Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

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

Diego Ximenes Mendes

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

Orientador: Edmundo Albuquerque de Souza e Silva

Rio de Janeiro
Janeiro de 2017

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

Diego Ximenes Mendes

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

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2017

Ximenes Mendes, Diego

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

IX, 38 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. 35 – 38.

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

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

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

Diego Ximenes Mendes

Janeiro/2017

Orientador: Edmundo Albuquerque de Souza e Silva

Programa: Engenharia de Sistemas e Computação

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

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

Diego Ximenes Mendes

January/2017

Advisor: Edmundo Albuquerque de Souza e Silva

Department: Systems Engineering and Computer Science

Contents

List of Figures **viii**

List of Tables **ix**

1 Introduction **1**

1.1 Contributions 1

1.2 Dissertation Outline 1

2 Literature Review **2**

2.1 Argus 2

2.2 NetNorad 4

2.3 CEM 5

3 Change Point Detection **7**

3.1 Problem Definition 7

3.2 Notation 8

3.3 Preprocessing 9

3.4 Sliding Windows 10

3.5 Optimization Model 12

3.6 HMM (Hidden Markov Model) 13

3.7 Bayesian Inference 15

3.8 Final Remarks 16

4 Methodology **17**

4.1 Measurement Process 17

4.2 Pipeline 18

4.3 Spatial Correlation 19

4.4 Events Times Correlation 21

4.5 Spatial-Time Correlation 23

4.6 Change Point Detection Issues 25

4.7 Differences to Previous Systems 31

5 Results **33**

6 Conclusions **34**

Bibliography **35**

List of Figures

2.1 Argus pipeline. [1] 2
2.2 Argus spatial aggregation. [1] 3
2.3 Facebook's network architecture. [2] 4

3.1 Median filter RTT. $w = 6$. 9
3.2 Median filter loss fraction. $w = 2$. 10
3.3 Online Sliding Windows. 11
3.4 Offline Sliding Windows. 11
3.5 Optimization model. 13
3.6 HMM filter. 14
3.7 Bayesian Inference. 15

4.1 Pipeline. 18
4.2 User-groups structure. 21
4.3 Network event location examples. 24
4.4 Co-occurent events. 25
4.5 Survey system snapshot. 27
4.6 Number of votes per change point histogram. 28
4.7 Classifications agreements. 29
4.8 Classifications disagreements. 30
4.9 Different classification pattern in the same time series. 31

List of Tables

Chapter 1

Introduction

1.1 Contributions

1.2 Dissertation Outline

Chapter 2

Literature Review

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

2.1 Argus

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

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

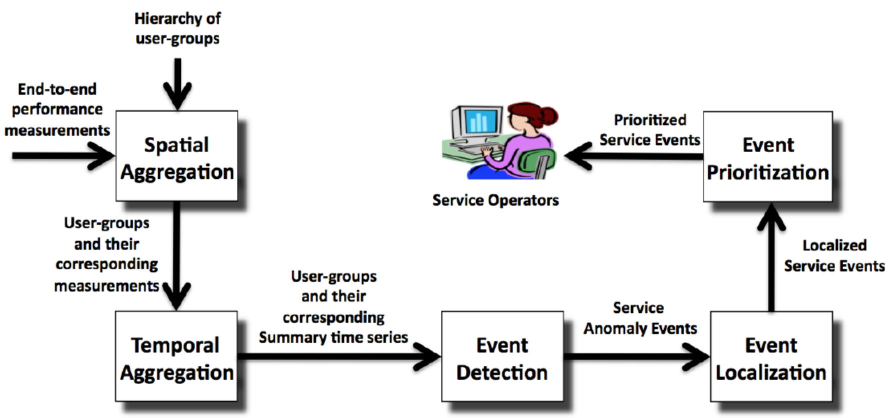


Figure 2.1: Argus pipeline. [1]

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

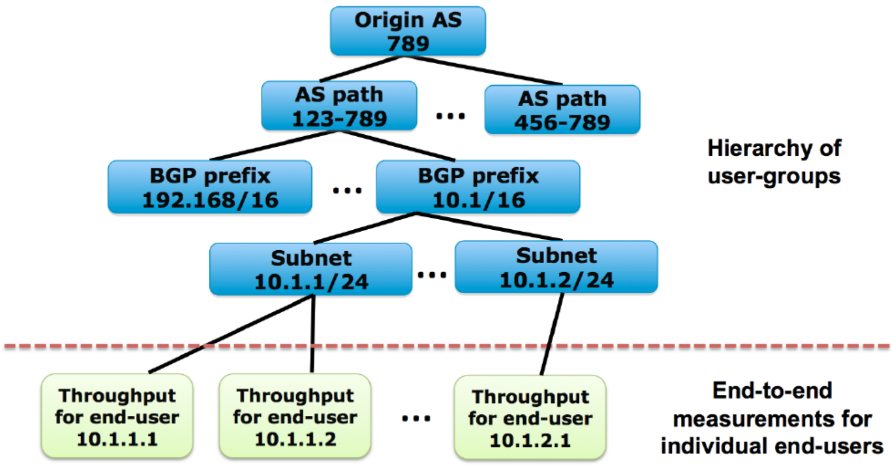


Figure 2.2: Argus spatial aggregation. [1]

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

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

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

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

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

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

2.2 NetNorad

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

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

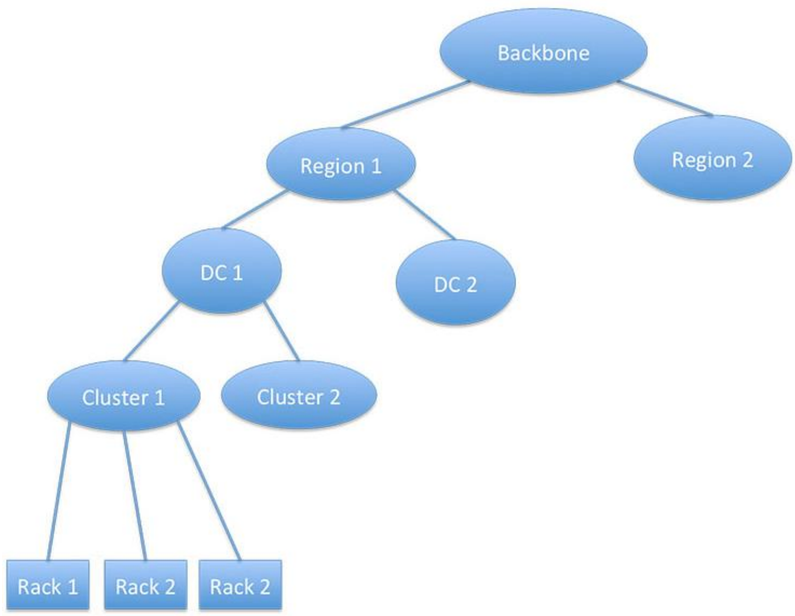


Figure 2.3: Facebook's network architecture. [2]

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

As with Argus, NetNorad applies spatial aggregation techniques. The pingers group the responses of machines that belong to the same cluster during a round, and tags them according with their relative location. Tags are defined by the following patterns: “DC” if the target cluster is in the pinger’s data center; “Region” if the target cluster is outside the pinger’s data center but within the same region; “Global” if the target cluster is outside the pinger’s region.

