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

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

Diego Ximenes Mendes

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

Orientador: Edmundo Albuquerque de Souza e Silva

Rio de Janeiro
Janeiro de 2017

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

Diego Ximenes Mendes

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

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2017

Ximenes Mendes, Diego

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

VIII, 19 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. 17 – 19.

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

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

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

Diego Ximenes Mendes

Janeiro/2017

Orientador: Edmundo Albuquerque de Souza e Silva

Programa: Engenharia de Sistemas e Computação

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

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

Diego Ximenes Mendes

January/2017

Advisor: Edmundo Albuquerque de Souza e Silva

Department: Systems Engineering and Computer Science

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Contributions

1.2 Dissertation Outline

Chapter 2

Literature Review of Fault Localization in Computer Networks Using End-to-End Measurements

This chapter briefly describes three projects (Argus, NetNorad and CEM) that deploy end-to-end measurements to localize faults in computer networks.

2.1 Argus

In [1] is presented Argus, a system to detect and localize QoS issues in ISP's networks. To achieve this goal, it uses end-to-end data and network global information, such as traffic passively monitored at end-users, the ISP network topology, routing tables, and geographic information. Argus's infrastructure is defined by the pipeline presented in figure 2.1.

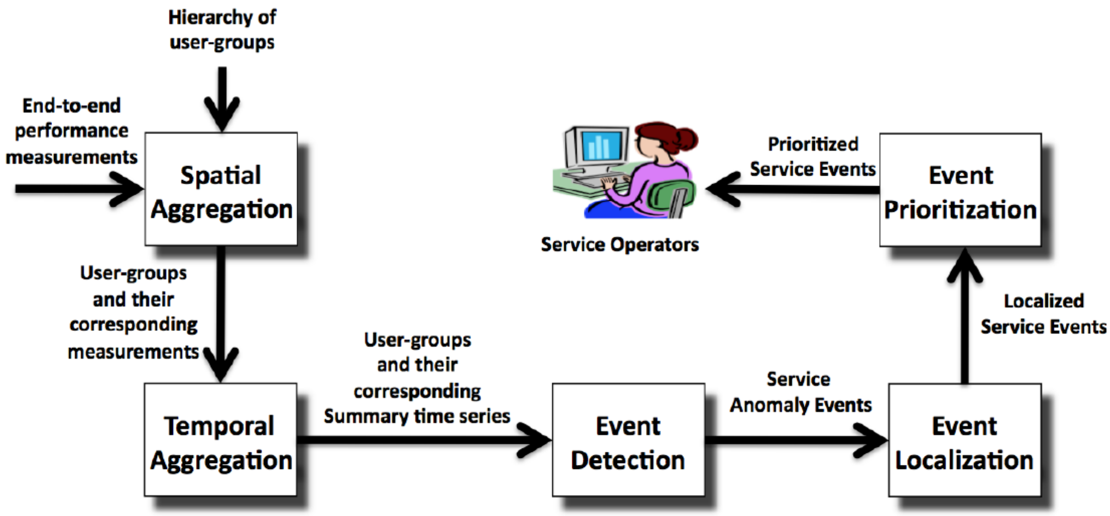

Figure 2.1: Argus pipeline.

The system operation starts with the spatial aggregation procedure, in which end-users are clustered into user-groups. This step avoids keeping track of the end-to-end performance metrics associated with millions of individual end-users, which improves the system's scalability. Each user-group is characterized by a set of end-users that share some common attributes, such as BGP prefix or AS. The used attributes imposes the possible fault locations to be infered. An example of an agreggation is presented in figure 2.2.

