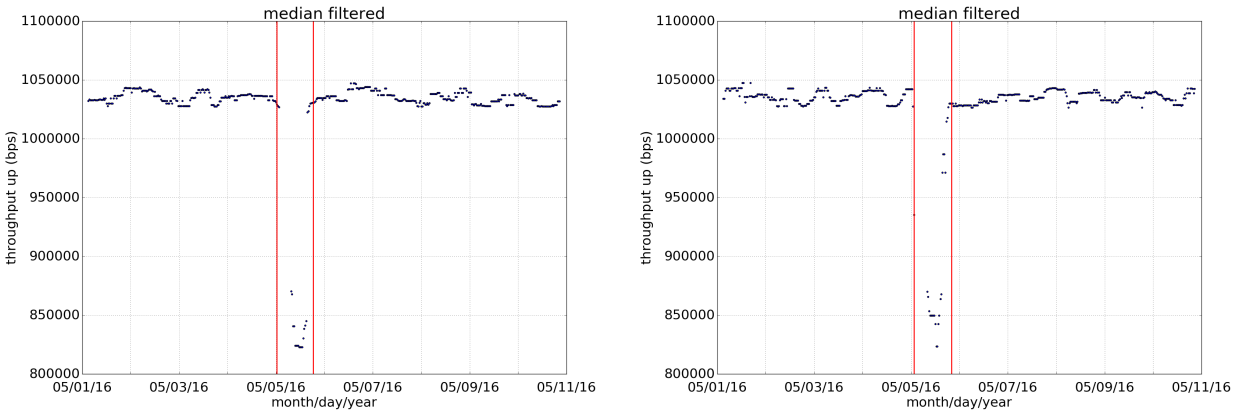
For all cases, the time series were preprocessed with a median filter. The change points were detected through the optimization model described in Chapter 3. For each QoS metric, the algorithms' hyperparameters were manually selected. It was opted to use conservative values, in order to avoid change points that, through a visual inspection, may be arguably not related to a network event. These algorithms choices were guided by two facts. First, through an empirical visual analysis, it was verified their reasonable performance with real data. Also, the impact of their hyperparameters can be easily interpreted, which is an important feature for manual tuning.

The fraction of clients threshold, used to check if an event can be located in a vertex, was set to 0.75. The δ parameter, used to verify if two change points are close, was set to 4 hours. It was disconsidered the presence of co-occurrent events of the same type in the same network region.

Section 5.1 presents examples with potential correct outcomes, and Section 5.2 exposes cases with possible wrong results. These conclusions were manually corroborated with a visual check.

5.1 Possible Correct Outcomes

Figure 5.1 shows a proper subset of clients that belong to a specific user-group, modeled by a zero indegree vertex.

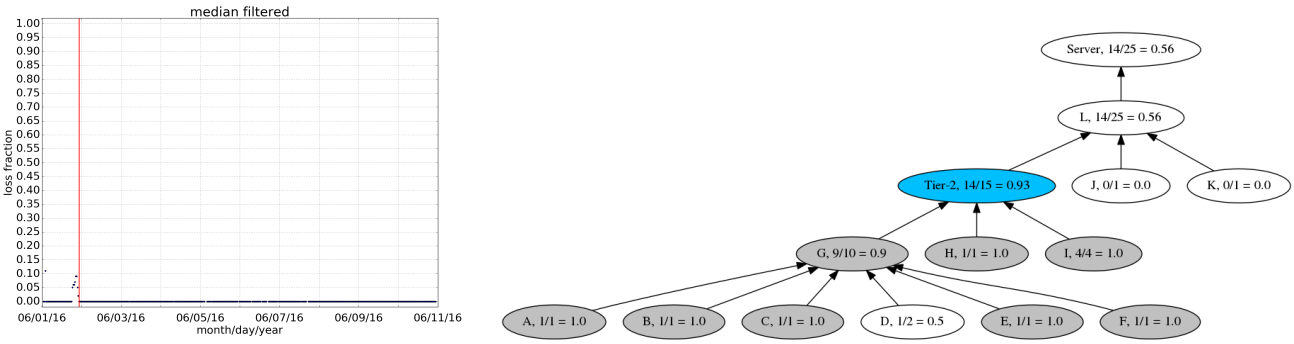


(a) Client 1. (b) Client 2.

