A Machine Learning approach for dynamic selection of available bandwidth measurement tools

Alessio Botta, Gennaro Esposito Mocerino, Stefano Cilio, Giorgio Ventre
University of Napoli Federico II, Napoli, Italy
a.botta@unina.it, gennaro.espositomocerino@unina.it, stefano.cilio@unina.it, giorgio.ventre@unina.it

Abstract—**Available bandwidth is a vital parameter for understanding network status. A huge number of tools have been proposed in literature, including general ones as well as tools specialized for specific network scenarios. In this plethora of possibilities, an expert user is currently required to select the best one according to the specific operating settings, in order to achieve accurate results without disturbing the existing traffic.**

In this paper we propose the use of automatic decision systems based on machine learning to substitute the expert user and choose the right tool for the current scenario. To verify if this is a viable solution, we created a custom dataset and tested different decision systems including a simple, threshold-based one and four algorithms based on machine learning: k-nearest Neighbors, Random Forest, Support Vector Machine, and Long Short-Term Memory networks. We used different features including CPU, memory, and bandwidth and verified that the decision systems based on machine learning achieve very good performance and can be considered as a promising solution for this important problem.

Index Terms—**Available Bandwidth, machine learning, network measurements.**

I. Introduction

Network performance is a hot research topic since the early days of the Internet. Several different tools have been proposed but the problem of accurately measuring network performance is still an open research problem. Measurement tools are typically divided in active and passive ones. Tools from the former category do the measurement injecting probing packets into the network under test and collecting them at receiver side to evaluate how some characteristics of the probing packets (e.g. the inter-packet time) has been altered during the journey. Analyzing such alteration they derive the measure of interest. The main distinguishing characteristic of passive measurement tools is the use of existing traffic, therefore injecting no additional traffic in the network. Not having such traffic under control, however, limits the possibilities of the measurements that can be done and introduces additional sources of inaccuracy.

One important parameter regarding network performance is available bandwidth (AB). It measures the unused capacity of a network path thus providing a vital information regarding current network status and what an application can expect to obtain in this particular moment. Due to its importance, several tools and techniques have been proposed in literature to measure AB, having varying accuracy typically related with a different level of intrusiveness (i.e. amount of traffic injected in the network to perform the measurement). The difficulty to estimate such important parameter has led to the spread of tools customized for specific network conditions, with no single tool able to reach the best trade-off between accuracy and intrusiveness in all the possible situations and a skilled user is necessary to decide which tool to use according to each specific case [1], [13], [15]. Moreover, combination of different measurements are needed to predict the available bandwidth [18].

Starting from this assumption, in this work we advocate the use of intelligent systems substituting skilled users to select the AB measurement tool most suited to the specific scenario under study. We envision a smart decision system, based on machine learning, properly trained and using a set of features on which to make an educated decision. To pave the way for this vision to come true, we used different decision mechanisms (from threshold-based decision systems to deep learning) and compared their performance. We used features considering the current occupancy of the network as well as the current load of the CPU and memory. Our results show that systems that choose which measurement tool to use by relying only on thresholds fail in picking the most appropriate one more often than those that base their decisions on a machine learning model, the latter being able to take into account network dynamics. This means that a measurement system that can adapt to the network conditions by reducing its intrusiveness when needed can be obtained relying on machine learning algorithms.

In this paper we provide the following contributions:

- we design and test an automated system based on machine learning to dynamically select the best tool to measure the available bandwidth;
- we create an artificial dataset to test our system;
- we test different machine learning tools for this task and compare their results with a simple threshold-based decision system.

The rest of the paper is structured as follows. In section II we cite some of the most popular tools for available bandwidth measurement. Next, in section III, we describe which kind of systems were tested, on which data set they were trained on and which measurement tools they could choose. In section IV we show how the systems were evaluated and what are the results obtained. Finally in section V we sum up the work and draw conclusions.