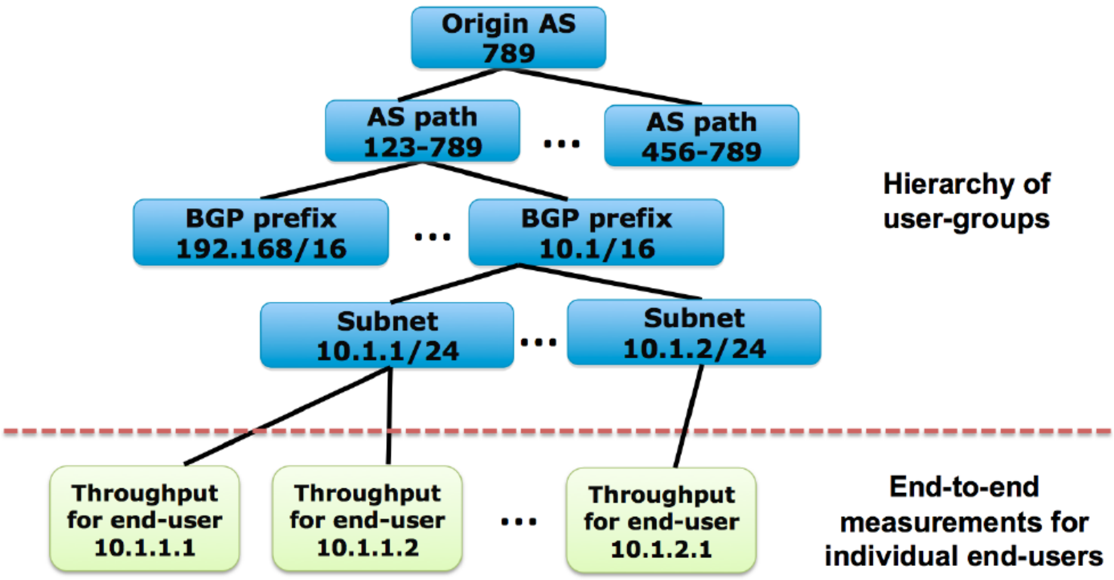

Figure 2.2: Argus spatial aggregation.

The temporal aggregation phase determine how the performance metrics collected from different end-users in a end-group are aggregated. In this procedure details about individual end-users are lost, but the goal is to detect events that impact the user-groups. Instead of analysing the raw individual end-to-end metrics, Argus focus in summary statistics, such as, percentiles, min, max, of the distribution, and this can reduce the noisy from raw data, since a end-group have data from different end-users. For each user-group the measurements of all end-users are aggregated by time-bins, and for each time-bin a summary statistics is selected, forming then a summary time series. Different statistics provides different advantages to track cerain type of issues, for exaple the min statistics applied to RTT can capture the physical propagation delay while the average can capture network congestion. Argus uses median as the default summary statistics, since it was found that empirically, is effective in tracking service or network issues while being robust to variability to individual end-users due to their local architecture.

The event detection procedure applies algorithms to detect anomalies in the summary time series. Argus uses a Holt-Winters variation which is an online procedure with low runtime and memory complexities, which is essential to the scale of the system.

Through spatial and anomaly event time correlations Argus is able to localize fault locations in the network in the event localization procedure, however the detailed procedure of how this process is defines was not published.

After the problem localizations the events are sorted according with their significance, which is defined by a metric obtained through the event detection algorithm, and considering the number of end-users affected by the event.

Argus was applied to RTT measurements in a CDN hosted in a tier-1 ISP. During a one month period using time-bins of 1 hour, Argus detected 2909 anomaly events, and in general, lower level user-groups were more responsible for these anomalies than the higher level groups. For each type of user-group, only a small fraction of the end-users are responsible for the anomaly events and the majority of the anomalies are very short in duration, in which 90% of the anomilies last for ar most 1 hour, which was the used time-bin granularity.

The following inferences are not presented in the paper. The fact that only a small number of end-users are responsible by the anomaly in the summary time series indicates that fault isolation could achieve higher precision using the spatial aggregation with finer granularity. It is not clear G-RCA accuracy, then I don't know if was made a Argus evaluation accuracy. Since the fault localization procedure is not presented, it is not possible to check which topologies types can be used.

2.2 NetNorad

NetNorad [2] consists of a Facebook's internal project to automate the detecion and mitigation of network faults in the Facebook's network. It is claimed that a human-driven investigation may take minutes or hours. Also more traditional used network monitoring techiques such as querying device information through SNMP or via command line interface can not encounter cases known as gray failures, in which the devices can not properly report its own malfunctioning. "Or the problem can be associated with the network structure"

NetNorad is mainly based in end-to-end measurements. Facebook's servers ping each other, in which a pinger sends UDP packets to responders, and the latter receive, timestamp and send the packet back. The process happens in turns, in which each pinger sends packets to all of its targets, collects the responses, and then repeats the procedure.

Facebook's network is structured hierarchically. At the lowest level there are servers in hacks, which are organized in clusters. A collection of clusters in the same building and serviced by a common network form a data center. The data centers are aggregated via a network that interconnects them within the same region and attaches to th e Facebook global backbone network. This infrastructre spreads across multiple regions around the world. Figure /ref presents an example of this architecture.

ADD IMAGE WITH THE TOPOLOGY

A small number of pingers is deployed in each cluster, but reponders are placed on all machines. All pingers share a single global target list, which consists of at least two machines in every rack. As with Argus, the system applies aggregation techiques, however, the imposed hierarchical structure of Facebook can simplify the procedure . When a pinger receives the responses in a given round, it aggregates results for machines that belong to the same cluster and tags them based on their location. Tags are based in the following rules: "DC" if the target cluster is in the same data center of the pinger, "Region" if the target cluster is outside the data center but within the same region, "Global" if it is outside the pinger's region.

