Open access • Proceedings Article • DOI:10.1109/CCNC46108.2020.9045301

TOM: a self-trained Tomography solution for Overlay networks Monitoring

Mohamed Rahali, Jean-Michel Sanner, Gerardo Rubino

Institutions: French Institute for Research in Computer Science and Automation

Published on: 10 Jan 2020 - Consumer Communications and Networking Conference

Topics: Network tomography, Overlay network, Edge device, Artificial neural network and Inference

TOM: a self-trained Tomography solution for Overlay networks Monitoring

Mohamed Rahali, Jean-Michel Sanner, Gerardo Rubino

In CCNC'2020

TOM: a self-trained Tomography solution for Overlay networks Monitoring

Mohamed Rahali[1], Jean-Michel Sanner[1], and Gerardo Rubino[2]

[1]Firstname.Lastname@b-com.com, IRT B<>COM, Rennes, France
[2]Gerardo.Rubino@inria.fr, INRIA Rennes – Bretagne Atlantique, France

***Abstract*—Network tomography is a discipline that aims to infer the internal network characteristics from end-to-end correlated measurements performed at the network edge. This work presents a new tomography approach for link metrics inference in an SDN/NFV environment (even if it can be exported outside this field) that we called TOM (Tomography for Overlay networks Monitoring). In such an environment, we are particularly interested in supervising network *slicing*, a recent tool enabling to create multiple virtual networks for different applications and QoS constraints on a Telco infrastructure. The goal is to infer the underlay resources states from the measurements performed in the overlay structure. We model the inference task as a regression problem that we solve following a Neural Network approach. Since getting labeled data for the training phase can be costly, our procedure generates artificial data for the training phase. By creating a large set of random training examples, the Neural Network learns the relations between the measures done at path and link levels. This approach takes advantage of efficient Machine Learning solutions to solve a classic inference problem. Simulations with a public dataset show very promising results compared to statistical-based methods. We explored mainly additive metrics such as delays or logs of loss rates, but the approach can also be used for non-additive ones such as bandwidth.**

I. Introduction

Software Defined Networking (SDN) [1] [2] and Network Function Virtualization (NFV) [3] are new paradigms that enable operators to manage their network resources and handle the increasing demands driven by the new 5G introduction. Operators will therefore have effective tools to better manage their infrastructures and offer scalable and on-demand services to their clients. The NFV concept aims to decouple network functions from the hardware infrastructure by deploying virtualized entities on Custom Off The Shelf servers. This enables creating customized virtual networks to support the varied services. Multiple virtual networks, also called slices [4], can be deployed over a common infrastructure. Therefore, network slicing will enable operators to partition their infrastructure to multiple slices to fulfill the customized requirements of the different clients and devices.

In the slicing concept, there are different levels of resource abstraction. Each layer has an abstracted view of the corresponding resources (for example: physical infrastructure, virtual infrastructure, and multi-domain environment). To ease incident diagnosis, the monitoring system must be able to infer the state of the underlying layer from the information collected in the overlay networks.

Network tomography [5] [6] studies the inference of internal network characteristics from end-to-end measurements. Typical targets are delays, loss rates, or bandwidth. Some approaches allow the discovery of networks topologies [7]. One of the interesting applications is to reduce the complexity of monitoring and diagnostic tool and protocols spread out all over the infrastructure.

Although these solutions provide accurate estimation and try to minimize the computational complexity, they require a holistic view of the network state. In fact, the monitoring strategy should be dynamically adjusted according to network changes which occurs frequently in virtualized architectures. Network tomography solutions fit well with the SDN and NFV paradigm. Indeed, one of the inherent features of the SDN architecture is the global overview of the network. In addition, the virtualized technologies allow deploying scalable and extensible solutions for storing and processing the collected data.

This work proposes a flexible monitoring solution for the inference of additive metrics (delay or loss rate at logarithmic scale) in the underlying infrastructure from end-to-end measurements performed on the exposed abstracted resources inside an NFV-SDN networking architecture. We cross the collected end-to-end measurements to compute the metrics on the shared resources using a trained Neural Network. Our solution can be easily deployed in an SDN-NFV environment and guarantees the scalability and the flexibility of the monitoring system. The proposed architecture is based on virtual resources for data collection and processing. Thus, the update of the monitoring strategy is flexible and not costly.