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

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

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

2.3 CEM

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

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

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

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

Chapter 3

Change Point Detection

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

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

3.1 Problem Definition

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

It is intuitive that the offline case is more robust, since there is more information to analyze. In practice, to increase the statistical confidence of a decision, the online definition is relaxed, and to decide if a point in time is a change point it is possible to use data up to a small window in the future, which in real time processing means that the application should wait until additional data is available. Hence, there is a trade-off between minimizing the time to detect a change and correctly classify a point. Therefore, in some cases, the online version can be transformed in offline by minor modifications.

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

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

3.2 Notation

An univariate time series composed of n points is defined by two vectors, $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$. 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 $i = 2, \dots, n$. Since unevenly spaced time series is considered, $x_i - x_{i-1}$ can be different for different i values. For $s \leq t$ the following notation is adopted: $\mathbf{y}_{s:t} = (y_s, \dots, y_t)$.

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

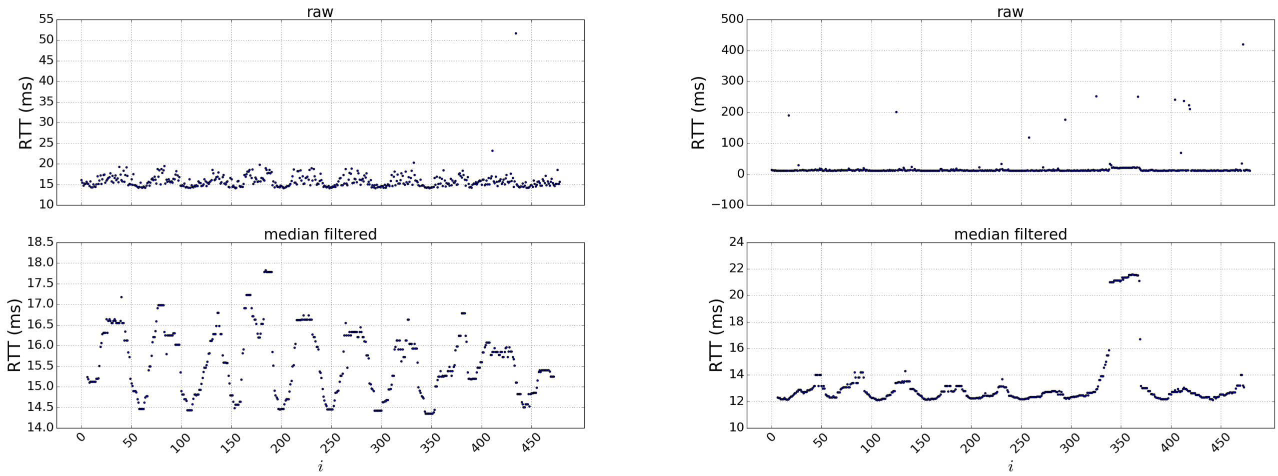
Through the previous definitions, change point detection algorithms mainly aim to find both k and $\boldsymbol{\tau}$.

3.3 Preprocessing

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

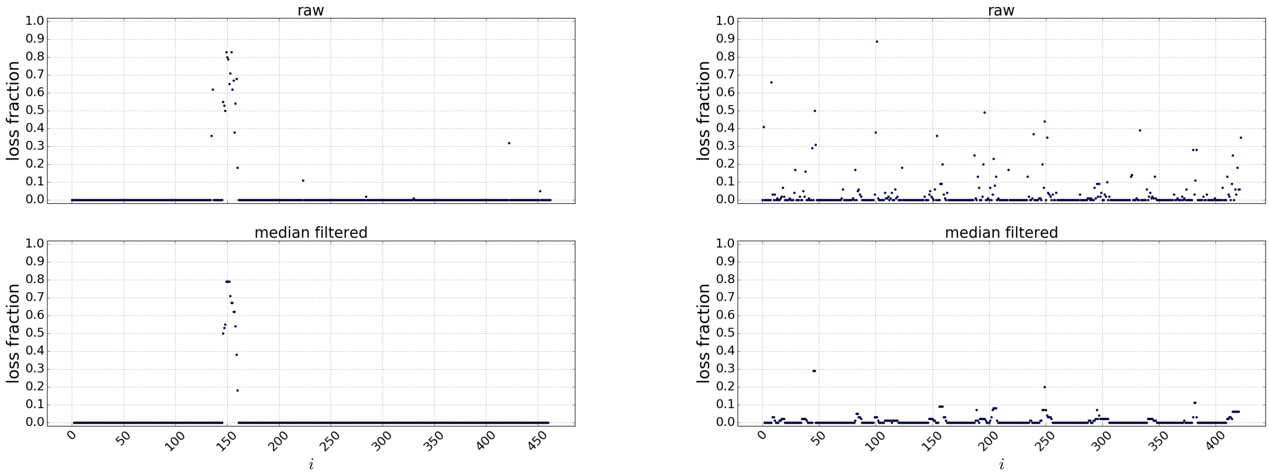
Since several time series are periodic, it was also considered the possibility of only using the trend, or the trend + residual transformation, resulted from the STL decomposition [8].

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



(a) (b)

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



(a) (b)

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

3.4 Sliding Windows

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

Algorithm 1 Online Sliding Windows

$i \leftarrow 1$
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 [9], Kolmogorov-Smirnov, and Earth Mover's Distance [10].

Figure 3.3 is an example of the online sliding windows execution. The top plot shows the RTT over time, and in the bottom plot, the (i, D_i) point represents the distance between $\mathbf{y}_{i-m:i-1}$ and $\mathbf{y}_{i:i+m-1}$. The red vertical line indicates a detected change point, and the green horizontal line illustrates the used threshold.