II. Related work

Many software tools for available bandwidth measurement have been proposed in the last years. Examples of such tools are PathChirp [3], Pathload [4], Abing [5], Spruce [6], DietTopp [7], and many others. The performance of these tools is usually measured in terms of accuracy, intrusiveness, and time needed to convergence [10], [14]. Along with this variety of tools, many comparative studies to assess performances in different conditions have been carried out [13]. Shriram et al. [9] have compared abing, pathchirp, pathload, Spruce and also included Iperf [8], which measures achievable TCP throughput and has become an unofficial standard in the research networking community. They found that pathload and pathchirp are the most accurate tools. Castellanos et al. [11] found that, even though the performance of none of the investigated methods are outstanding, pathChirp excels as the best tool in terms of both accuracy and efficiency. Ahmed Ait Ali et al. [12] obtained results showing that Pathload is the most intrusive tool and, in some cases, can be very slow. Pathchirp overestimates the available bandwidth and IGI is inaccurate. Finally, Spruce seems to be the tool that offers the best performance with regards to the criteria considered. The aforementioned tools show to perform better or worse according to the different setup and condition of the network and on manual fine tuning of involved parameters. G. Aceto, et al. introduce a unified architecture for network measurement (UANM) capable of automatically managing different measurement stages such as the choice of the measurement tool and its fine tuning to cope with different network setups and with different measurement purposes and constraints [1].

More recently, in innovative scenarios such as software defined networks (SDN), the problem of available bandwidth estimation has once again gained attention. The reason is that the traditional methods are inadequate and poor performing when facing the new challenges posed by the different network configuration and behaviour, thus the urge to introduce novel approaches to the problem [2].

In this paper we propose a system based on machine learning to dynamically select the most fitting measurement tool according to varying network condition. In literature, machine learning techniques have already been used to perform available bandwidth estimation as an alternative to the aforementioned traditional tools. N. Sato, et al. [16] proposed PathML, an AB estimation method based on a data-driven paradigm that uses machine learning with a large amount of data. They performed a set of experiments over LTE networks and compared their approach with PathQuick3. Ling-Jyh Chen, et al. [17] proposed a different machine learning-based approach, they used a Support Vector Machine (SVM) and two probing models in their evaluation: the packet train model and the pathChirp-like model.

In conclusion, several approaches for measuring the AB have been presented in literature, mainly dealing with the selection and configuration of measurement tool in a deterministic fashion or with the use of machine learning to do the measurements. In this work instead we advocate the use of machine learning for selecting one of the several already available AB measurement tools according to the specific operating conditions. For details, see section III

III. Tools and data

Our aim is to build an automatic intelligent decision making system able to choose the most appropriate measurement tool for available bandwidth among three representative candidates used in our tests: iperf, pathload, and pathchirp. The tools are listed from the most to the least accurate and, at the same time, intrusive. A tool is more intrusive as more probing traffic it generates and injecting more probing packets requires also more CPU (and memory in some cases).

To better understand what is meant by most appropriate, we should recall that the measurement tool is running on a system together with other applications whose execution should not be by any means affected by the measurement process. These applications consume, and hence need, different amounts of cpu, memory and bandwidth. The most appropriate measurement tool is the one that can guarantee the best accuracy by using only the available resources as not to interfere with the applications in execution. So, when the system has low spare resources the most appropriate tool is the least intrusive, while when the system is less busy it is more appropriate to choose a more accurate, yet more intrusive, tool.

It is important to notice that the measurement tool has to be chosen in advance and will be used for the following 30 seconds, so the automatic tool needs to predict which tasks will be in execution and how much bandwidth they will require, and choose the measurement tool accordingly. As input features, in addition to the bandwidth used by the task in execution in the previous samples, also cpu and memory usage are passed. Variation in these two parameters indeed can hint that a certain task has stopped or started. In Tab. I some of the tasks that can be executed by the system in our scenario are listed. Each of these tasks needs resources in different amounts, hence it is possible to infer which one is running from the available resources. If, for example, the system is running a task "Slam on board", the cpu, memory and bandwidth consumption is in the ranges specified in the table. When one or more of these values change and takes a value out of the ranges, it means that the system has moved to another task, or is idle. These tasks have been thought in the real of Cloud and Dew robotics [19] [20].