Figure 5.1: Before first hop.

From the 4 clients that belong to this user-group, the system detected the illustrated events only in these 2 customers. Then, considering the established suppositions, these clients must share a physical equipment before the first hop that caused the events.

Figure 5.2a shows a client with a specific network event, and Figure 5.2b presents the user-group structure in which this customer belong. The vertices are defined by a tuple, in which the first entry is a label to the user-group, and the second specifies the fraction of clients that detected the considered event. The gray vertices represent possible locations resulted from the analysis starting in zero indegree user-groups. The blue vertex indicates the correlation between these results.

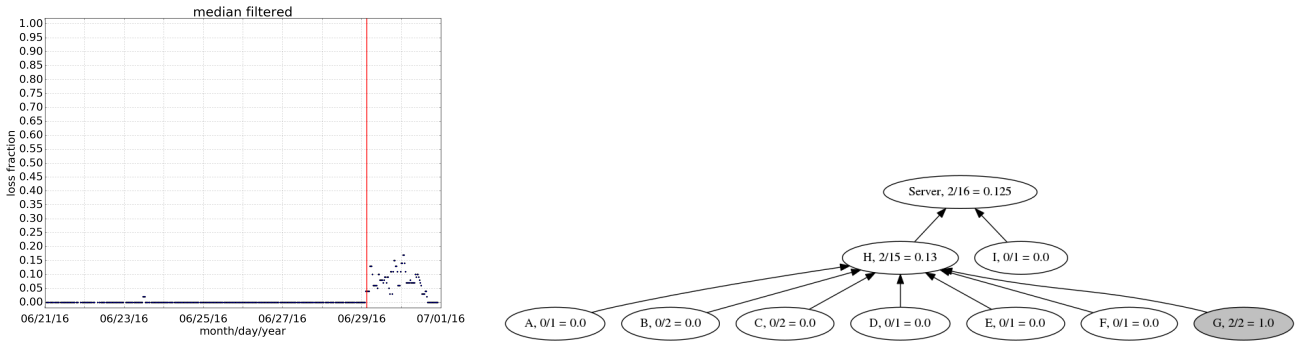


(a) Client 1. (b) User-groups structure.

Figure 5.2: Correlation of problem locations detected by analysis that started in zero indegree vertices.

It was visually verified that the same change point pattern of Figure 5.2a is present in all clients that belong to the subtree rooted in the tier-2 vertex. Therefore, the system incorrectly did not detect the event in one of the D user-group clients. However, since several other G children detected the event, the system's output was not compromised by this error. Through the correlation of the zero indegree analysis, the only match between the problem locations was the tier-2 group.

Figure 5.3 presents an example of a network event that was only identified in one zero indegree user-group.



(a) Client 1. (b) User-groups structure.

Figure 5.3: Network event in only one zero indegree user-group.

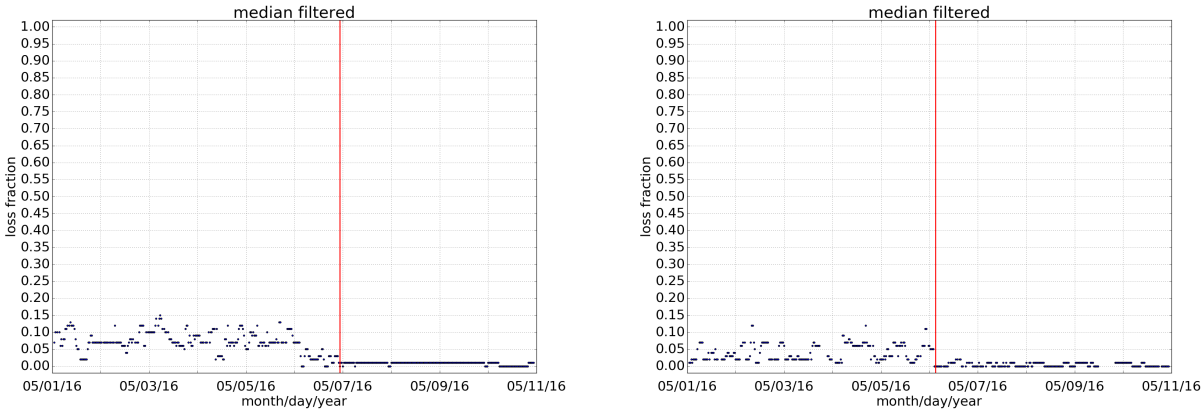
Table 5.1 summarizes the number of events, discriminating per metric and type ("Improvement" or "Failure"). The system didn't detect inconclusive events.

