



**HAL**  
open science

# Leveraging Web browsing performance data for network monitoring: a data-driven approach

Imane Taibi, Yassine Hadjadj-Aoul, Chadi Barakat

## ► To cite this version:

Imane Taibi, Yassine Hadjadj-Aoul, Chadi Barakat. Leveraging Web browsing performance data for network monitoring: a data-driven approach. GLOBECOM 2022 - IEEE Global Communications Conference, Dec 2022, Rio de Janeiro / Hybrid, Brazil. pp.1-6, 10.1109/GLOBECOM48099.2022.10001139 . hal-03763839

**HAL Id: hal-03763839**

**<https://inria.hal.science/hal-03763839v1>**

Submitted on 29 Aug 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Leveraging Web browsing performance data for network monitoring: a data-driven approach

Imane Taibi, Yassine Hadjadj-Aoul  
Université Rennes 1, Inria, CNRS, IRISA, France

Chadi Barakat  
Inria, Université Côte d’Azur, France

**Abstract**—Monitoring network performance becomes crucial today since it allows content providers to ensure a good quality of their services by identifying the root causes of service degradation. Also, it gives the end-user a better understanding of the performance they have (state of the networks). A widely used monitoring technique involves performing measurements from within the browser in an effort to capture the network status as close as possible; we talk about Web-based network monitoring. Many Web measurement tools have recently been proposed, however, most of these tools either have a high computational cost or exaggeratedly consume data. In this paper, we propose a lightweight solution able to estimate the underlying network status accurately and perform Web troubleshooting in order to detect anomalies. We develop and implement a distributed system that collects measurements at both levels: browser and network. Then, we build an original network monitoring framework based on Bayesian Gaussian Mixture Models (BGMM) coupled with an algorithm to detect in real time the occurrence of anomalies. We follow a browser-based passive measurement and data-driven approach to derive our inference models, which leads to an efficient Web browsing troubleshooting solution.

**Index Terms**—Network measurement, Web browsing, deep learning, clustering, network monitoring.

## I. INTRODUCTION

Web browsing is one of the most dominant activities on the internet that allows accessing a massive amount of services. In other words, the Web has become today a principal pillar of the information age. However, the accelerated and continuous growth of the Web traffic can still lead to unexpected and unwanted performance behaviours that negatively impact the quality of service delivered to the customer, thus leading to session and even service abandonment. Therefore, content providers need to ensure a good quality of their services. As for end-users, it is important to know about the actual state of their network, especially when the quality of service drops, so that they are able to take the appropriate actions towards improving the quality of their communications.

It thus becomes crucial to monitor the Web and network performance in order to detect anomalies and bottlenecks in the underlying network and identify the root causes of performance degradation. Web browsing can give us hints on the network performance since Web pages are loaded with different network conditions over time. So detecting anomalies as well as any significant change of the network state relies on finding a way to separate these conditions from each other, cluster them, and track them over time.

Understanding the relationship between Web performance and the actual network state is key to network troubleshooting. However, given the complexity of the Web [1], this task is very challenging [2]. Indeed, today’s Web pages incorporate plenty of objects fetched from multiple servers through multiple connections and use complicated rendering technologies with advanced underlying protocols such as HTTP 1.1, HTTP/2 and QUIC (to add to transport and mac protocols). Furthermore, choosing the Web metrics that faithfully reflect the end-user Quality of Experience (QoE) is another complex task [3]. For example, the PLT (Page Load Time) is a widely used metric. However, recent studies deduce that this metric alone cannot accurately estimate Web browsing quality. Consequently, there were several research works to find more suitable metrics that are closer to page rendering and are more end-user oriented, such as SpeedIndex and Above The Fold (ATF) [4].

Web-based network monitoring, that is about establishing the link between Web and network performance, involves performing measurements from within the browser in an effort to capture the network status as close as possible. Many Web measurement tools have been proposed to conduct measurements from within the browser itself, like Fathom [5], a Firefox plugin, or Speedtest.net [6]. These approaches have a significant computational cost or exaggeratedly consume data. In fact, existing tools usually implement an active approach with traffic injection in order to guarantee high precision of network measurements, such as latency and bandwidth, which incurs cost on the network and adds significant overhead; this additional measurement traffic will very likely disturb the normal network conditions which existing tools try to measure.