The remainder of this paper is organized as follows. Section II summarizes the main related works for the monitoring of overlay networks and on network tomography issues. Section III describes the context of network slicing where our solution can be deployed. Section IV gives an overview about the approach we follow based on Neural Networks. The testing environment and the results analysis are presented in Section V. Finally, Section VI concludes this paper and gives an overview of our future work.

II. Background and related works

Overlay network monitoring [8] [9] presents an interesting opportunity for the inference solutions proposed in network tomography. Two main families of approaches can be distinguished: in one of them, exact solutions are the goal,

and the method consists in adding multiple conditions on the traffic collection points and the supervised paths. The related tools are mainly based on algebraic procedures. The other approaches are statistical in nature, and the goal is the estimation of the unknown metrics. This paper can be classified in the second category, even if it is based on Machine Learning instead of statistical techniques.

In [10], the authors propose an algebraic solution to select the necessary paths in an overlay network to fully describe an additive metric on the rest of the available paths. Paper [11] proposes a heuristic algorithm for the placement of overlay networks dedicated to the diagnosis of the underlying network. Other approaches have been published for the inference of additive metrics from end-to-end measurements. These solutions are suitable for the slice monitoring use case. In [12], the authors proposed an algorithm to identify the minimal number of nodes and their placement in a given topology to collect monitoring traffic and to guarantee the identifiability of all the links. The selected points exchange traffic in order to collect end-to-end measurements that will be used to infer the link metrics. Paper [13] deals with the same issue, but the idea is to adapt the monitoring strategy to the network topology changes. These types of solutions usually impose a set of rules for traffic collection points and probing paths. These constraints have to be satisfied in order to ensure proper operations of the monitoring system.

Concerning the methods relying on parametric statistical techniques, the unknown network parameters are considered random variables that follow a specific predefined probabilistic model. The model is then adjusted to fit the collected data. In [14], the authors use multicast probing for loss rate inference. The network topology is fully covered with overlapping multicast trees and unicast paths to collect the end-to-end loss measurements. Then, a Maximum Likelihood Estimator is applied to infer the internal loss rates. In [15], the authors mix unicast and multicast probing to propose the *Flexicast* framework for delay inference in a tree topology. The problem is formulated around a likelihood function where the link delay metrics are latent discrete variables. The likelihood is then maximized with the Expectation-Maximization [16] algorithm. Observe that this procedure is characterized by a slow convergence time that is a drawback when dealing with large-scale networks.

In some recent works, Machine Learning solutions have been proposed for network monitoring and troubleshooting. For instance, [17] uses Supervised Learning to identify link failures. They mix classification techniques with a regression model to locate the failed links. The training is done with multiple features like traffic flow information, Round-Trip Times and loss rates. In this work, we also use Supervised Learning but with automatically generated labeled data. That is, we build a self-trained Neural Network as an inference solution tool for additive link metrics.

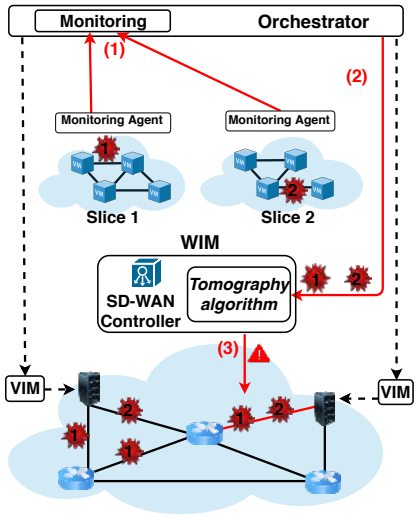


Fig. 1: Slices monitoring

III. Context: Slicing monitoring use case

Network slicing allows creating multiple virtual networks on top of a shared infrastructure. Hence, a failed node in the shared resources impacts the slices that share it. The customer who manages the slice makes regular checks on his deployed services in the virtual infrastructure to verify if the slice provider respects the Service Level Agreement (SLA). The monitoring agent in the slice periodically reports about the measurement performed at the virtual layer to a centralized monitoring unit. This entity can detect the SLA violations and transfer this information to the WAN Infrastructure Manager (WIM). The WIM manages the infrastructure where the slices are deployed. It aggregates the information transferred from the different clients and looks for the root causes of the reported incidents thanks to the tomography algorithm.