Metric	Events	
	Improvement	Failure
RTT	1520	1532
Round trip loss fraction	310	262
Maximum achievable upstream throughput	730	641

Table 5.1: Number of events.

5.2 Possible Incorrect Outcomes

Figure 5.4 shows two clients that belong to the same zero indegree user-group.

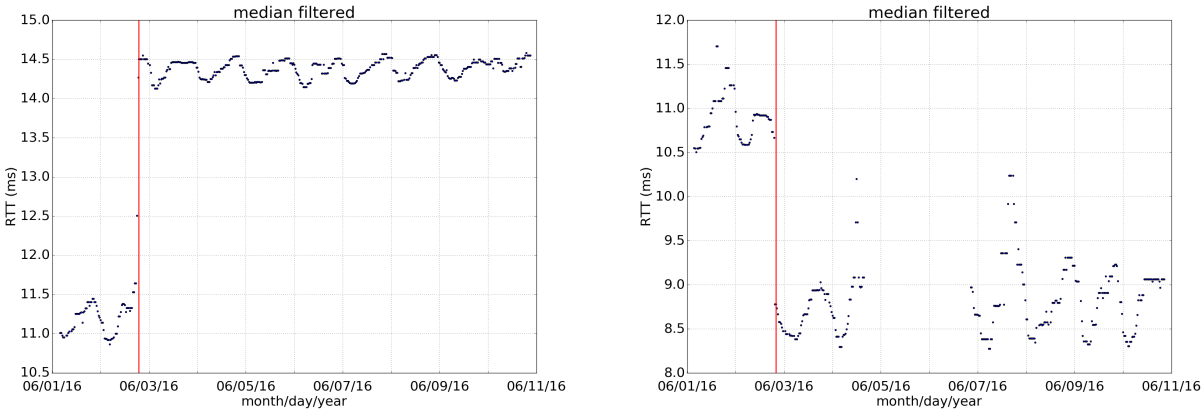


(a) Client 1. (b) Client 2.

Figure 5.4: Time correlation unmatch.

Visually, both time series have similar patterns. However, in the left client, the change point was detected near the end of 05/07/16 day, while in the right customer, the change was identified in the beginning of the same day. All 17 clients of this zero indegree vertex belong to one of these two cases. Considering the used δ value, the system recognized two different events, nonetheless, these changes are possibly related to a single event. This exemplifies the problem difficulty when there isn't a ground truth, and also the importance of the algorithms and their hyperparameters selection. Such kind of subjectivity is common in the current dataset.

Figure 5.5 shows two clients that belong to the same zero indegree vertex.



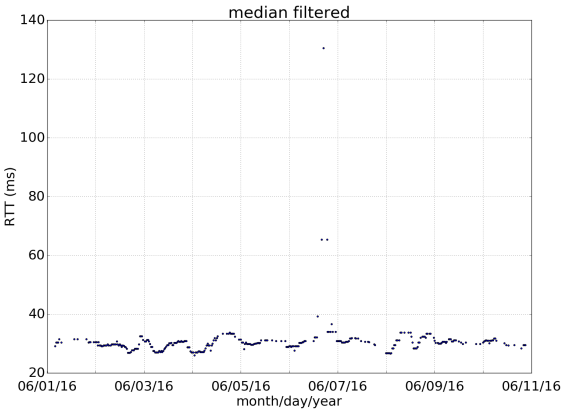
(a) Client 1. (b) Client 2.

Figure 5.5: Untraceable location. Plots with different scales.