A. Dataset

The dataset we used to test our systems, represents a real scenario in which there are some tasks that use different resources like CPU, memory and bandwidth. Our systems must measure AB considering CPU, memory, and the bandwidth that currently running tasks use, choosing the more appropriate monitoring tool which will then perform the measurement for a certain period of time.

Every row of the synthetic dataset show the cpu, memory and bandwidth usage in average over a period of 10 seconds.

Task	Cpu(%)	Mem(%)	bw(kbps)	Duration(s)
Slam on Board	(65-85)	(40-60)	(250-750)	(30-60)
Remote Slam	(15-25)	(15-25)	(8000-12000)	(10-60)
Video Capture & Encoding	(70-90)	(15-25)	(800-1200)	(20-40)
Video Capture & Transmission	(15-25)	(15-25)	(15000-30000)	(10-40)
idle	(1-10)	(1-5)	(5-25)	(10-60)

TABLE I: Table of Tasks

The dataset has been built by randomly picking the task in execution from those in Tab. I. We can see that each task has a range of values for CPU, memory, bandwidth and duration. Random values from these ranges are extracted to form a new row and n copies of this row will be appended to the dataset where $n = T_r/10$ and T_r is the randomly picked time for the task. We created two datasets using this approach, a large one, having 418,363 rows, and a small one, having 323 rows. We also used the small dataset because it may be more representative of real situations. Considering that each sample represents 10 seconds, the large dataset represents about 48 days, while the small one represents 53 minutes.

The dataset has been labeled considering how much bandwidth is available and how much is needed by the three measurement tools to perform the measurement. Once a choice has been made, the AB measurement tool will run for the following 30 seconds, this means that it should consider the bandwidth used by the tasks running on the system over that time interval as in the following:

$$bw_m(T) = max(bw(T), bw(T + 10), bw(T + 20)) \quad (1)$$

where $bw(T)$ is the bandwidth required on average by the tasks over the time interval $[T, T + 10)$. Since the chosen measurement tool runs for 30 seconds, a choice is needed only when time is 0 or a multiple of 30.

B. Decision systems considered

We used different types of algorithms to make the decision about the AB measurement tool: simple system, classification systems, and prediction systems. The simple system just takes as input $bw(T)$ and outputs the tool assuming that this value will remain stable during the measurement period and that the CPU and memory consumption of the tool will never interfere with existing tasks. Regarding the classification systems, we used three different types of supervised learning classification algorithm: k-nearest Neighbors (KNN), Random Forest (RF), and Support Vector Machine (SVM). These were trained by giving them in input the features (cpu, memory, bandwidth) and the labels (tool1, tool2, and tool3). For the prediction systems, we used a Deep Long Short-Term Memory (LSTM) network. Two different versions of LSTM have been used, the first one has been trained on sequences of the bandwidth, CPU, and memory over time while the other has been trained on bandwidth sequences only. In both cases, they predict the bandwidth in $T + 10$ and $T + 20$ based on past values of

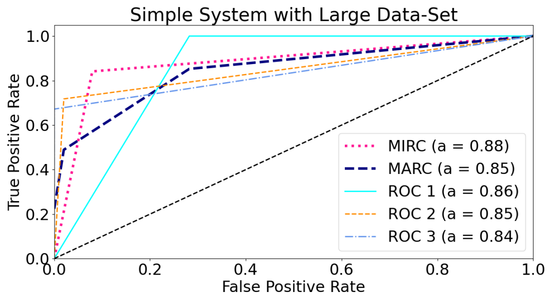

Fig. 1: Simple System: ROC Curves large data-set

the features. Then, we used such value to decide the AB measurement tool to use.