In this paper, we propose a lightweight solution for monitoring network performance that deploys a browser-based passive measurement and data-driven approach, able to provide a very good estimation of the underlying network performance. Our solution that we introduced in an earlier work [7], [8] leverages the wealth of passive measurements available within the browser to estimate the underlying network conditions with the help of machine learning. In the present work, we consolidate this solution and builds upon it towards differentiating between the different network conditions that face the Web pages visited by the user, which is essential for network anomaly detection and Web browsing troubleshooting. Our main contributions here are (i) a distributed system that collects measurements at the browser level and the network level, (ii) an original network monitoring framework, based on

Bayesian Gaussian Mixture Models (BGMM), able to provide information on the underlying network state for the different visited Web pages by the user, and (iii) an algorithm to detect in real time the occurrence of anomalies and identify the Web pages that are affected by them, thus leading to an efficient Web browsing troubleshooting solution.

The rest of the paper is organised as follows. In section II, we first provide a background on network performance monitoring. In Section III, we describe in detail our approach for Web-based network monitoring using BGMM clustering. In Section IV, we present the platform implementation, then we study the efficiency of our monitoring solution in detecting anomalies. We conclude the paper in Section V.

## II. BACKGROUND

Network performance monitoring is an active area of research encompassing a significant number of techniques. One of its goals is to assess the network state and detect abnormal behaviours deviating from what we consider normal. Performance stability, improvement, or degradation are all possible outcomes of monitoring. Considerable efforts have been made to address this issue in the literature. Many of these solutions focus on solving the problem in specific domains, by leveraging the power of statistical and machine learning techniques [9], [10], [11]. In fact, a basic network anomaly detection system monitors the performance behaviour of the underlying network and collects the metrics needed to create baseline models of typical network behaviour [12], [13]. It continually analyses relevant information deviations to reveal performance anomalies, and performs root-cause analysis to pinpoint the associated bottlenecks.

Network monitoring is challenging because of the scale and dynamics of today’s internet infrastructure. It is thus difficult to tell the difference between regular and abnormal scenarios since the line between the two is often blurry and shifts over time. To enable this distinction, unsupervised learning is often used [14], it consists in clustering the network data according to a set of features, and in identifying those clusters that diverge from the dominant clusters representing the normal behaviour. Here, we follow this approach.

Data clustering can be defined as a method of analysis that involves studying a set of data that is not labelled and grouping it into coherent and homogeneous subsets (i.e., clusters). We focus in our work on two popular clustering techniques, K-means and Gaussian Mixture Models (GMM). For GMM, we consider two variants, the classical GMM variant and the Bayesian GMM (BGMM) one. We will give further details on the models later in the modelling part.

## III. WEB-BASED NETWORK MONITORING USING BGMM CLUSTERING

Unlike traditional network monitoring methods based on active measurements, our goal is to develop a lightweight solution that deploys a browser-based passive measurement approach. We then use pre-calibrated machine learning models

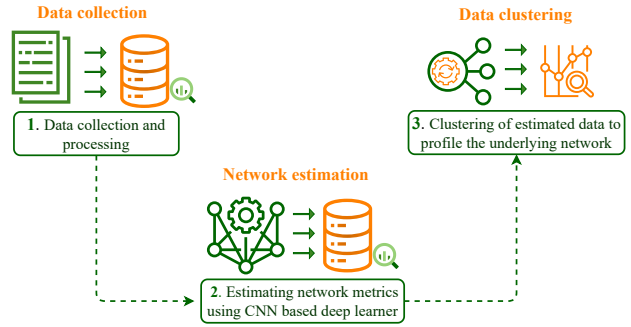


Fig. 1: Our approach in three phases

to bridge the gap between the browser-level measurements and the network performance.