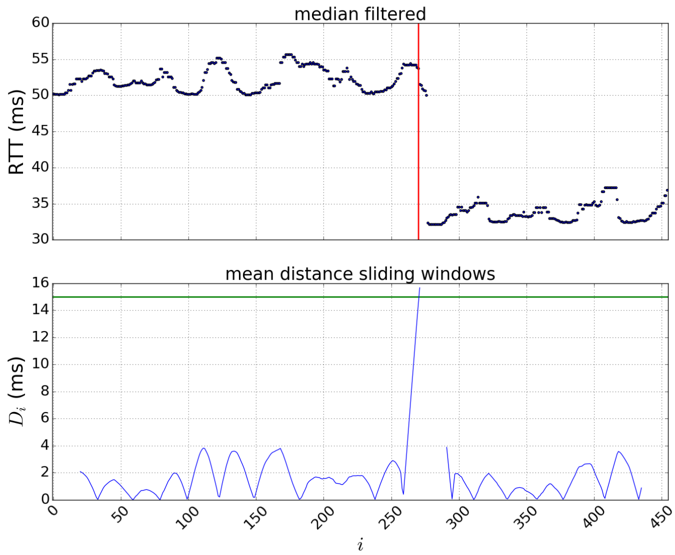


Figure 3.3: Online Sliding Windows.

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

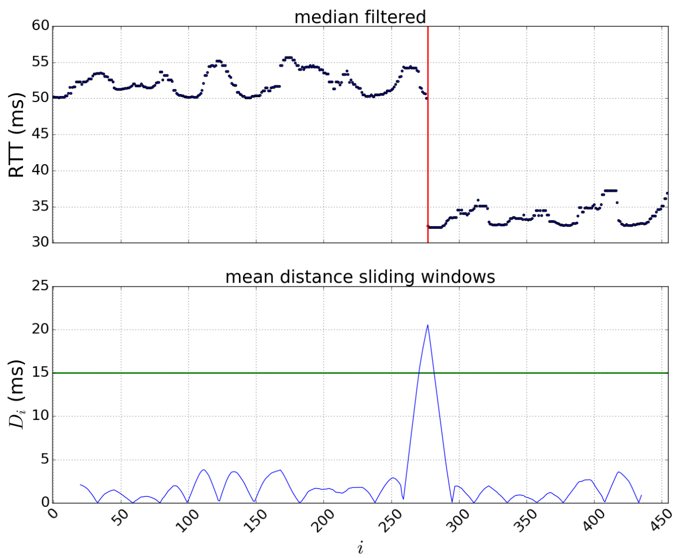


Figure 3.4: Offline Sliding Windows.

As stated in [11], a performance improvement can be achieved concurrently executing the same sliding windows algorithm with different windows lengths. This change facilitates the detection of segments with distinct number of points.

3.5 Optimization Model

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

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

Therefore, given a fixed k, the optimal segmentation is obtained through the following optimization problem, which is called the constrained case [12]:

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

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

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

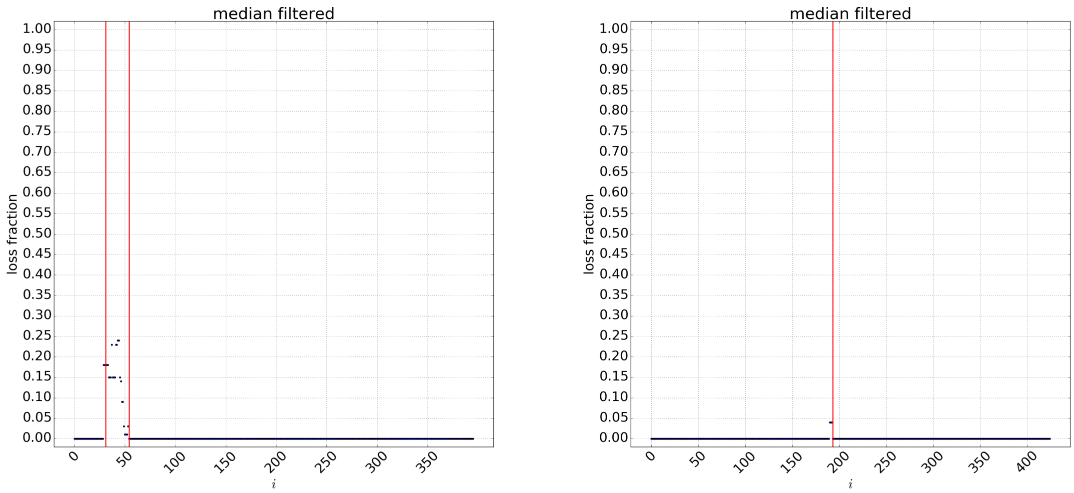
When the number of change points is unknown, an usual way is to introduce a non decreasing penalty function $g(k)$. Then, the new optimization problem, called penalized case [12], is:

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

This problem can be solved in $O(Kn^2f(n))$. However, if the penalty function is linear in k, the problem can be formulated more efficiently and solved in $O(n^2f(n))$. Also, there are several pruning algorithms to speedup the computation [12–14], in general trying to reduce the $\boldsymbol{\tau}$ search space but maintaining optimality.

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

Figure 3.5 presents two output examples.



(a) (b)

Figure 3.5: Optimization model.

3.6 HMM (Hidden Markov Model)

The idea that each segment is associated with a specific latent configuration has a direct interpretation to a HMM model [15–17]. In this context, each window is related to a hidden state of a HMM, and the observation distribution of this state represents the distribution of that segment. Therefore, the mechanism models the time series using a HMM, and through the hidden state path, assesses the times when a transition between different hidden states occur.

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

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

In [17] is stated that when using a fully connected structure, the time interval that a time series stays in the same hidden state is low, which can not reflect real data. To overcome this problem, [17] suggests to increase the time that a time series stands in the same hidden state using a dirichlet prior regularization. However, it was empirically verified that it is difficult to choose good parameters for this strategy. Instead, to surpass this issue, when using the best hidden state path, this dissertation used a HMM only as a filter, which acts as a dimensional reduction that takes into consideration temporal patterns. In this scenario, the best hidden state sequence is the input to a sliding windows method with a discrete Hellinger distance.