Fig. 1 illustrates the steps and the call flow between the monitoring units in the different layers to find the potential causes of an observed SLA violation on different slices:

1. A monitoring agent in each slice performs regular observations on the deployed virtual links.
2. Based on the load and performance measurements of data collected from each agent and its resources verification rules, the orchestrator detects the potential shortages of the virtualized resources and reports them to the WAN Infrastructure Manager (WIM). In the example of Fig. 1, the monitoring system notices an SLA violation in slices 1 and 2.
3. The WIM maps the virtual resources with their corresponding representation in the shared infrastructure. Thanks to the tomography algorithm, the WIM infers the possible root causes to the reported incidents and consequently, it performs remediation actions such as relocation of existing paths.

TABLE I: **List of main used variables**

variable	description
$G = (\mathcal{V}, \mathcal{L})$	network graph topology, node set $\mathcal{V}$, link set $\mathcal{L}$
$\mathcal{P}$	path set, P paths
Y	end-to-end metrics, vector, size P
X	link metrics, vector, size L

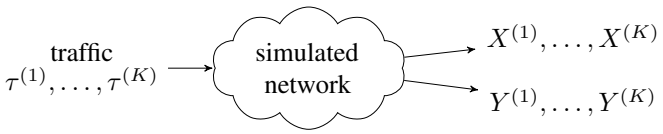


Fig. 2: We inject K different traffic configurations $\tau^{(1)}, \ldots, \tau^{(K)}$ into a simulated network and we measure both path metrics Y and link metrics X, obtaining K pairs $(X^{(k)}, Y^{(k)})$, which constitute our training database. Of course, we can measure only X and deduce Y using (1).

IV. Neural Network-based tomography

A. Network model and notation

Tab. I summarizes the main notation adopted in this paper. Consider the network topology (graph) $G = (\mathcal{V}, \mathcal{L})$ that hosts multiple virtual infrastructures, with the set of nodes $\mathcal{V}$, the set of edges or links $\mathcal{L}$, and denote $|\mathcal{V}| = V$ and $|\mathcal{L}| = L$. Each of these virtual infrastructures is composed of virtual machines connected by virtual links. These virtual links represent an abstraction of paths in the underlying network. Let $\mathcal{P}$ denote the set of paths selected. A path p is represented by a Boolean vector of size L. If link i belongs to this path, $p(i)$ is equal to 1, and to 0 otherwise. Let A be the Boolean matrix whose rows correspond to the paths vectors. Thus, $A(i, j)$ is equal to 1 if j belongs to path p_i, and to 0 otherwise. We denote by Y the vector of size P representing the observed metrics on the virtual paths; $Y(i)$ represents the metric measured or observed on path p_i. X denotes the vector of size L representing the unknown link metrics on the shared infrastructure; $X(i)$ represents the metric on link i. Finally, we assume known an upper bound of those link metrics, denoted by B. Using this notation, the end-to-end metrics can be computed by (1):

$$AX = Y. \tag{1}$$

The objective here is to evaluate X knowing Y and A, that is, to find an appropriate solution to this linear system. The typical situation here is that of an undetermined system (in practice, the number of equations P and the number of unknowns L satisfy, in general, $P < L$). In this paper we focus on this undetermined situation.

B. Additive metrics inference with Neural Networks

Our proposal is to transform the inference inverse problem described by (1) into a regression problem and to solve it using a Neural Network. For this purpose, since we know the architecture of the network, we can simulate the distribution of traffic through the network in many configurations, and observe many pairs (X, Y). This can then be used to learn the connection between the two vectors, in the sense $Y \to X$, following a Machine Learning approach. Figure 2 resumes this process.

We will consider a classic Neural Network architecture of the Feed Forward type, and for our tests here, with only one hidden layer. Each of our layers is fully connected to the next one (or to outside for the third layer), that is, both the neurons in the hidden and the output layers receive input from each of the neurons in the previous layer. The dimension of the input layer is equal to the number of paths P while L, the number of links, is the number of outcomes from the output layer. The dimension of the hidden layer is a variable parameter that we adjust for optimizing the performance. This happens, as expected, for a number of hidden units much larger than the dimension of the input.