In real-life, Web browsing traffic run in different network conditions that we don’t know. Discovering these conditions and tracking them over time is an important step towards understanding the underlying network and detecting the prevalence of anomalies, targeting part or all Web browsing activity. For example, an overall change of network performance is a sign of local network problem. A part of the traffic being degraded is a sign of a remote problem specific to the impacted websites, rather than a general access problem. Given these observations, we propose here a Web-based network monitoring framework capable of inferring the underlying network conditions and identifying network anomalies for the different visited Web pages. As shown in Fig. 1, our approach consists of three main phases. First, we collect Web measurements from the browser, such as PLT and other Web page characteristics (see Table I). Next, in order to estimate the network state from the Web performance metrics and the page characteristics, we rely on the results of our earlier work [7], where we propose a convolutional neural network (CNN) model that has as input the different Web metrics and as output the RTT (Round-Trip Time) estimate and the network download speed for the visited Web pages. We then cluster the obtained estimations for a set of visited Web pages by the user using clustering techniques such as Gaussian Mixture models in order to profile the underlying network. We end up proposing an algorithm to track the clusters in real-time to detect any network anomaly and those pages impacted by it.

TABLE I: Web performance metrics

Web QoS metrics	Web page features
Connect Start	Page Size (Size)
DNS	Maximum objects size (Max Objects)
Request	Number of objects (NumObjects)
Response	Protocol supported (Protocol)
DOM	Median of objects size (Median Objects)
First Paint (FP)	Total objects size (Objects’ Size)
First Contentful Paint (FCP)	First Quantile of object’s size (Q1 Objects)
Page Load Time (PLT)	Third Quantile of objects size (Q3 Objects)
	Maximum objects size (Max Objects)

### A. Data collection

To validate our method, we need labelled network experiments where the ground truth about the network conditions

is known. The purpose of this phase is thus to collect a large dataset that captures the link between Web browsing performance and network performance using controlled experiments. For this purpose, we developed a distributed system where different network conditions are emulated and measurements of both network and Web are collected.

Our system to carry out the study involves three main entities; the Experimenter, the Web client and the Database. The Experimenter generates and sets the network configurations, using a traffic shaping emulation tool, and then launches the Web client. The Web client is composed of the browser, which is responsible for loading a specific Web page, and running an extension that collects all the information that we need to build our model. Finally, the database is where the collected data is stored. We will tackle the framework development thoroughly in the implementation part.

### B. Data-driven network estimation

This phase consists in estimating the underlying network metrics using the data collected within the browser. It is achieved using a CNN model, which we show in [7] to yield a higher estimation accuracy compared to other traditional machine learning techniques. We consider a regression variant of CNN that has as input the different Web metrics (see Table I), and Web page features, such as the number of objects, their size and the protocol supported. We obtain as output a tuple of estimations  $(\hat{r}tt, \hat{b}d)$  reflecting the network conditions under which the page was browsed.

### C. Data clustering

Here, we give a detailed overview of the GMM model used to perform the clustering of network conditions, based on the estimations provided by the deep learner. Later on, we justify its efficiency by comparing its performance to other clustering methods. In general, a mixture model like the GMM is a statistical model used to parametrically estimate the distribution of random variables by modelling them as a sum of several other simple distributions. In particular, a Gaussian mixture model is a linear combination of a finite number of Gaussian components with unknown parameters. Assume the existence of a multi-state random variable  $X = \{x_i | i \in 1, \dots, n\}$ , the probability density  $g(x)$  of the Gaussian's mixture modelling  $X$  can be expressed as the weighted sum of  $M$  other components whose densities are  $g_k(x)$ ,  $k \in 1, \dots, M$ :

$$g(x, \theta) = \sum_{k=1}^M \pi_k g_k(x, \theta_k), \quad (1)$$

where  $\pi_k$  represents the prior probability of a data point belonging to component  $k$ . The  $\pi_k$  satisfies the probability conditions  $\sum_k \pi_k = 1$  and  $0 \leq \pi_k \leq 1$ .  $\theta$  and  $\theta_k$  respectively denote the parameters of the model  $g$  and  $g_k$ .