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

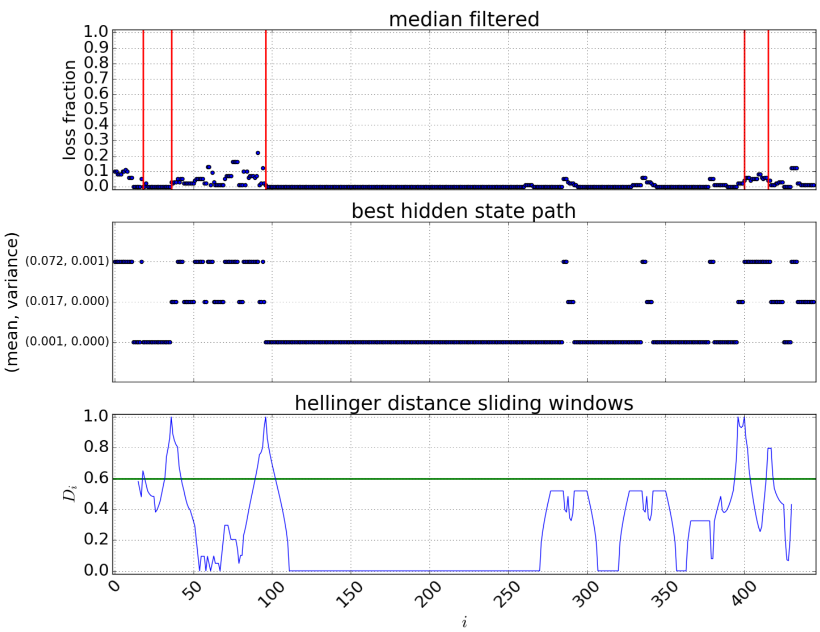


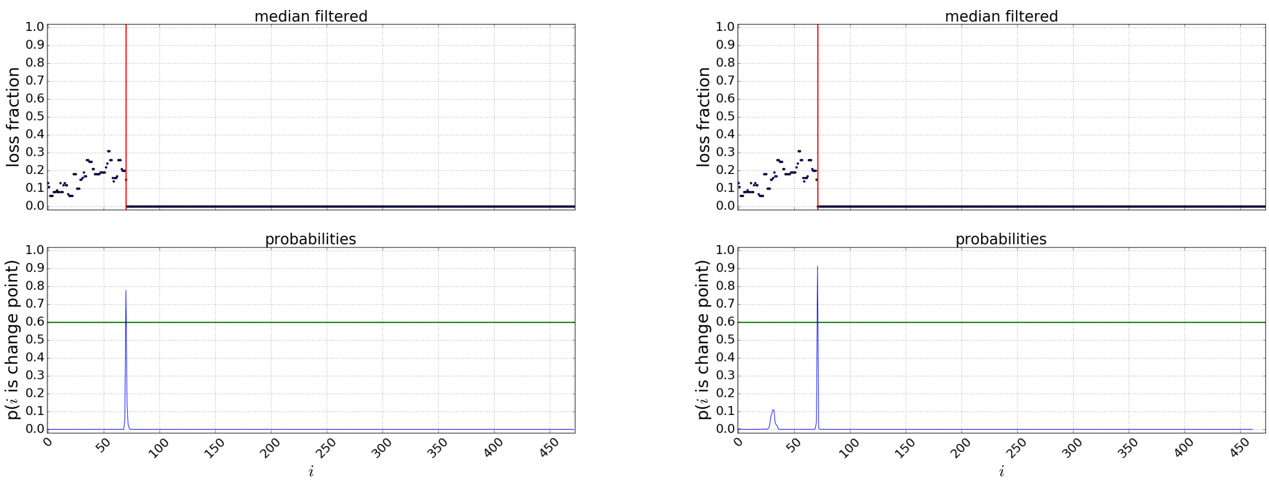
Figure 3.6: HMM filter.

3.7 Bayesian Inference

There are several Bayesian methods which aims to assess the probability that a point is a change point. Following an offline fashion, the work of [18] recursively calculates, for each i, the probability of $\mathbf{y}_{i:n}$ given a change point at i. With these probabilities is possible to simulate the time of the first change point, and then, compute the conditional distribution of the time of the second change given the first, and so on. To achieve this, the mechanism assumes that observations are independents, and that each segment is modeled by conjugate priors. Also, the procedure considers priors to model the number of changes and the time between two consecutive change points. Depending of the priors choices, the overall complexity of this method can be $O(n^2)$.

In [19] it is also considered that parameters of different segments are independents, and that data within a window is iid. However, through an online mode, the procedure is concerned with the estimation of the distribution of the length of the current time since the last change point, called run length, given the data so far observed. To achieve this, the method assumes the probability of current run length given the last run length as a prior. Assuming exponential-family likelihoods to model a segment, the time complexity to process a point is linear in the number of points already observed.

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



(a) Offline. (b) Online.

Figure 3.7: Bayesian Inference.

3.8 Final Remarks

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

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

Chapter 4

Methodology

This chapter describes the proposed data analytics workflow. Besides, the QoS data gathering procedures are briefly presented, and the proposed system is compared with previous projects.

4.1 Measurement Process

The time series used in this work represent network end-to-end measures of a cable-television infrastructure, which runs DOCSIS with asymmetric download and upload bandwidths. A home router 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 in each home router every half hour and, by the end of every day, the results are transferred to a database. The software responsible for these procedures was developed by TGR (a startup at UFRJ), and is spread over approximately four thousand customers of a major Brazilian Tier-3 ISP.

To measure the round trip packet loss fraction and the RTT between the home router and the associated server, the home router sends a train of short UDP packets, and the server bounces back them. 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 TCP data transfers from the end-user to the server. Also, the traceroutes from the end-users to the respective servers are collected with the same frequency. Traceroutes from servers to end-users are not gathered. In [20] is presented a preliminary descriptive investigation of these metrics.

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

are presented during the text, as they are needed.

4.2 Pipeline

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 further sections. The analysis seeks for events during a specific time period, in a single measurement server, and considers an 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.

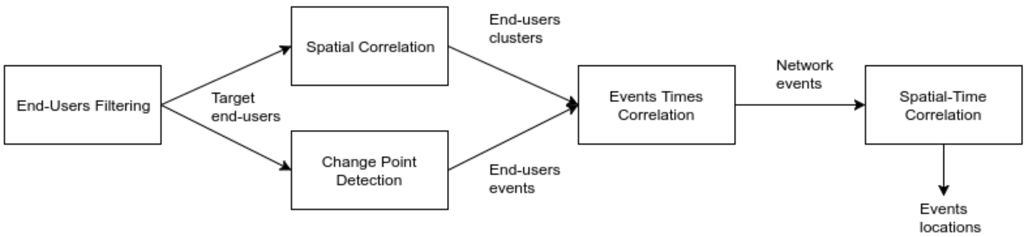


Figure 4.1: Pipeline.

The workflow starts with the End-Users Filtering step, which aims to remove 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. Also, the change points are classified in three types of events: failure, improvement, or inconclusive. For the used QoS metrics, a change point defines a failure event if the average of the window after this point is greater than the average of the window before. The improvement event is analogous, however is characterized by a mean decrease. The inconclusive event means that the segments averages are too close.