C. Simulated traffic for the training phase

The training of the Neural Network requires a large volume of labeled data. In our problem, the features are the end-to-end measurements and the labels are the link metrics. Hence, constructing the dataset requires collecting exhaustive link level metrics in a small time window for each input example. This process should be repeated in different network conditions to avoid the over-fitting of the model. Thus, collecting real data for the training introduces an important overhead in the process. This scenario seems to be unrealistic since the objective of the monitoring operation is to afford useful information about the network state without disturbing the network functioning. To avoid this issue, we train our Machine Learning tool using simulated data.

The Neural Network has to learn how to approximate the link metrics from the path measurements. In other words, it must capture the spatial correlations created by the topology that enable to find relations between the end-to-end measurements and the link metrics. In addition, the Neural Network should not be over-fitted to some specific values and should take into consideration the temporal variability of the link metrics. We propose to generate exhaustive random samples of link metrics denoted X'. In fact, with each link metric, we associate a random value between 0 and B to construct one example of X'. Its associated simulated end-to-end measurement Y' is computed using $Y' = AX'$. This procedure is repeated multiple times to construct an important number of couples (X', Y'). The computed end-to-end metrics Y' will be the features of the training, while the labels will be the generated link metrics examples X'.

D. Training step

The inference of link metrics from path measurements can be considered as a regression problem. The input is the path-level measurements and the expected output is the link-level metrics estimation. We train the Neural Network using the iterations of forward and backward propagation with the

artificially created data. In our experiments, we use the classic Adam optimizer algorithm [18] for backward propagation. It is possible to pass all the training examples multiple times through the forward and backward process. An *epoch* in the learning process represents one forward and one backward propagation.

E. Testing step

After training the neural network with artificial data, we use it to estimate the link metrics from the path measurements. Observe that if there is a change in the network topology or the used paths, we have to repeat the training step taking into consideration the new updates. Observe also that this is a one-shot step, done only once. Using the trained Neural Network in the operational phase is then a quick procedure.

V. Model evaluation and result analysis

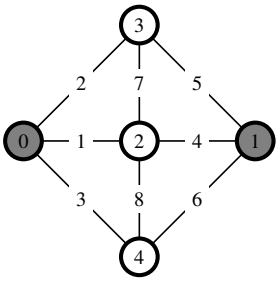
A. Singular Value Decomposition-based reference method

To illustrate the performances reached of our proposal, we compare it with a basic reference solution [10] based on Singular Value Decomposition (SVD) [19]: basically, every matrix A of dimension $m \times n$ can be written as $A = USU'^{\mathrm{T}}$ where U is an $m \times m$ unitary matrix (that is, $U\,U^{\mathrm{T}} = I$), S is an $m \times n$ diagonal matrix whose diagonal is composed of the singular values of A (the square roots of the eigenvalues of matrix $A^{\mathrm{T}}A$) and U' is an $n \times n$ unitary matrix. The pseudo-inverse of matrix A, denoted A^{+}, is $A^{+} = U'S^{+}U^{\mathrm{T}}$, where S^{+}, the pseudo-inverse of matrix S, is the transpose of the matrix obtained by replacing the non zero singular values of A in S by their corresponding inverses. The link metrics can then be computed by multiplying the pseudo-inverse A^{+} by the end-to-end measurement vector Y:

$$X = A^{+}Y. \tag{2}$$

We use this method to compare it with our proposal to show the efficiency of Machine Learning solutions to solve a classic statistical problem. Comparing the performances of the different Machine Learning models in this inference task is in the scope of our future work.