We run GMM taking as input the unlabelled estimated values of the underlying network metrics (Delay and Bandwidth) and as output the predicted clusters together with the parameters of the associated Gaussian components. Several methods exist to estimate these parameters, the widely used

one being the Expectation-Maximisation method (EM), which proceeds in an iterative way following the Maximum Likelihood principle. One of the challenges is how to set the optimal number of clusters; that is, the number of Gaussian components that fits best the data. For that, we use an extension of the EM algorithm built using the Bayesian variational inference technique (BV). For instance, the new method called BGMM [15] for Bayesian Gaussian Mixture Model will eventually not only estimate the cluster parameters but will also give an approximation of the clusters distribution itself.

Clustering validation is an essential step to assess how good the clustering model is. We consider for this validation three widely used scores: Purity (P), Rand index (R) and Fowlkes Mallows index (FM) [16].

**Purity.** This index measures how pure each cluster is, which means to what extent the elements of a given cluster are included in the ground truth partition. It is given by

$$P = \frac{1}{n} \sum_i \max_j |C_i \cap T_j|, \quad (2)$$

where  $n$  is the total number of data points,  $i$  is the cluster index and  $|C_i \cap T_j|$  is the number of points that are common between the found cluster  $C_i$  and the ground truth partition  $T_j$ . This purity index is in the range  $[0, 1]$ ; the closer it is to 1, the better the compatibility with the ground truth.

**Rand index.** Given clustering  $C$  and ground-truth partitioning  $T$ , the pairwise measures utilize the partition and cluster label information over all pairs of points. Let  $(x_i, x_j)$  be any two different points in  $T$ . If both  $x_i$  and  $x_j$  belong to the same cluster, we call it a positive event, and if they don't belong to the same cluster, we call that a negative event. There are four possibilities to consider, depending on whether there is an agreement between the cluster labels and ground-truth labels. Based on this concept, the Rand index is used to express how similar a cluster is to a ground-truth partition and is given by  $R = (TP + TN)/N$ , where  $TP$  denotes the number of true positive events,  $TN$  the number of true negatives, and  $N = \binom{n}{2}$  the number of pairs in the data set. The Rand index has a value between 0 and 1; a higher value indicates a better similarity, which signifies a better clustering performance.

**Fowlkes-Mallows index.** This index measures how well the clustering model performs using two pairwise indicators, pairwise precision and pairwise recall values. Recall measures the number of data points classified correctly over all points in the same ground-truth partition, while precision measures the number of data points classified correctly over all points in the same cluster. The Fowlkes-Mallows index is defined as the geometric mean of precision and recall:  $FM = \sqrt{recall \cdot precision}$ .

The FM index is also between 0 and 1, with 1 being a scenario with perfect clustering,  $FP = 0$  &  $FN = 0$ .

### D. Real-time anomaly detection

Our goal here is to propose a heuristic algorithm with a periodic process of resettlement that allows tracking clusters over time in order to detect network performance anomalies.

Let  $P$  be the current group of visited Web pages with network performance estimations  $(\hat{r}t, \hat{b}d)$  (one pair of values per page visit) and let  $Q$  be the group of Web pages that are to be visited successively with rate  $\lambda$ . We consider a window of pages  $W$  of temporal span  $T$  after which we reinitialize the entire process. In this work,  $T$  is taken equal to 48 hours. Let  $i \in Q$  be a Web page that is visited at time  $t_i$  within the window  $T$ , and let  $B$  be the baseline distribution of network performance over  $P$ . We proceed as follows for each page  $i$ . First, we define  $tag(i)$  that checks if the data point  $i$  is marginal or not to the baseline distribution depending on the criterion  $pvalue(i)$ . The marginality of a Web page indicates that it scrolls from the normal, so we tag it red, otherwise we consider it of normal performance and tag it blue. Second, we apply BGMM based clustering to see if a new network state emerges. A perfect scheme would be to perform clustering for each point  $i$ . However, one must consider reducing the cost of calculation of our clustering model. So we introduce a step  $K$  ( $K = 10$ ) of re-clustering (only marginal pages are considered in the count). Third, by checking the clustering results, we have two possibilities; whether there is an emergence of a new cluster or not. A new cluster composed of red points is a clear sign of a change in network conditions. If the change is to the worst, we return a detected anomaly.