IV. Results obtained

The performances of the proposed systems have been compared by examining Roc curves and AUCs. A ROC curve (receiver operating characteristic curve) is drawn by plotting True Positive Rate (TPR) against False Positive Rate (FPR) at different classification thresholds. AUC, namely the area under the ROC curve, measures the ability of the classifier to distinguish between classes and ranges in values from 0 to 1, the higher is the AUC the better the classifier works. In multi-label classification, as our case, a ROC curve can be drawn individually for each class or, alternatively, MIRC and MARC curves can be drawn to get averaged performance of the classifier among the classes. Macro-average Roc curve (MARC) plots TPR on FPR by computing the metrics independently for each class and then taking the average, on the other hand Micro-average Roc curve aggregates the contributions of all classes to compute the average metrics. The two curves may differ due to imbalanced dataset.

These measures are depicted in the following graphs with labels: “MIRC” is the micro-average ROC curve, “MARC” is the macro-average ROC curve, “ROC 1” is the ROC curve of class 1 (tool 1), “ROC 2” is the ROC curve of class 2 (tool 2), “ROC 3” is the ROC curve of class 3 (tool 3) and “a” is the Area Under the ROC Curve (AUC). The tests have been carried out with the proposed datasets using the first 70% of dataset for the training, and the last 30% of the dataset is for the testing. Then, we made a 10 fold cross-validation.

A. Simple System

To test the performance of the Simple system, we compared the predictions it generated with the real values (true tools). We tested it with the two proposed dataset and plotted the ROC curves. The choice is made by seeing only the bandwidth feature excluding the CPU and memory features. Figure 1 shows the ROC curves with large dataset.

From the figure we observe that in general the system achieves good performance (it detects around 86% of tool 1, 85% of tool 2 and 84% of tool 3). As expected, the same results have been obtained with the small dataset, making the simple System a simple comparison without keeping track of