B. Simulation environment



Paths:
0-3-1
0-2-1
0-4-1
0-4-2-3-1
0-2-3-1
0-2-4-1

Fig. 3: Topology A

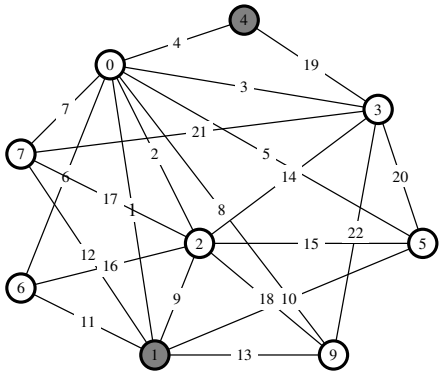
In order to evaluate the performances of our procedure TOM, we tested it using two different topologies taken from [20]. For each topology, we selected two nodes to exchange the monitoring traffic and we considered a predefined list of paths. Thus, these points correspond to the data centers where the virtual machines are deployed as described in Fig. 1. The used paths represent the mapping between the virtual links and their corresponding representations in the shared infrastructure. The two topologies are provided with a dataset of multiple samples of delay measurements performed on the different links. The traffic was simulated with the `Omnet++4` network emulator. For each topology, we computed the end-to-end delay on each path, that is, the sum of the delays on the links composing them, and we estimated the link metrics.

The error is evaluated using (3):

$$\text{Error (in \%)} = 100\frac{|V^{\text{estimated}} - V^{\text{real}}|}{B}, \tag{3}$$

where V^{real} is the exact value of the metric V we are interested in, $V^{\text{estimated}}$ is the estimation, and B is an upper-bound of V.

The first topology, shown in Fig. 3, has five nodes and eight edges. The second, depicted in Fig. 4, is composed of nine nodes and twenty-two edges.



Paths:			
4-3-7-0-5-1	4-0-2-6-1	4-3-7-2-1	4-0-5-2-1
4-0-2-9-1	4-0-2-7-1	4-0-3-2-1	4-3-9-2-1
4-3-7-2-1	4-0-9-1	4-0-2-5-1	4-0-6-2-1
4-0-7-2-1	4-3-5-1	4-3-0-1	4-0-2-1
4-3-7-1	4-0-6-1	4-3-9-1	4-0-7-1
4-3-2-1	4-0-5-1		

Fig. 4: Topology B

We made multiple tests to compare the results of our approach with the method described in Subsection V-A. We also evaluate the accuracy of the estimations and the computing time as a function of different criteria like the number of layers and their sizes, the size of the training data-set, the activation function chosen and the number of epochs in the learning process. After these experiments, we concluded, for instance, that a single hidden layer was enough to obtain a good performance (see V.C.2 below).

C. Results

We evaluate the performances of our solution regarding different parameters. The variation of the activation function and the number of epochs does not have a significant impact on the results. In the next tests, we use the Rectifier Linear

Unit (*ReLU*) activation function and three epochs for the training phase. The errors indicated in the different figures are given in relative terms and in % as described by (3).

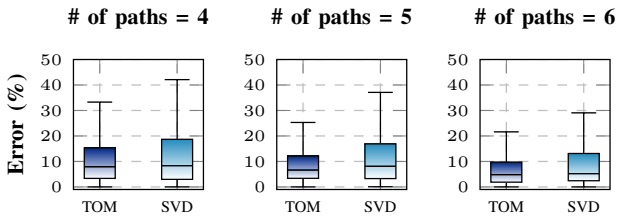


Fig. 5: Topology A: error vs # of paths

1) Used paths: In these illustrations, we use only one hidden layer, as previously stated, and we fix the number of examples in the training data set to $5 \cdot 10^5$.

For a fixed number of paths in one test, we select the first ones from the predefined list. We evaluate the performances of our proposal with two topologies and we compare it with the SVD-based solution. Fig. 5 and Fig. 6 illustrate the dispersion of the percentage error of the two solutions with the two studied topologies A and B.

Increasing the number of paths obviously enhances the accuracy of the solutions. The TOM technique has always a better estimation accuracy. The difference is more visible with topology B. In fact, with topology A and 6 paths, the median of the absolute error with our Neural Network is 4.8%, while it is 5.2% with the SVD-based solution. With topology B, when we use for example 19 paths, the median is 5.9% for the TOM method, while it is 22.7% when using the SVD.