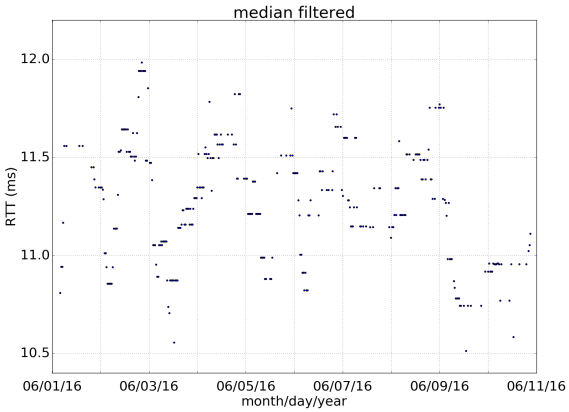
It is possible to note that both RTT time series changed their pattern nearly at the same time, however, one case is characterized by an improvement, while the other as a failure. The traffic between these customers and the server doesn't go to through the tier-2 ISP, therefore, according to the assumptions, these patterns

indicate simultaneous events that occurred at different equipments before the first hop. Nonetheless, the system detected the same RTT modifications, at the end of 06/02/16, in several other clients. Comparing these customers, it was verified that many of them don't share several attributes. For instance, they executed measurements against different servers, including both ON-NET and OFF-NET cases, their upstream traffic went to completely different IP layer equipments, and they were located at different Brazilian states. This can indicate a single global network event, that simultaneously affected different network regions. As an example, the event could be a centralized change of DOCSIS' parameters, that is concurrently propagated to all customers. As with local events, this work doesn't have access to this kind of information, however, it is intriguing the fact that some of the clients detected a failure while others perceived an improvement. It was manually identified 3 of such cases, and the RTT was the only impacted metric. These cases possibly explain why RTT events are considerably more frequent than events of other metrics (see Table 5.1), since several identified events would be related to a single real event.

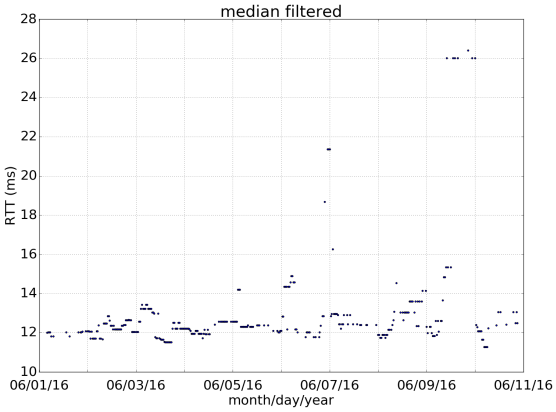
Motivated by the previous example, next is presented an analysis of the RTT per hop, which is obtained through the traceroute measurements. However, this investigation can be fairly different from the analysis of the end-to-end RTT measures. First, their methodologies are remarkably distinct. Second, the RTT associated with hop i can be constantly and considerably larger then the RTT related to a hop j, where $j > i$. This can be explained by the fact that some routers prioritize forwarding instead of answering ping packets. Therefore, the analysis of this data was not included in the framework. For instance, Figure 5.6 presents the RTT per hop, disregarding the server hop, of the client of Figure 5.5a.