Regarding the baseline distribution, we update it when there is an emergence of a new network state. If no new state emerges within time  $T$ , we consider the baseline distribution to be stale, so we updated and reset the set of Web pages  $P$ .

#### IV. PERFORMANCE EVALUATION

##### A. Framework setup

We build an experimentation framework that integrates our approach based on the joint use of passive measurements and deep learning (CNN). Our framework is built around an automated process that consists of sending and receiving messages between the Experimenter and the Web client. The Experimenter is divided into a simple Finite State Machine (FSM), developed in python, and a Web page to load the experiments. For the Web client, we use Google Chrome as browser, and interact with it via a JavaScript extension we developed using Chrome Navigation Timing API and Performance Navigation API, which are w3c recommendations.

We define the network conditions we want to emulate as N-tuples defined as follow:  $\{TC_1^i, \dots, TC_N^i\}$ , where  $N \in 2, \dots, 5$  is the number of different network conditions faced by the user at a random time,  $i$  is the  $i^{th}$  experimentation and  $TC = (RTT, BD, p)$  is a tuple of network delay  $RTT$ , network bandwidth  $BD$  and  $p$  being a group of Web pages visited by the user and assigned to a particular network condition ( $RTT$  and  $BD$ ). The values of  $RTT$  and bandwidth are picked randomly from a list of samples generated by the FAST technique (Fourier Amplitude Sensitivity Test [17]); FAST is a sampling method that covers a given space based on relevant frequencies, which allows an efficient scan of the area to be sampled. These values are then enforced by the network emulator using Linux Traffic Control tool *tcconfig*. The group

of Web pages  $p$  is chosen by picking randomly  $|p|$  Web pages from the total group of Web pages  $P$  such that:  $|p| = \lceil \frac{|P|}{N} \rceil$ . In this work, we consider as  $P$  the 500 top popular Web pages according to Alexa ranking.

Having under our control the end-to-end network path can be a real challenge; in fact, we have to load real Web pages from the cloud. Unfortunately, this latter is out of our experimental network. In such circumstances, *tcconfig* may face difficulties in enforcing the desired configurations. Thus, the need to validate and check samples of the N-tuples ( $RTT$  and bandwidth) before starting the actual experimentation through  $RTT$  noise estimation and throughput tests.

Our platform performs the measurement of Web performance metrics listed in Table I, then gives estimations of delay and bandwidth for each visited page using the CNN-based deep learner we proposed in [7], [8]. By applying our approach (all scenarios), we obtain a dataset composed of 8000 different network scenarios for 500 Web pages.

##### B. Results

We run BGMM over the tuples of estimated bandwidth and  $RTT$ . We tune BGMM to find automatically the optimal number of clusters, in the limit of 5 clusters. For the covariance type parameter of BGMM, we set it to 'full' since we want to find completely separated clusters with singular covariance matrix each. For the weight concentration prior parameter, we specify the Dirichlet process, and we set 'k-means' as the initialisation method in initial parameters.

**General case.** The boxplot in Fig. 2 displays the dispersion of the global clustering accuracy over all network configurations (x-axis). The histogram in Fig. 3 gives the accuracy of finding the same number of clusters as the ground-truth.

We notice a minor gap between the different scenarios, with the accuracy decreasing with the number of parallel network conditions. In general, the 5 TCs scenario shows the least accuracy for both the clustering and the identification of the right number of clusters. Still, the value is as high as 85% for the first case and 90% for the second case. For the 2 TCs scenario, the accuracy of clustering is even higher (90% on average), and the precision of finding the exact number of clusters is perfect (near 100%). As for the 3 TCs and 4 TCs scenarios, they display an accuracy higher than 85% for the first case and 95% for the second case.