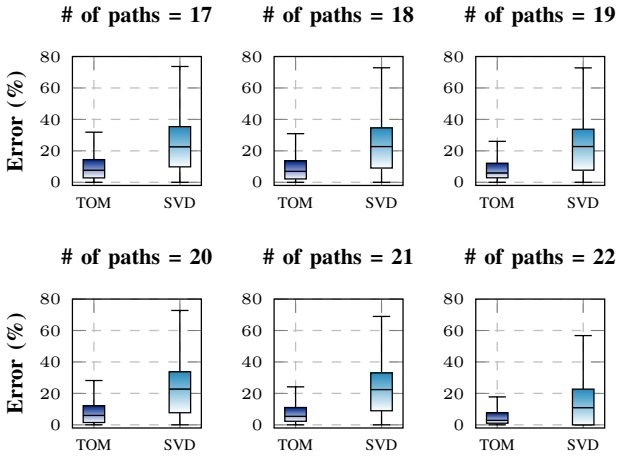


Fig. 6: Topology B: error vs # of paths

Using only 6 paths with topology A gives an under-determined linear equation system. Thus, multiple solutions are possible. However, there are some links that are sufficiently covered by the set of paths so that the link metric can be computed exactly. Both the Neural Network and the SVD solutions compute these metrics with good accuracy. The difference is significant on the other links. In fact, the SVD-based approach gives one possible solution that satisfies the linear equation system, but not necessarily one close to the optimal one. The Neural Network is trained to choose the best solution that minimizes the absolute error. The difference between the two solutions is more evident when using topology B, because of the larger dimensions. The implicit covering of the correlations between the observed path metrics, given the topology of the network and the set of paths selected, done by Machine Learning technology manifests more clearly in these situations, when networks approach more realistic sizes.

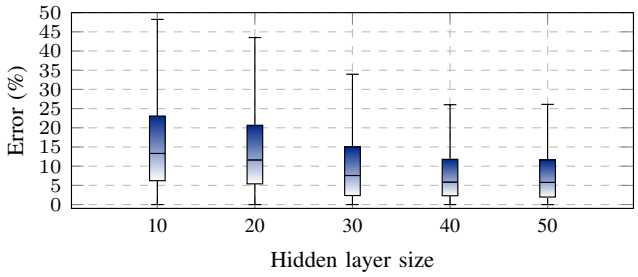


Fig. 7: Topology B: error vs hidden layer size

2) Hidden layers: In this section, we study the impact of the number of hidden layers on the estimation accuracy and the computing time.

In these tests, we use topology B with 19 paths and the number of examples in the training data set is fixed to $5 \cdot 10^5$.

The main conclusions of this set of experiments are as follows. Firstly, varying the number of the hidden layer does not impact significantly the accuracy of the estimations. That is why we used only one large middle layer in our final evaluations. This doesn't preclude future tests with deeper architectures (see the Conclusions), but our goal here is to illustrate the approach. Secondly, we study the effect of the hidden layer size. Fig. 7 shows the variation of the absolute error regarding the hidden layer size. Increasing the hidden layer size enhances accuracy until reaching about 40 neurons. From this value, the error stagnates. The training time increases more or less linearly with the hidden layer size (see Fig. 8), increasing slightly at the end of the considered range. All these observations fit with the usual behavior of these technologies.

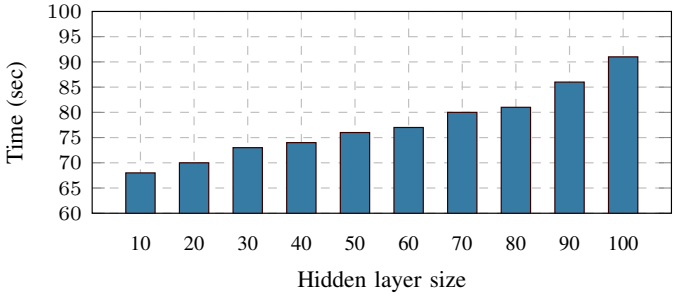


Fig. 8: Topology B: training time vs hidden layer size

3) Training data-set: The size of the generated data-set for the training is another important parameter for the accuracy and the training time. In these tests, we use topology B with 19 paths and only one hidden layer. We study the effect of the number of training examples with three datasets of sizes 10^4, 10^5 and 10^6 respectively as described in Fig. 9.