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 Events Times Correlation aims to combine similar events of different end-users. For example, if three end-users detect a failure at the same time, then this information will be 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 the events cause, 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 check 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 end-user can be categorized in two contexts. 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 infrastructure. The latter situation represents a simpler IP topology, since the Tier-3 ISP uses static routing and doesn't applies load balancing nor MPLS, techniques that are usual in the Tier-2 ISP 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 to be consumed by the Spatial Time Correlation procedure 4.5. This work doesn't have access to the ISP network topology, 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. Also, CMTSs are reported with their private IPs, since they are in the same subnetwork of the end-users. Therefore, to differentiate distinct CMTSs, the user-groups are defined by their reported IP and the first public IP that appears from their hop onward. Additionally, there are CMTSs configured to not answer traceroute pings.

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 [21]. The per destination load balancing can be disconsidered 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 example, a router can employ that packets with equal source/destination address, protocol, and source/destination port, 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 specfic 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 suppositions. Additionaly, 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 folowed by the QoS mesurements packets. Then, in this scenario, even with static paths in the Tier-2 ISP network, the correlation between traceroutes results with end-to-end metrics can reach wrong conclusions.

Therefore, this work supposes 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 in the same set of routers in the Tier-2 network. This implies that if an event occur in a Tier-2 router, it is not possible that only one of these end-users perceive the event. This is a 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. The "Tier-2" and "Server" user-groups are self explanatory. The others are represented by a tuple, in which the first field indicates the reported IP in the traceroute, and the second specifies the first hop, from the associated position onward, that reported a public IP.

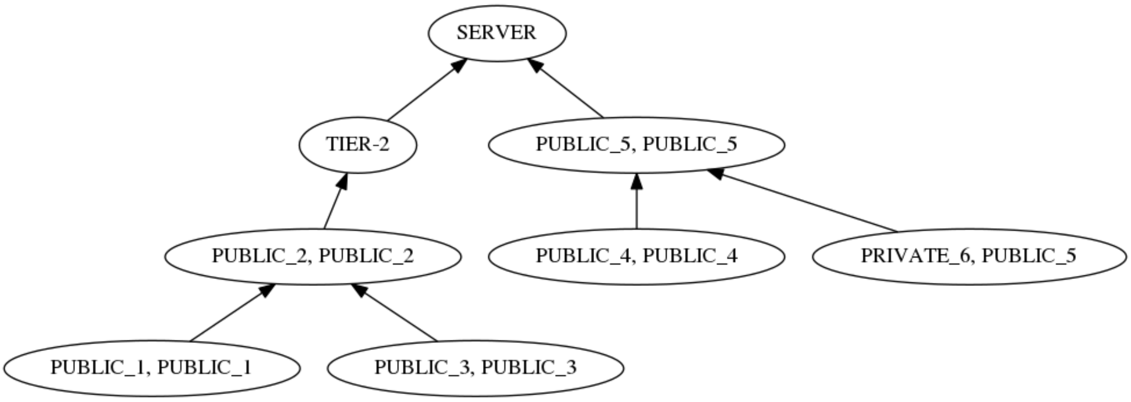


Figure 4.2: User-groups structure.

Since the links were oriented from the end-users to the server, it is possible to model this structure as a DAG. The zero in degree vertexes 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 [21]. 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 vertexes compression, it is only considered clients with static traceroutes during the specified time period. Situations characterized by Tier-2 routers that doesn't appear as consecutive elements in the traceroute are also deleted.

4.4 Events Times 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 detect the same network event, such as an equipment failure, at different times in different clients. The first one is related to the fact that the time series are not regularly sampled. The second one 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 point location, a detected change point is transformed into an interval according to a time tolerance. Therefore, given a parameter δ, a change point identified at time t implies in the existence of a change point within $[t-$

$\delta, t+\delta]$. To be consistent, change point algorithms must report locations 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: sweeping time 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. An end-user 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. As an example, not necessarily selecting the minimum number of network events that cover all end-user events is the most appropriate answer.

To this goal, it was developed a heuristic inspired in the Inexact Voting in Totally Ordered Space problem [22]. 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

Let l be the end-users events of a specific type sorted by time
while l is not empty **do**
    Select the interval d with the greatest number of end-users events, in which all these events are at most 2δ apart. In case of ties, select the left most one
    Report the mean time of the d extremes as the network event time
    Remove all end-users events from l that are in d
end while

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

A more straighforward solution, that was not used in this work, is to create regular time-bins, and interpret all end-users events that occur at 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 to different end-user events, associated with this network event, be located in different bins.

If co-occurrent events with 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 reduce with the increase of the time series sampling frequency.

4.5 Spatial-Time Correlation

This step matches network events from different user-groups to define a set of feasible events locations.

It is not possible to check if changes in round trip metrics, such as RTT, are caused in the path from the end-user to the server or vice versa. Therefore, correlate those metrics with the available traceroutes can lead to erroneous or inconclusive outcomes. 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's 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.

The method expects that a change point detection algorithm setup is able to detect all end-users affected by a network event, or no end-user is identified. Besides, with exception of the Tier-2 user-group, the mechanism supposes that if an event occur in an user-group, all traffic that goes through this 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 hypothesis.

In the first reasoning part, it will be processed cases without the presence of the Tier-2 user-group. Also, it will be removed traceroutes paths that doesn't start in a zero indegree vertex. Additionally, it will be disconsidered co-occurrent events with the same type. Later, these restrictions will be removed and treated separately. Unless stated differently, in the rest of this section is considered an unique network event in the period of study.