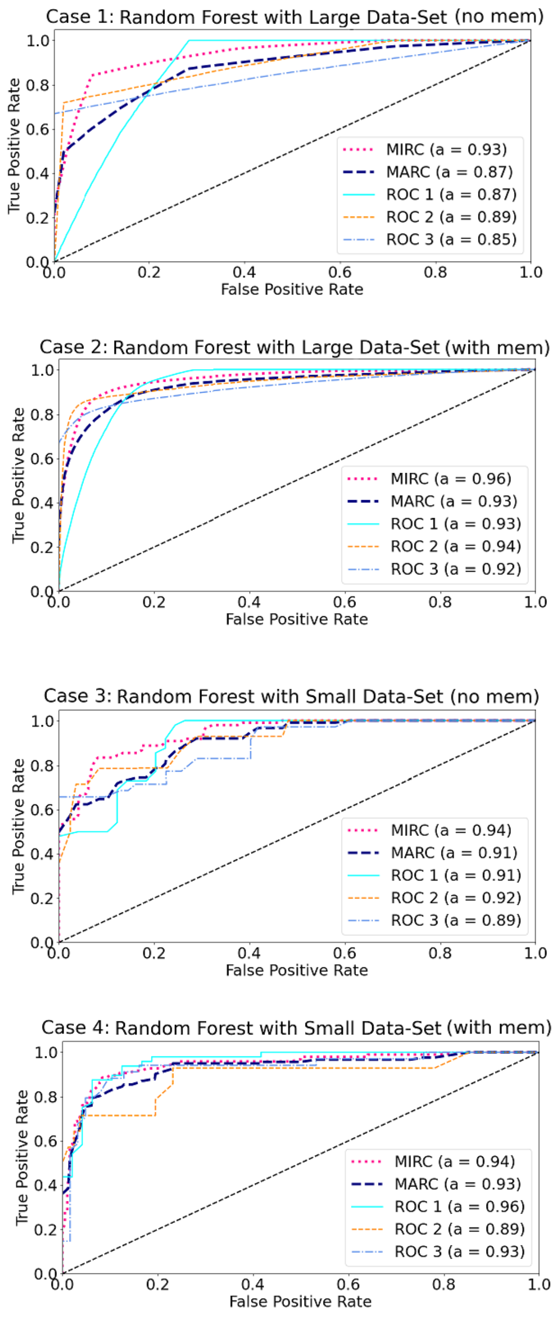

Fig. 2: RF ROC Curves

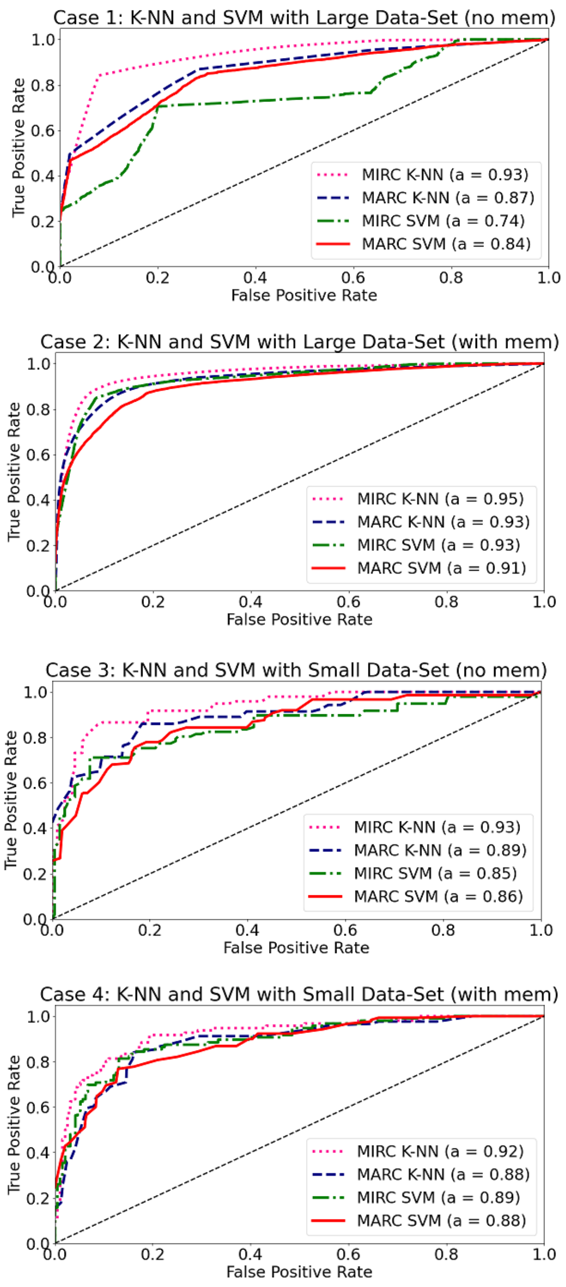

Fig. 3: MIRC and MARC Curves of the KNN and SVM

the past nor learning from the dataset. Thus, its performance is independent from the size of the dataset and cannot be increased adding more samples. Notice that Simple system is representative of tools like "UANM" [1] which only looks at the current value, do not make predictions and have a static association between features and tools.

B. Classification Systems

We have tested the classification systems with two different combinations of the two datasets described in Sec. III-A, thus obtaining four cases: case 1 and 2 refer to the large dataset, and case 3 and 4 to the small one. The difference between case 1 and 2 (as well as between case 3 and case 4) is that in the latter case we have tried to add memory: for each row, we added the four previous values to such row. In this way, the classification system considers also what happened a few time intervals before the current one.

Therefore, datasets without memory take as input samples of CPU, memory and bandwidth at time T, while datasets with memory take as input also samples up to 40 seconds in the past. For the best classification system (Random Forest), we plotted the ROC curves, the micro (MIRC) and macro