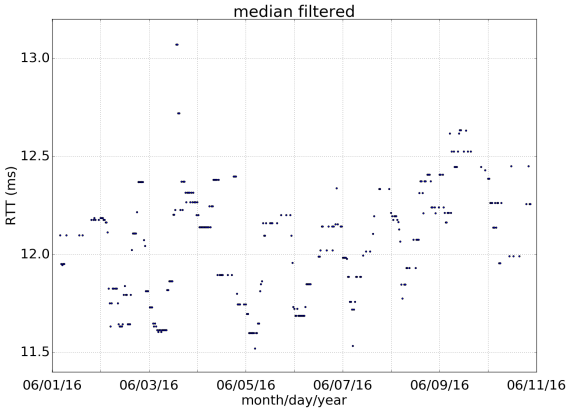
(a) Hop 1



(b) Hop 2



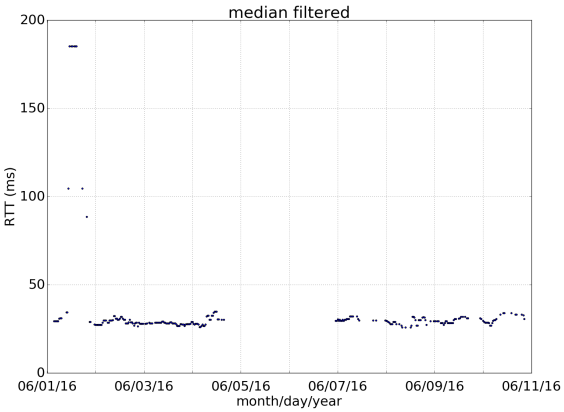
(c) Hop 3



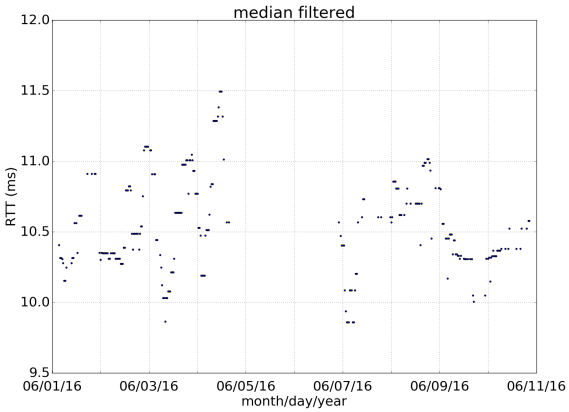
(d) Hop 4

Figure 5.6: RTT per hop of client of Figure 5.5a. Plots with different scales.

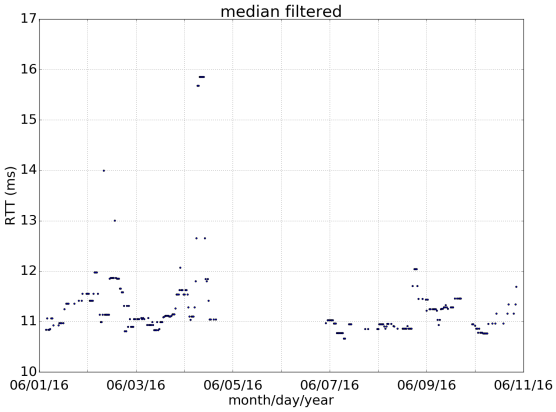
It is not possible to visually identify the same change pattern of Figure 5.5a in the hops analysis. Figure 5.7 is analogous, however, related to client of Figure 5.5b.



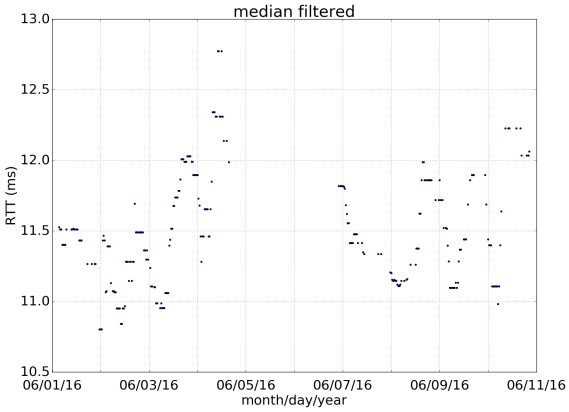
(a) Hop 1



(b) Hop 2



(c) Hop 3



(d) Hop 4

Figure 5.7: RTT per hop of client of Figure 5.5b. Plots with different scales.

In this case, it can be noticed that the RTT is larger in the beginning of the first hop time series. This same pattern was detected in the RTT between the client and the server.

5.3 Final Remarks

Various events related to the maximum achievable upstream throughput consist of a substantial mean increase perceived by a single client, as is exemplified in Figure 5.8. Possibly, this type of event indicates a change in the customer's contracted bandwidth.