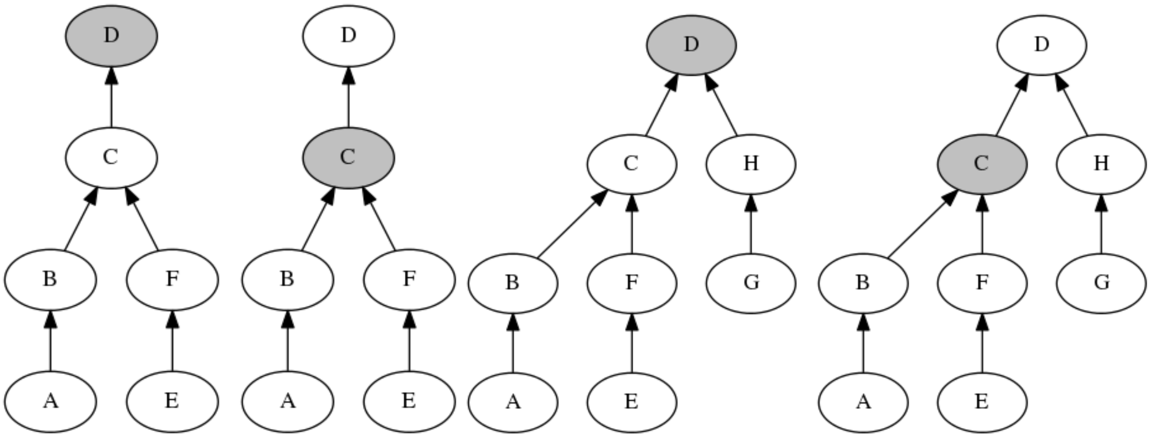
The procedure starts analysing zero indegree vertexes. Suppose that only a proper subset of the clients that belong to a zero indegree node perceive the event. In this situation, it is possible to afirm that this vertex is not the cause of the event, since not all clients detected it. For the same reason, the cause is not located in posterior nodes in the path to server. Then, if this set size is bigger than one, these end-users must share a network equipment, which is the cause, that doesn't respond to traceroute pings, and 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 can 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 above 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 the event is surely located in the first hop or before. Otherwise, the procedure continues to the next hop.

When applied to different zero indegree vertexes, this method results in possible events locations that are prefixes of the paths to the server. However, different paths share common network equipments, then 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 dont't belong to them and still perceive the event.

Figure 4.3 presents several topologies and different network events. The gray vertexes indicate a network event that occurred in this location.



(a) (b) (c) (d)

Figure 4.3: Network event location examples.

In Figure 4.3a, the analysis starting at vertex A results in the following feasible event locations, {A, B, C, D}. The reasoning through 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 analyis made through the zero indegree vertexes result 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 paths 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 established restrictions. To overcome this issue, when analysing 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 perceived the event.

Now, suppose paths that doesn't start in zero indegree vertexes, which should be processed after the analysis explained above. 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, that doesn't answer to traceroute pings and is the cause.

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

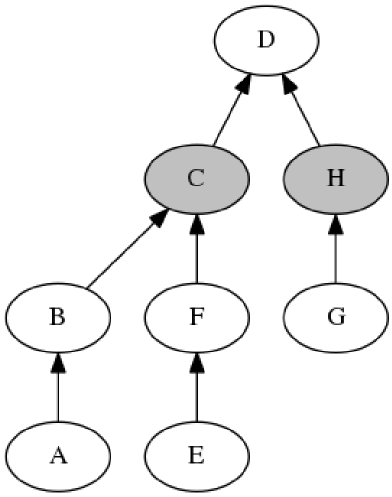


Figure 4.4: Co-occurent events.

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}. Correlating the paths results leads to the location D, which is a mistake. Therefore, when considering co-occurrent events with the same type, the step of correlating the conclusions of different paths must not be applied, and the union of these results should be the final outcome.

From this procedure, it is possible to formulate a problem to define the minimum number of end-users to be monitored, and their placement to localize events with the minimum number of feasible locations.

4.6 Change Point Detection Issues

As stated in Chapter 3, one of the main issues of this work is the algorithms and parameters selection. In general, this process requires a dataset to enable the evaluation of an algorithm setup.

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 [23]. 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 [17]. However, this can introduce unreal 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, whithout even analyzing the time series. However, in the application domain of the present work, this approach would be impractical. First, this would need the expertise of how the configurations of network topology, routers congestion, physical equipment problems, among other features, affect the different end-to-end QoS metrics. Second, this kind of information is absent in the dataset, and would be too complex to collect it.

Another way is to use visual annotations, as it was done in [24]. Also, manual labeling is usual for anomaly indentification in Interner traffic traces [25]. 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 [26], 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. It was only presented raw loss fraction time series with 10 days of data. Also, it was selected the ones that have at least 85% of the maximum possible number of points during the specified period, considering that data is sampled at most two times in a hour. Change points can be interpreted as rare events in this dataset, and several data streams have almost all measures with zero losses. Therefore, to increase the entropy, it was selected time series that have 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.

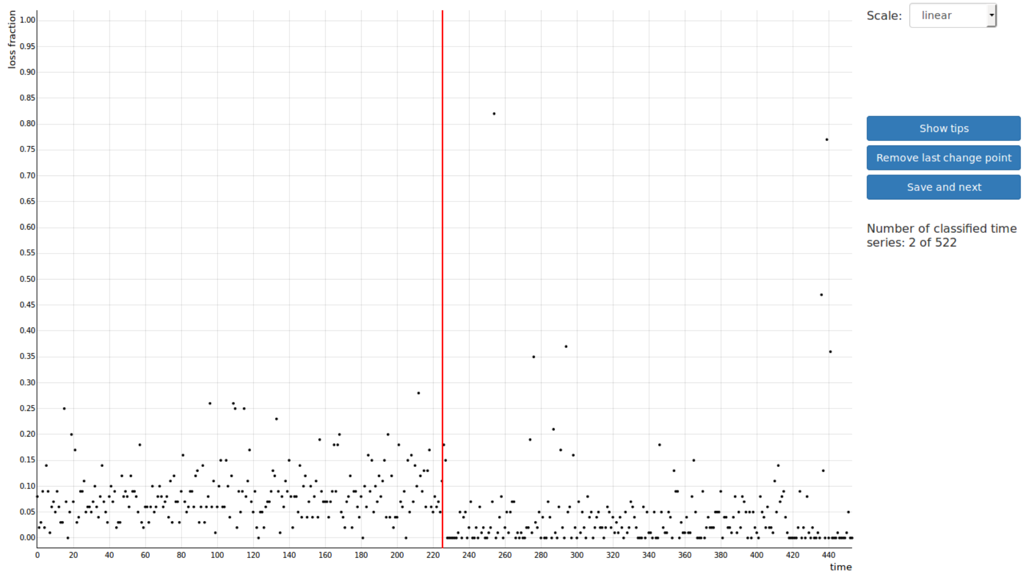


Figure 4.5: Survey system snapshot.

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 Events Times Correlation procedure 4.4. Therefore, it was considered that each specialist voted to a set of change points positions. The time tolerance was set to 7 hours, which means 14 consecutive points.

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.