Now, we check whether the accuracy of clustering varies if we change the number of Web pages until we reach the maximum 500. We consider all the scenarios in Fig. 4 and show a CDF plot displaying the accuracy as a function of the number of visited Web pages. We can notice how the performance of our approach increases significantly with the number of Web pages. For example, to reach an accuracy of 75%, we need 50 Web pages for 2 TCs, 175 Web pages for 3 TCs, 225 for 4 TCs, and 500 for 5 TCs. These results give hints on how to choose the minimum number of Web pages to consider for clustering to achieve the best possible accuracy.

**Case of two TCs.** The identification of the different network conditions depends on the distance that exists between these

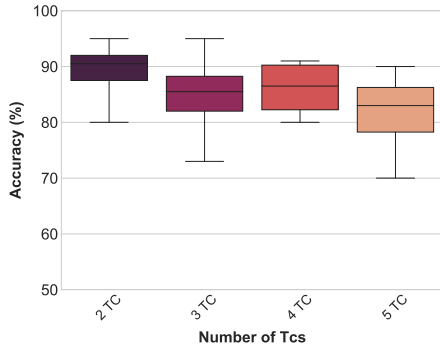


Fig. 2: BGMM clustering accuracy for 500 Web pages

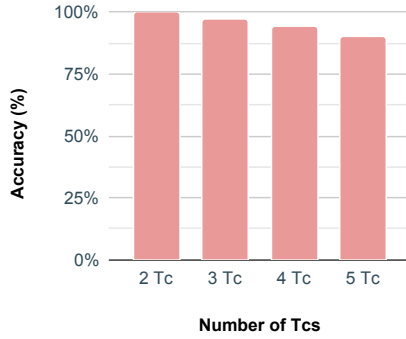


Fig. 3: Accuracy of finding the right number of clusters

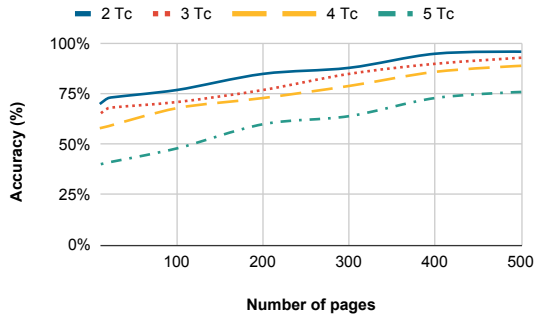


Fig. 4: Clustering accuracy versus number of pages

conditions. We study this relationship for the case of two different conditions ( $N = 2$ ). We plot in Fig. 5 the heatmap of the minimum number of pages needed to achieve 85% of accuracy, for different distances in terms of delay and bandwidth. Clearly, the more distant the conditions are, the less the number of pages required, with a few pages enough to separate conditions different by 7Mbps and 300ms, and 500 pages allowing to differentiate scenarios with less than 1Mbps and 50ms differences. We further provide results where the number of pages is unbalanced between the two conditions. For some specific scenarios, Fig. 6 shows that the balance helps in improving clustering accuracy, and any unbalance can be compensated by increasing the number of pages to cluster.

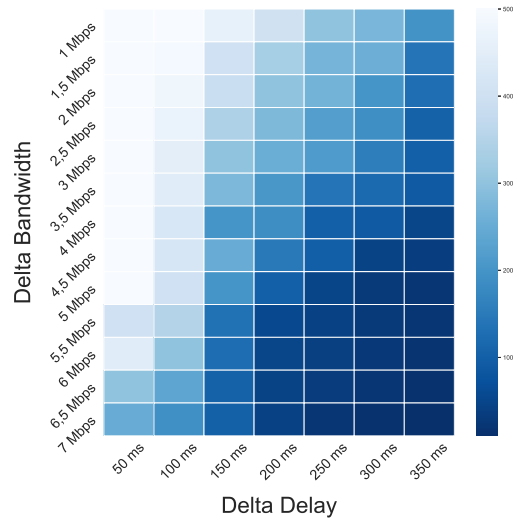


Fig. 5: Heat map of the minimum number of pages needed to achieve 85% clustering accuracy