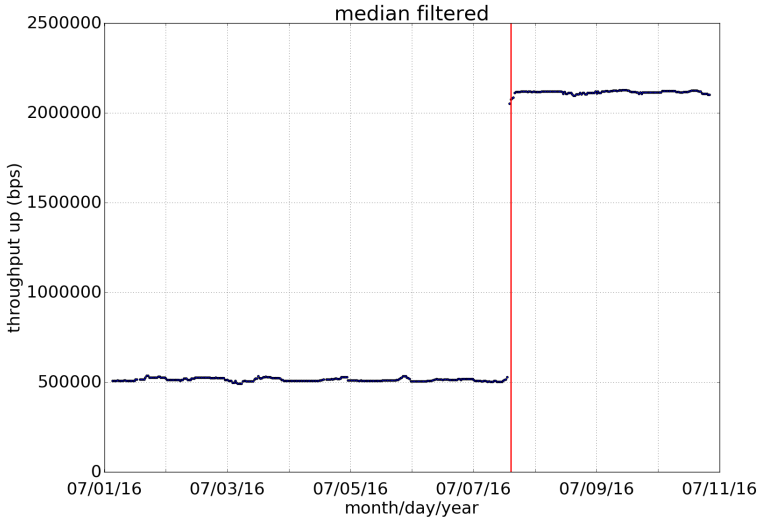
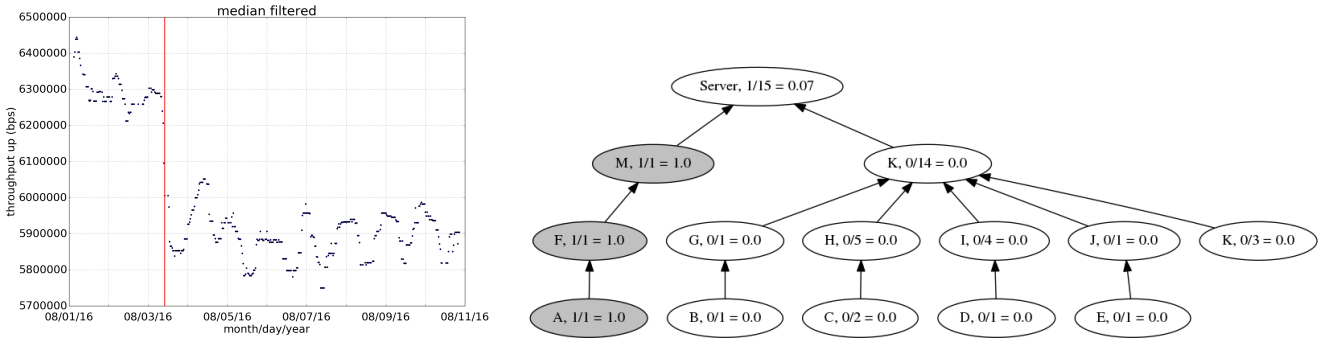


Figure 5.8: Possible contract bandwidth change.

Several problem locations are supported by a change point detected in a single client. Figure 5.9 exhibits an example.



(a) Client 1.

(b) User-groups structure.

Figure 5.9: Clients sparsity.

It is not possible to exactly locate the event, since the three gray vertices are composed by the same end-user. This can be avoided with an appropriate selection of the tracked customers.

Chapter 6

Conclusions

Considering the specific ISP's topology, and the current end-to-end measurement methodology, this dissertation proposes a data analytics framework to detect and localize network events. For such purpose, the mechanism tracks statistical changes in the end-to-end QoS time series from different clients, and correlates these patterns with traceroutes. Finally, several outcomes were presented when the procedure was applied to real data.

The results show that, considering the exposed restrictions, it is possible to use only end-to-end QoS metrics, and traceroutes, from different clients, to identify and localize network events. However, due to the lack of an events dataset ground truth, important questions persist unanswered. For instance, a quantitative accuracy study would allow to check which types of events can't be handled by the proposed mechanism. A dataset would also allow a precise analysis of the events patterns, how they impact the QoS metrics, and fine tune the system.

The use of end-to-end measurements has the advantage of dealing with metrics that are directly related with the service perceived by the customers. As an example, this feature can be used to rank simultaneous failure events according with their impact to the end-users. Nonetheless, would be interesting to study the impact of introducing internal network information to the framework.

Besides the data availability, the current measurement process imposed several restrictions to this work. In order to better explore the proposed solution, a further continuation of this project would require a stronger partnership with the ISP. In addition, in order to improve the proposed framework's performance, it would be desirable to adapt the measurement software. However, this dissertation can be used as a first guide to the ISP's engineers.

6.1 Contributions

Next is summarized this dissertation contributions.

- An automatic procedure, that only uses the available end-to-end QoS measurements, and traceroutes, to detect and localize network events in the specific tier-3 ISP's infrastructure.

- A list of possible improvements in the measurement methodology currently employed by the startup, in order to enhance the proposed system's performance. This list is presented in the next Section 6.2.