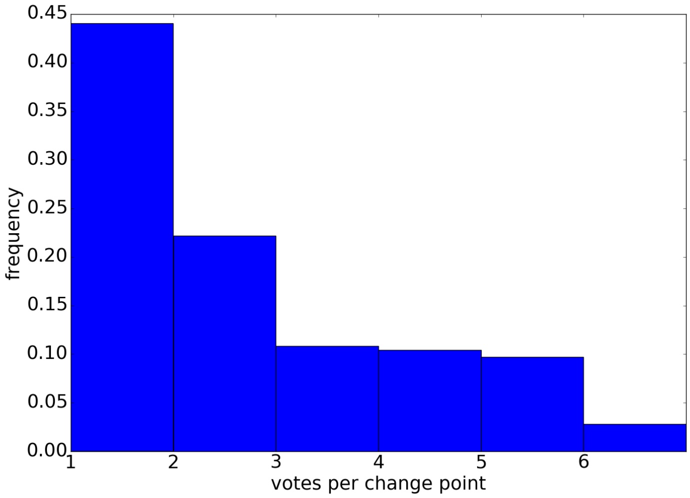
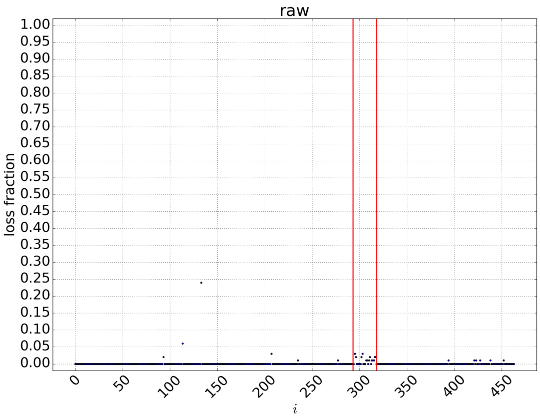


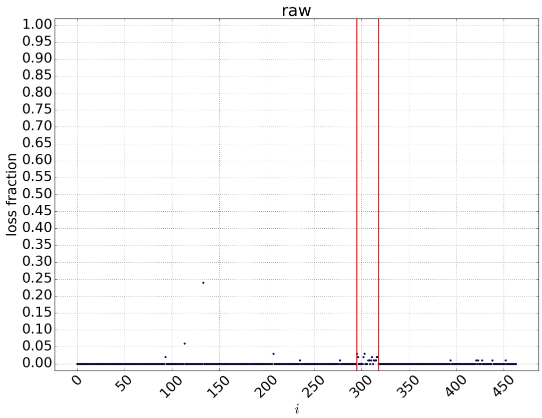
Figure 4.6: Number of votes per change point histogram.

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.

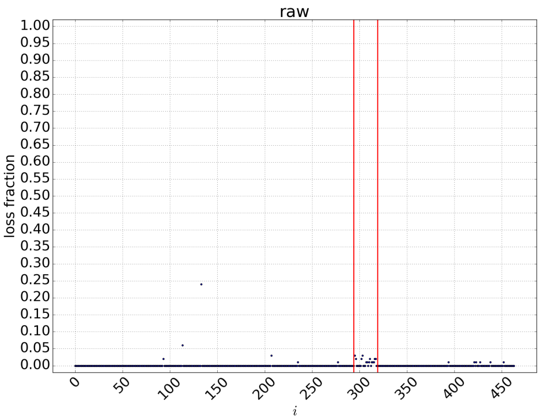
In Figure 4.7 is presented the specialists classifications in a time series with a high level of agreement on the change points locations.



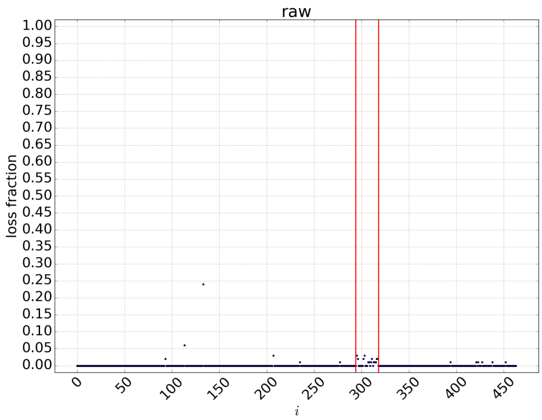
(a) Specialist 1



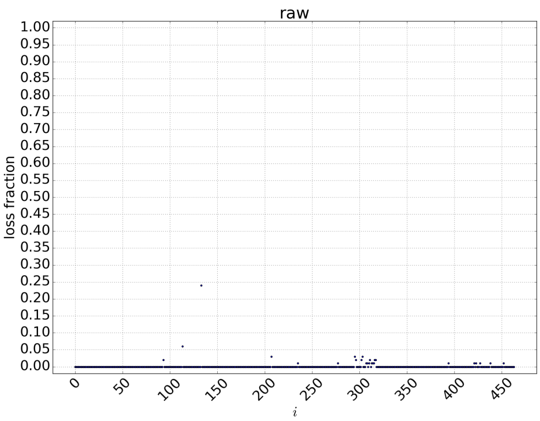
(b) Specialist 2



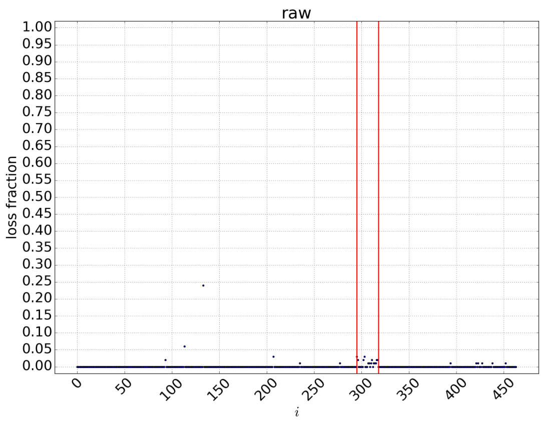
(c) Specialist 3



(d) Specialist 4



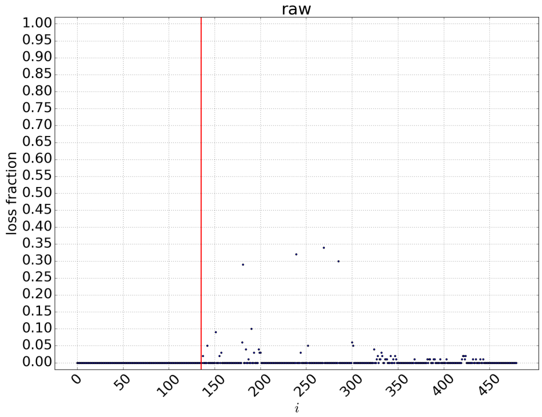
(e) Specialist 5



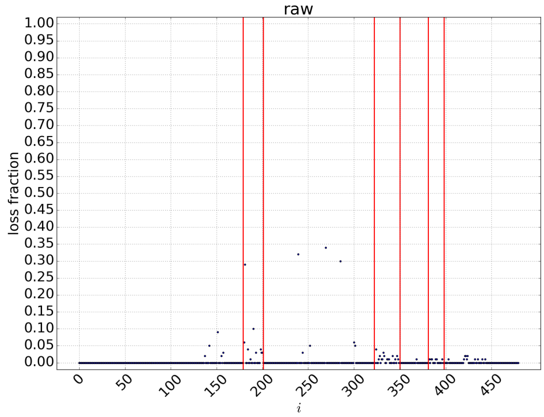
(f) Specialist 6

Figure 4.7: Classifications agreements.