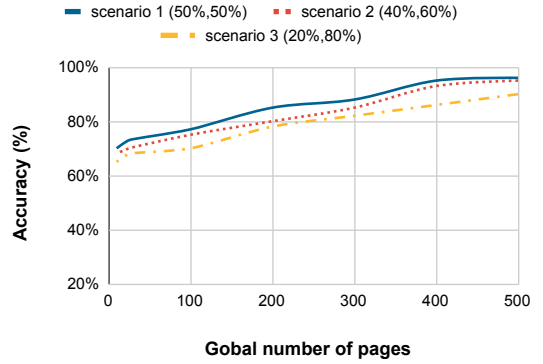


Fig. 6: Clustering accuracy versus number of pages for different balances between the network conditions

### C. Comparison with other clustering methods

TABLE II: Comparison between clustering models

	Purity Index	Rand Index	FM score
K-means	0.502	0.662	0.67
GMM	0.84	0.772	0.78
BGMM	0.89	0.886	0.812

Here, we compare the performance of our BGMM based approach with other two well known clustering methods: K-means and classical GMM. For K-means, we use a model with  $N$  components fit. For GMM, we use a model with  $N$  components fit and Expectation-Maximization for parameter estimation. We consider the Elbow Method to determine the optimal number of clusters in a range between 2 and 5. As shown in Table II, K-means shows the least performance, especially for the purity index, which illustrates the difficulty of the task. GMM comes after BGMM; in particular, the purity index can go up to 0.89 overall with BGMM.



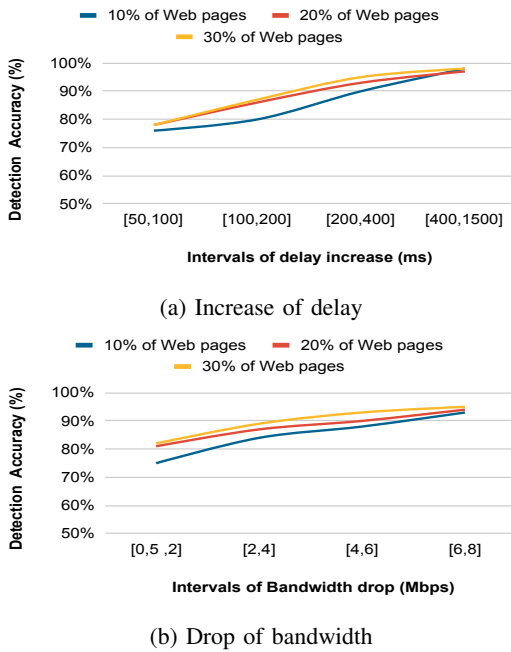


Fig. 7: Accuracy detection of delay increase and bandwidth drop for different percent of Web pages

#### D. Anomaly detection validation

Here we validate our algorithm for anomaly detection. We consider two different scenarios with specific anomalies. The first one is a drop in the bandwidth value for a percentage of Web pages (10%, 20% or 30%). We vary the drop from 0.5 Mbps to 8 Mbps. The second one consists of an increase in the delay respectively for 10%, 20% and 30% of Web pages.

We validate our algorithm over 100 visited Web pages (maximum anomaly duration). Indeed, we randomly pick 200 Web pages with homogeneous network conditions from the dataset. We consider 100 pages as the group  $P$  representing the baseline distribution. We consider the rest as the group  $Q$  representing the Web pages that arrive after the anomaly occurs. We include the anomaly in group  $Q$  for a percentage of the pages, then launch the algorithm.

We want to check the ability of our algorithm to detect these anomalies. Fig. 7 gives an overview of the results. It shows the detection accuracy of anomalies depending on whether we are dealing with a drop in bandwidth or an increase in  $RTT$ . We observe that the accuracy of detecting anomalies increases significantly when we have larger shifts in network performance. Anomaly detection accuracy also increases when the number of pages impacted by the anomaly increases. For example, if a drop in network bandwidth occurs between 6 Mbps and 8 Mbps, the detection accuracy can be as high as 93%, and an increase in delay between 400 ms and 1500 ms gives an accuracy that can be as high as 98%. In all our scenarios, the accuracy was found to be above 75%. Note that one can further increase the accuracy if the anomaly lasts longer than the 100 page limit we consider here.