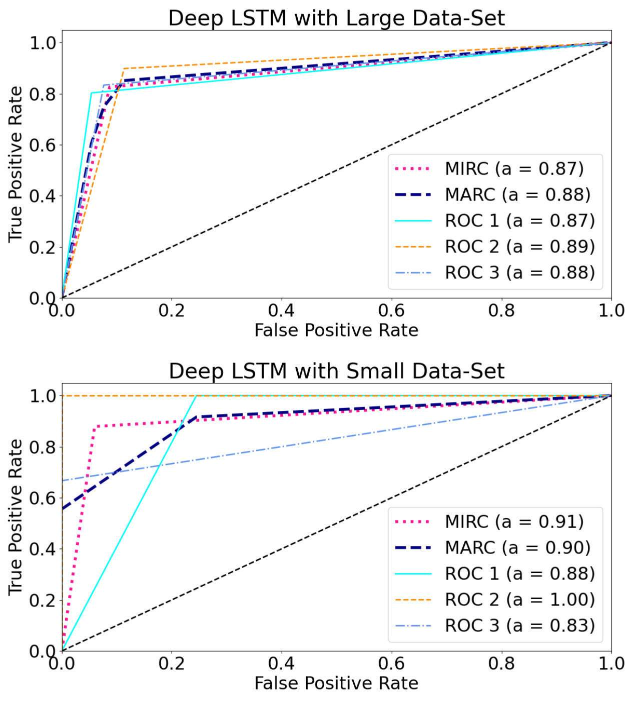

Fig. 4: Deep LSTM: ROC Curves

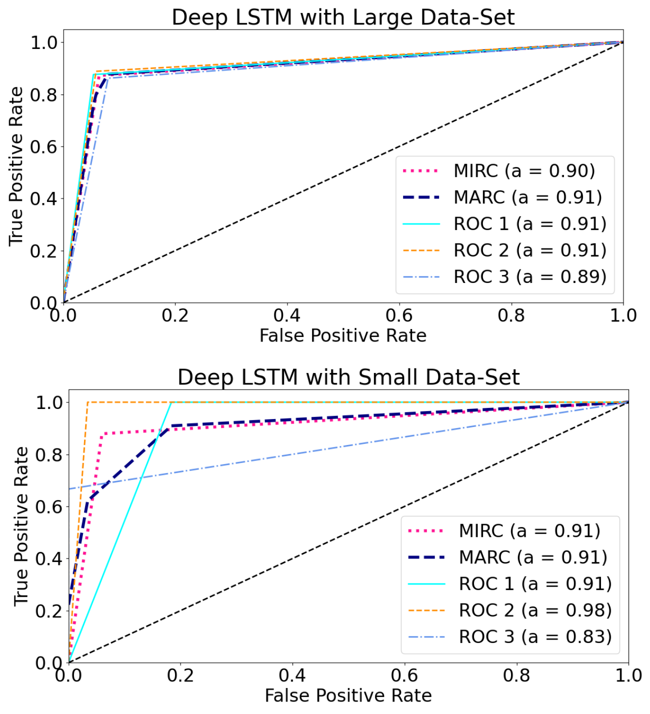

Fig. 5: Deep LSTM only Bandwidth: ROC Curves

(MARC) average while for the others (K-nearest Neighbors and Support Vector Machine), we plotted only the MIRC and MARC. Fig. 2 shows the RF ROC curves of case 1, 2, 3, and 4, and fig. 3 shows the micro and macro average of the KNN and SVM related to case 1, 2, 3, and 4.

With large dataset we have an improvement of the results by adding memory (case 2). In case 1, Random Forest achieves the best performance detecting around 87% of tool 1, 89% of tool 2 and 85% of tool 3 while in case 2 it achieve the best performance detecting around 93% of tool 1, 94% of tool 2 and 92% of tool 3.

With the small dataset without memory (case 3) it achieves the best performance detecting around 91% of tool 1, 92% of tool 2 and 89% of tool 3. We do not have a significant improvement of the results by adding memory (case 4), but still Random Forest achieves the best performance detecting around 96% of tool 1, 89% of tool 2 and 93% of tool 3.

C. Prediction Systems

We tested the two LSTM Systems proposed only with two of the datasets described in Sec. III-A, not considering the cases in which we have datasets with memory as for the classification systems. This is because the LSTM model already has memory using the concept of the look back, which is the number of previous time steps to use as input variables to predict the next time period. Fig. 4 shows the ROC curves of the Deep LSTM with CPU, memory, and bandwidth.