Each cluster have three time series reflecting different viewpoints, one for the same data center, other for the same region, and other for the global network. For each time series the system tracks percentiles over 10-minute intervals. Tracking multiple percentiles allows the identification of the nature of the events in the network. For example, a packet loss spike at the 50th percentile means that there is likely a failure affecting the majority of traffic into or out of a cluster, while a peak at the 90th and not at 50th percentile would indicate that there is a high level of loss affecting a small number of targets. For each percentile and proximity tag is defined two thresholds, one for trigger an alarm and other for clear an alarm. This infrastructure allows alarms to be raised about 30 seconds far from the event.

Metrics are collected from an end-to-end perspective, therefore it is necessary to distinguish if the events are caused by an end-host failure or a network fault. The pinger applies an outlier detection, discarding targets that reports too high packet loss relative to their general population. Also the same logic is applied to discard reports from pingers that have anomalous pattern according to their population.

To localize the fault the following correlation analysis is applied. If loss to a cluster is reported at data center, region and global tags, then the fault is probably located at the cluster data center. If all clusters within a data center report packet loss, then the issue is likely to be a layer above the clusters. These rules doens't determine the exact fault location, however a Facebook tool called fbtrcrt, similar to the UNIX traceroute tool, improves this analysis. fbtracert explores multiple paths between two endpoints, and can analyse the packet loss at every hop, correlating the resulting path data to find the common failure point. When fbtracrt is unable to find the failure, then there is a human involvemnt to find it.

The following inferences are not presented in the paper. The work doesn't presents an accuracy analysis.

2.3 CEM

In /cite is proposed a framework called Crowdsourcing Event Monitoring (CEM), which aims to realize online detection (within seconds or minutes), of service-level events through monitoring software that runs inside or alongside applications on the end systems where they are used.

In the framework, each end-host uses its own passively gathered performance information to detect local problems as potential network events . The framework doesn't care how these events are detected, as long they correspond to service-level problems. Concurrent events ocurring in multiple performance metrics of a service (e.g., download and upload rates), increases the confidence that the problem is independent of the service. After detecting the events a end-host pushes them to a distributed storage to further analysis. The fact that the events detection procedures are realized at the end-hosts increases the scability of the system.

To isolate the scope of the network events, multiple locally detected events are correlated from the same network region. This correlation is made in a central fashion. The first problem to tacked is to reason if concurrent events are caused by a network fault. There are several reasons to different hosts experiences concurrent events. One is the target of the work, which is a network fault. Another one is that the service can experience a problem not caused by the network, for example, a high volume of requests in a web service. Also it is possible that concurrent events occur only by chance, for example, users experience interference on separate wireless routers. To solve this problem the framework provides a statistical model to determine if concurrently events are a coincidence or not. This model takes in consideration service-specific dependencies and the rate of observed local events ocurring at the same the same time in a network. In this model, the confidence in a detected event being due to a network increases with the number of hosts detecting the event, and with the increase of independent performance metrics indicating the event. To isolate the problem the framework uses structure information about the network and the hosts geographic locations, however, ther work doesn't provides any details of how this correlation is made.

CEM was deployed and evaluated in a P2P system, using BitTorrent traces collected from users of the Ono plugin for the Vuze BitTorrent client. The detected events by the system was compared to aground truth of network events gathered from public available event reports of ISPs.

The following inferences are not presented in the paper. CEM provides a high level framework structure, however lack for several details in the deployment procedure.

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 data, and some adopted solutions which are not described in the literature.

3.1 Problem Definition

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

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

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

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

3.2 Notation

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

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

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

3.3 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 sliding windows method.

Algorithm 1 Sliding Windows

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

In this mechanism, when the distance between the distributions is above some threshold α a change point is reported. This is a common approach for an online application, however, it is possible to increase the classification accuracy in offline cases. As an example, the top plot of Figure 3.1 presents a simulated time series. The segment $\mathbf{y}_{1:1000}$ was generated sampling a $N(1, 0.2)$ distribution, and $\mathbf{y}_{1001:2000}$ was sampled through $N(5, 0.2)$. The distribution of a window was constructed binning the data with bins of size 0.02. The bottom plot of the same figure presents the associated Hellinger distance [3] between two sliding windows, where the point (i, H_i) represents the distance between the windows $\mathbf{y}_{i-100:i-1}$ and $\mathbf{y}_{i:i+99}$.