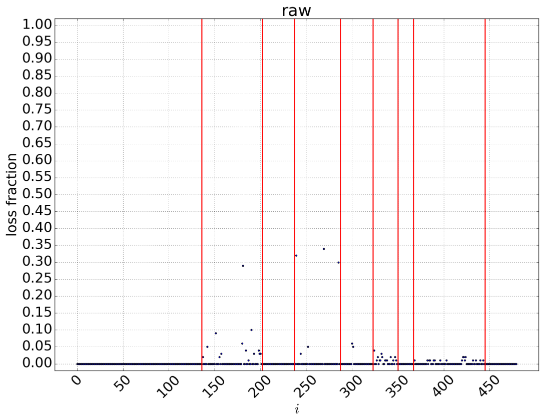
Figure 4.8 shows a time series with several disagreements. It was verified that this case is the most representative in the constructed dataset, fact that corroborates with the problem subjectiveness.



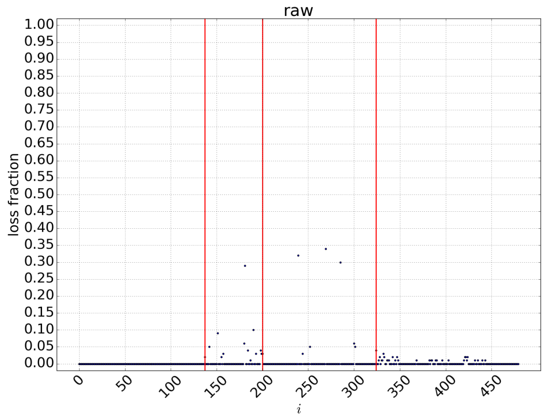
(a) Specialist 1



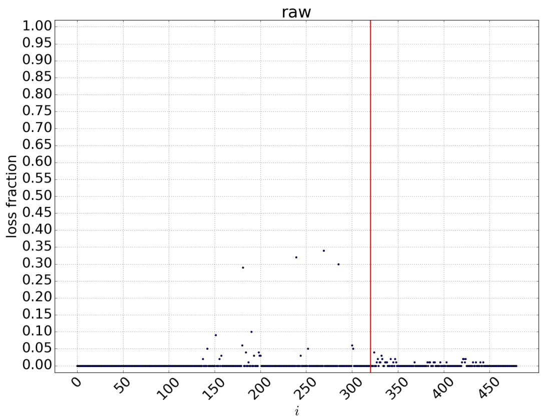
(b) Specialist 2



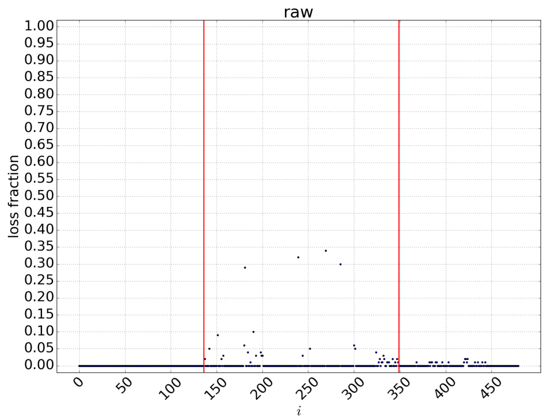
(c) Specialist 3



(d) Specialist 4



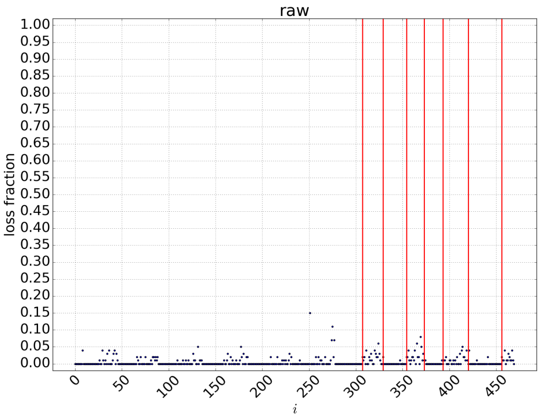
(e) Specialist 5



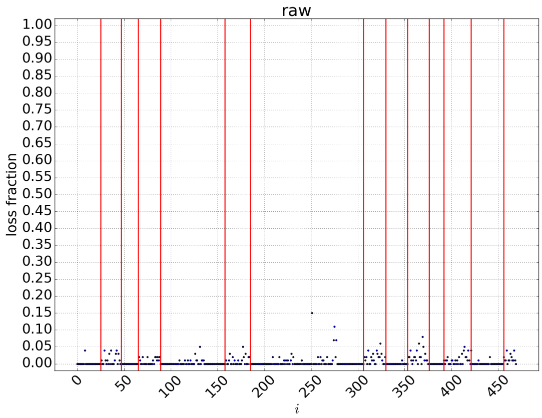
(f) Specialist 6

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 to Previous Systems

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 project, since the finer granularity requires more time series to be spread in different network locations, and the current dataset has a low number of end-users.

To detect faults, Argus applies an anomaly detection procedure, but the present work uses a change point detection method. 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 it's standard. The change point detection only seeks for points where the statistical properties change, and doesn't take into consideration a standard time series behaviour.

Additionally, beyond the network edge point of view, Argus and NetNorad uses some internal network information, which is absent in the present work.

Chapter 5

Results

Chapter 6

Conclusions

Bibliography

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

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

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

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

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

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

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

[8] 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.

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

[10] 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.

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

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

[13] 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.

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

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

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

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

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

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

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

[21] 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.

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

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

[24] 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. Disponível em: <http://jmlr.org/proceedings/papers/v28/hocking13.pdf>.

[25] 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.

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

[27] SUNDARESAN, S., DE DONATO, W., FEAMSTER, N., et al. “Measuring home broadband performance”, *Communications of the ACM*, v. 55, n. 11, pp. 100–109, 2012.

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

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