We can see that the small dataset presents slightly higher values for ROC curves areas than the large dataset one, if we consider the aggregate of the three classes. The ROC curves areas of each single class show that the small dataset model outperforms the large dataset one only in recognizing class 2, while the large dataset model seems to be better balanced. The LSTM system with CPU, memory, and bandwidth (with large dataset) achieves very good performance detecting around 87% of tool 1, 89% of tool 2 and 88% of tool 3 while with the small dataset, the LSTM system achieves good performance detecting around 88% of tool 1, 100% of tool 2 and 83% of tool 3.

Figure 5 shows the ROC curves of the Deep LSTM only with Bandwidth. Comparing the large and small dataset, we can see that the same considerations can be repeated as in the previous case. It is interesting to compare fig. 4 and fig. 5, indeed we can see that removing CPU and memory from the input of the LSTM model, performance improves both for the small dataset case and for the large dataset case. This can be explained considering that CPU and memory do not carry useful information that is not yet contained in the bandwidth for improving the predictive power of the models. In particular the LSTM system with only bandwidth achieves excellent performance detecting around 91% of tool 1, 91% of tool 2 and 89% of tool 3 with large dataset. Regarding the small dataset, it also achieves excellent performance detecting around 91% of tool 1, 98% of tool 2 and 83% of tool 3.

Tab. II compares the best prediction system with only bandwidth and the best classification system (RF) both with small dataset without memory. Both systems detects around 91% of tool 1. Tool 2 is detected more accurately by the prediction system (98%) compared to RF classification system (92%), while the latter detects more precisely tool 3 (89%)

	Tool 1	Tool 2	Tool 3
Prediction System	91%	98%	83%
RF classification system	91%	92%	89%

TABLE II: Comparison between Prediction and Classification Systems

compared to the prediction system (83%).

V. Conclusion

Our aim was to develop a smart decision system to choose the best AB measurement tool in each specific network condition. To meet this goal, we created a threshold-based system that chooses the measurement tool only based on the current bandwidth. We saw that the threshold-based system was not a good choice because it made a simple comparison, and did not keep track of the past, so it did not make accurate predictions. Predictions are instead necessary because the measurement lasts for a certain period of time (30 seconds), thus the system must choose the more appropriate monitoring tool according to CPU, memory and bandwidth that tasks will use in the next 30 seconds.

Because of this, we considered different types of smart systems: the classification and the long short-term memory (LSTM) ones. Regarding the classification systems, we tested three models: k-nearest neighbors, random forest, and support vector machine. For the LSTM system we tested two different versions: the first one was trained on sequences of the bandwidth, cpu, and memory over time while the second one was trained on bandwidth sequence only. To test these models, we created a synthetic datasets composed of several tasks which used the cpu, memory, and bandwidth differently, and each had a different runtime. In addition, we generated two types of those datasets: a large synthetic dataset with about 418,363 rows and a small synthetic dataset with 323 rows. For both datasets, we made two cases where in the second one we added memory to the synthetic dataset. Then, we tested our classification systems in four cases: large synthetic dataset (case 1), large synthetic dataset with memory (case 2), small synthetic dataset (case 3) and small synthetic dataset with memory (case 4). Our results show that the performance of the classification systems was high, and there was a slight improvement adding memory to the dataset and in this case, Random Forest achieved the best performance. Both LSTM systems performance was high. Moreover, the LSTM system with only bandwidth performed slightly better then the LSTM system with bandwidth, cpu, and memory.

In summary, all the systems achieve good performance, and suggest that systems based on machine learning can actually be used in place of expert users to decide which available bandwidth estimation tool has to be used in a specific network condition. The choice of which decision system to use should be made on the basis of considerations other than performance, such as the duration of training phase, the amount of resources required, etc.. Our ongoing work is mainly concerned with collecting data from real scenario in order to test the decision system with real data.

References

[1] Giuseppe Aceto, Alessio Botta, Antonio Pescapè, Maurizio D'Arienzo, *Unified Architecture for Network Measurement: the case of available bandwidth* Journal of Network and Computer Applications, Elsevier, Volume 35, Issue 5, September 2012, Pages 1402-1414.