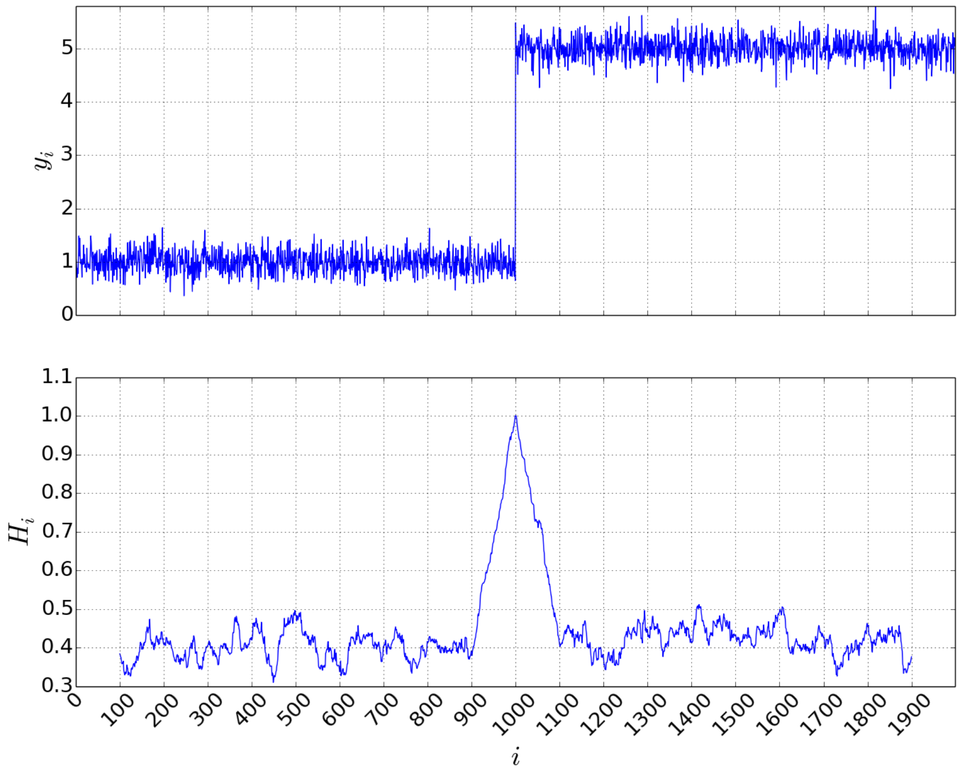

Figure 3.1: Toy example of a sliding windows method.

It can be observed that there is a peak on the distance in the exact location where the distribution changed. However, using only the threshold method it is possible to prematurely infer the position of the change point. Therefore, an alternative is to also use a peak detection algorithm. Besides, the distance function choice has a direct impact on the classification accuracy.

As stated in [4], a performance improvement can be achieved concurrently executing the same sliding windows algorithm with different windows lengths. This change which facilitates the detection of segments with distinct number of points.

3.4 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 [5]:

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

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. Two approaches can be used 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 [5], 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 [5–7], in general trying to reduce the $\boldsymbol{\tau}$ search space but maintaining optimality.

3.5 HMM (Hidden Markov Model)

The idea that each segment is associated with a specific latent configuration has a direct interpretation to a HMM model [8–10]. 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 [10] 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, [10] suggests to increase the time that a time series stands in the same hidden state using a dirichlet prior regularization.

3.6 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 [11] 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 $O(n^2)$, considering that the likelihood of a segment can be evaluated in $O(1)$.

In [12] 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.

Chapter 4

Methodology

Chapter 5

Results

Chapter 6

Conclusions

Future work: - collect datailed topology from the ISP which will enable more precise localizations - previous problems database and end-users reclamations to the ISP call center. This can enable the problem be interpreted as a supervised problem. - model the system as a reinforcement learning procedure, in which operators can feedback the system with correct/mistakes in the detection and localization of problems, then the system can be able to adapt automatically chossing the best algorithms and hyperparameters. - active increase mesuremet frequency in locations with potential problems, avoiding incorrect classifications and decreasing the detection time. - choose best network metrics that can increase the system performance - if a more data is available try space aggregation techniques, as in Argus - correlate path change in traceroute with changes in end-to-end metrics - the proposed mechanism can also be expanded to detect attacks on networks - is possible to detect and localize problems only with data collected passively?

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

[4] 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>.

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

[6] 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.

[7] 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>.

[8] 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>.

[9] 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>.

[10] 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>.

[11] 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>.

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

[13] 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>.

[14] 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>.

[15] 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`>.

[16] 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`>.

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

[18] 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`>.

[19] 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`>.