## V. CONCLUSIONS

We presented and implemented a lightweight Web-based network monitoring solution able to infer the underlying network performance and detect network anomalies. Our solution consists of (i) an original network monitoring framework, based on Bayesian Gaussian Mixture Models (BGMM), able to provide information on the underlying network state for the different visited Web pages by the user, and (ii) an algorithm to detect in real-time the occurrence of performance anomalies. We validated our approach with controlled experiments where the network conditions were varied while the Web pages were browsed. Validation results showed that our approach can yield accurate detection results even in scenarios with small shifts in network performance and a few percentage of Web pages impacted. We will keep developing this solution towards a deployment in the wild and the identification of the root causes of the degraded network performance based on the identified subset of pages impacted by the anomalies.

## REFERENCES

- [1] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Characterizing web page complexity and its impact," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 943–956, June 2014.
- [2] R. Schatz, T. Hößfeld, L. Janowski, and S. Egger, *Data Traffic Monitoring and Analysis*. Springer-Verlag, 2013.
- [3] E. Bocchi, L. De Cicco, and D. Rossi, "Measuring the quality of experience of web users," in *Proc. of the 2016 Workshop on QoE-based Analysis and Management of Data Communication Networks*, ser. Internet-QoE '16. New York, NY, USA: ACM, 2016, pp. 37–42.
- [4] D. Da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the gap between QoS metrics and Web QoE using Above-the-fold metrics," in *PAM 2018 - Int. Conf on Passive and Active Network Measurement*, Berlin, Germany, Mar. 2018, pp. 1–13.
- [5] M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson, "Fathom: A Browser-based Network Measurement Platform," in *ACM Internet Measurement Conference*. Boston, United States: ACM, Nov. 2012, pp. 73–86.
- [6] Speedtest, <https://www.speedtest.net/>.
- [7] I. Taibi, Y. Hadjadj-Aoul, and C. Barakat, "When Deep Learning meets Web Measurements to infer Network Performance," in *IEEE CCNC 2020*. Las Vegas, United States: IEEE, Jan. 2020, pp. 1–6.
- [8] I. Taibi, Y. Hadjadj-Aoul, and C. Barakat, "Data driven network performance inference from within the browser," *IEEE ISCC*, pp. 1–6, 2020.
- [9] S. Fu, J. Liu, and H. S. Pannu, "A hybrid anomaly detection framework in cloud computing using one-class and two-class support vector machines," in *ADMA*, 2012.
- [10] T. Wang, J. Wei, F. Qin, W. Zhang, H. Zhong, and T. Huang, "Detecting performance anomaly with correlation analysis for internetware," *Science China Information Sciences*, vol. 56, pp. 1–15, 08 2013.
- [11] X. Gu and Y. Tan, "Online performance anomaly prediction and prevention for complex distributed systems," 2012.
- [12] T. Wang, J. Wei, W. Zhang, H. Zhong, and T. Huang, "Workload-aware anomaly detection for web applications," *Journal of Systems and Software*, vol. 89, pp. 19–32, 2014.
- [13] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.
- [14] H. Cui, E. Biersack, and E. S. Antipolis, "Distributed troubleshooting of web sessions using clustering."
- [15] J. Lu, "A survey on bayesian inference for gaussian mixture model," 2021.
- [16] M. Rezaei, "Clustering validation," Ph.D. dissertation, Itä-Suomen yliopisto, 2016.
- [17] S. Tarantola and T. A. Mara, "Variance-based sensitivity indices of computer models with dependent inputs: The Fourier Amplitude Sensitivity Test," *Int. J. for Uncertainty Quantification*, vol. 7, no. 6, Apr. 2017.