[2] Pèter Megyesi, Alessio Botta, Giuseppe Aceto, Antonio Pescapè, Sàndor Molnàr, *Challenges and solution for measuring available bandwidth in software defined networks*, Computer Communications, Elsevier, Volume 99, 1 February 2017, Pages 48-61

[3] Vinay J. Ribeiro, Rudolf H. Riedi, Baraniuky Jiri Navratil, Les Cottrellz, *PathChirp: Efficient Available Bandwidth Estimation for Netw,rk Paths*, Passive and Active Monitoring Workshop (PAM 2003)

[4] Manish Jain and Constantinos Dovrolis, *Pathload: a measurement tool for end-to-end available bandwidth*, Proceedings of Passive and Active Measurement Workshop, 2002.

[5] J. Navratil, R.L. Cottrell, *ABwE: A Practical Approach to Available Bandwidth* Proceedings of the 4th International workshop on Passive and Active network Measurement, PAM 2003 (2003).

[6] J. Strauss, D. Katabi, F.Kaashoek, *A measurement study of available bandwidth estimation tools* Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, IMC 2003 (2003).

[7] A. Johnsson, B. Melander, M. Bjorkman *DietTopp: A First Implementation and Evaluation of a Simplified Bandwidth Measurement Method* Proceedings of the 2nd Swedish National Computer Networking Workshop (2004).

[8] NLANR: Iperf v1.7.0 (2004) http://dast.nlanr.net/Projects/Iperf.

[9] Alok Shriram, Margaret Murray,Young Hyun, Nevil Brownlee1, Andre Broido, Marina Fomenkov, and kc claffy, *Comparison of Public End-to-End Bandwidth Estimation Tools on High-Speed Links*, Passive and Active Network Measurement Workshop (2005).

[10] Emanuele Goldoni and Marco Schivi, *End-to-End Available Bandwidth Estimation Tools, An Experimental Comparison*, Traffic Monitoring and Analysis Workshop, 2010.

[11] Carlos Ubeda Castellanos et al., *Comparison of available bandwidth estimation techniques in packet-switched mobile networks*, 2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications.

[12] Ahmed Ait Ali, Fabien Michaut, Francis Lepage *End-to-End Available Bandwidth Measurement Tools: A Comparative Evaluation of Performances*, IPS-MoMe 2006 IEEE.

[13] Dixon Salcedo, Cesar D. Guerrero and Roberto Martinez *Available Bandwidth Estimation Tools: Metrics, Approach and Performance*, International Journal of Communication Networks and Information Security. 10. 580-587.

[14] Alessio Botta, Antonio Pescapè, Giorgio Ventre, *On the performance of bandwidth estimation tools*, 2005 IEE Systems Communications.

[15] Alessio Botta, Alan Davy, Brian Meskill, Giuseppe Aceto, *Active techniques for available bandwidth estimation: Comparison and application*, Data Traffic Monitoring and Analysis, 2013, Springer, Berlin, Heidelberg.

[16] N. Sato, T. Oshiba, K. Nogami, A. Sawabe and K. Satoda, *Experimental comparison of machine learning-based available bandwidth estimation methods over operational LTE networks*, in 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 2017 pp. 339-346.

[17] Ling-Jyh Chen, Cheng-Fu Chou and Bo-Chun Wang, *A machine learning-based approach for estimating available bandwidth*, TENCON 2007 - 2007 IEEE Region 10 Conference, Taipei, 2007, pp. 1-4.

[18] 3GPP, *Study on enhancement of management data analytics*, Reference 28.809, Technical report, Release 16, url: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails .aspx?specificationId=3694

[19] Alessio Botta, Luigi Gallo, Giorgio Ventre, *Cloud, Fog, and Dew Robotics: Architectures for next generation applications*, 2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)

[20] Giovanni Stanco, Alessio Botta, Giorgio Ventre, *DewROS: a Platform for Informed Dew Robotics in ROS*, 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)