6.2 Future Work

Next is presented a list of future development directions of this work. Considering the detection and localization of events, several adaptations to the current measurement methodology are suggested.

- The used QoS metrics are affected by equipments in the path from the server to the end-user, as well as those in the reverse direction. However, only the path information from end-users to servers are available. Hence, the traceroutes from servers to end-users should also be collected. Besides, instead of only considering the round trip loss information, the one way loss fraction in both directions can be tracked. Further, the maximum achievable one way throughput measurement can be implemented using UDP instead of TCP, which eliminates the interference of performance degradation in the reverse path.

- The Hybrid Fibre Coaxial (HFC) plant between the home router and the first hop of the traceroute can be incorporated to the analysis. Also, in addition to only using end-to-end QoS metrics, the proposed mechanism can be extended to use internal network devices information, such as signal-to-noise ratio.

- A network failure events dataset can be built with ISP's data. As an example, customers' complaints gathered from call centers can be used to, during a specific time period, infer clients affected by a Quality of Experience (QoE) deterioration, which can then be translated to true network failures. Additionally, it is possible to use records from current failure detection methods deployed by the ISP, such as manual inspection, or through equipments that are able to report specific faults. However, through preliminary talks with ISP's engineers, both databases are noisy, and mining useful information from them can be a challenging task. For instance, there are cases in which devices flaws are manually detected and corrected, but those information is not stored. Also, during the dataset construction, the inferred events times can be considerably different from their true occurrence time. Besides, since the

tier-2 ISP is not a project partner, this complete data of the network infrastructure may not be available. As stated in Chapter 4, a dataset could open new supervised learning possibilities to the change point detection problem, such as hyperparameter optimization and model selection. In this context, Recurrent Neural Networks can be useful to handle non-stationary segments and unevenly spaced time series.

- Once a network events dataset is constructed, the correlation between change points of different QoS metrics can reveal useful information about the network behavior. This analysis was not done since the algorithms' hyperparameters couldn't be optimized, hence, these comparisons could reach wrong conclusions.

- A deeper knowledge of the tier-2 ISP's infrastructure can be used to model the tier-2 network with finer granularity in the Spatial Correlation procedure, which can improve the system's event localization precision.

- It is planned in the ISP's roadmap to increase the number of tracked customers. In this case, the system's computational performance can benefit from data aggregation techniques, as it was done in Argus. Besides, this increase will naturally improve the internal network equipments coverage by the end-to-end measurements, which, as the previous topic, can enhance the events localization precision.

- Once the system is deployed, the algorithms and parameters can be selected through a reinforcement learning approach. If network operators feedback the outcomes' correctness, the system can adaptively optimize the used strategies.

- Considering a real time processing environment, in order to decrease the event detection delay, the system can adaptively control the measurement frequency. Increasing the amount of data related to potentially problematic regions, can improve the system's output confidence in a short time period. Also, it is possible to reduce the measurement frequency in well behaved localities, which can lower the traffic overhead generated by measurements, and increase the data analytics computational performance.

- Instead of centrally process the time series, it is possible to instrument the home gateways to detect changes in an online fashion. Then, as with CEM, the home routers could push this information to a central database for further analysis.

- Extend the mechanism to deal with other types of network infrastructures.

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.

[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.

[5] CHATFIELD, C., YAR, M. "Holt-Winters forecasting: some practical issues", *The Statistician*, pp. 129–140, 1988.

[6] AMINIKHANGHAHI, S., COOK, D. J. "A survey of methods for time series change point detection", *Knowledge and Information Systems*, pp. 1–29, 2016.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

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

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] AUGUSTIN, B., CUVELLIER, X., ORGOGOZO, B., et al. "Avoiding traceroute anomalies with Paris traceroute". In: *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 153–158. ACM, 2006.

[24] PARHAMI, B. "Voting algorithms", *IEEE transactions on reliability*, v. 43, n. 4, pp. 617–629, 1994.

[25] 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.

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

[27] RINGBERG, H., SOULE, A., REXFORD, J. "Webclass: adding rigor to manual labeling of traffic anomalies", *ACM SIGCOMM Computer Communication Review*, v. 38, n. 1, pp. 35–38, 2008.

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

[29] 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.

[30] 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.