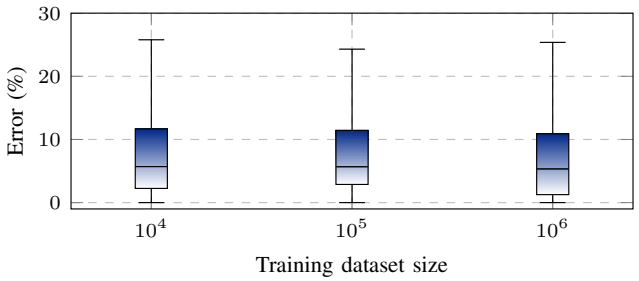


Fig. 9: Topology B: error vs dataset size

The training data size has a similar impact as the hidden layer size. Increasing the number of training examples enhances the accuracy of the estimations until reaching the best performance. Then, adding additional examples for the training does not impact the accuracy as shown in Fig. 9, it increases only the training time (see Fig. 10).

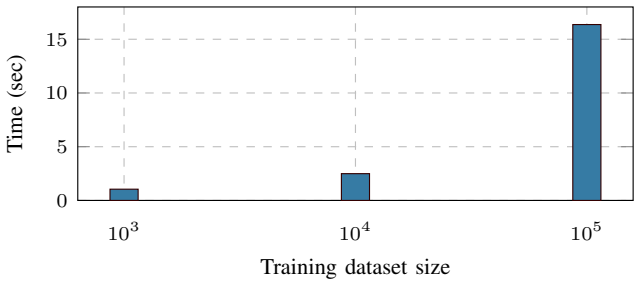


Fig. 10: Topology B: training time vs dataset size

VI. Conclusions

In this paper, we propose a Neural Network solution called TOM for the inference of metrics in networks. The described work has been done in the context of SDNs, but it can be exported to other environments. We used additive metrics such as delays or logarithms of loss rates, leading to a linear algebraic context, but the approach can be used for other metrics such as bandwidth. One of the main features of our proposal is the use of simulated data in the training step. In addition, the learning phase can be carried out in a very short time period, which allows to easily manage changes in topologies. We used an emulated network traffic to evaluate the performances of our procedure. The results show that our Machine Learning approach gives better estimations than the pseudo-inverse statistical method.

Many points remain to be explored. Choosing an appropriate set of paths is one of the main open issues in this type of approach. The exploration of the learning tools used was just initiated in our work; a deeper exploration of the techniques available and their possibilities for our problem is one of our possible future tasks.

References

[1] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[2] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, 12 2017.

[3] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, 09 2015.

[4] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, July 2016.

[5] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statistical Science*, vol. 19, no. 3, pp. 499–517, 2004.

[6] E. Lawrence, G. Michailidis, V. N. Nair, and B. Xi, "Network tomography: A review and recent developments," in *In Fan and Koul, editors, Frontiers in Statistics*. College Press, 2006, pp. 345–364.

[7] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak, "Efficient network tomography for internet topology discovery," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 931–943, June 2012.

[8] A. Dusia and A. S. Sethi, "Recent advances in fault localization in computer networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 3030–3051, Fourthquarter 2016.

[9] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, pp. 165–194, 11 2004.

[10] Y. Chen, D. Bindel, and R. H. Katz, "Tomography-based overlay network monitoring," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '03. New York, NY, USA: ACM, 2003.

[11] M. Demirci, F. Gillani, M. Ammar, and E. Al-Shaer, "Overlay network placement for diagnosability," in *2013 IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 2236–2242.

[12] A. Gopalan and S. Ramasubramanian, "On identifying additive link metrics using linearly independent cycles and paths," *IEEE/ACM Transactions on Networking - TON*, vol. 20, pp. 906–916, 06 2012.

[13] T. He, A. Gkelias, L. Ma, K. K. Leung, A. Swami, and D. Towsley, "Robust and efficient monitor placement for network tomography in dynamic networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1732–1745, June 2017.

[14] C. Liu, T. He, A. Swami, D. Towsley, T. Salonidis, A. I. Bejan, and P. Yu, "Multicast vs. unicast for loss tomography on tree topologies," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, 2015, pp. 312–317.

[15] E. Lawrence, G. Michailidis, and V. N. Nair, "Network Delay Tomography Using Flexicast Experiments," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 68, no. 5, pp. 785–813, 2006.

[16] T. K. Moon, "The Expectation-Maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, Nov 1996.

[17] S. M. Srinivasan, T. T. Huu, and M. Gurusamy, "Machine learning-based link fault identification and localization in complex networks," *CoRR*, vol. abs/1812.03650, 2018.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[19] G. H. Golub and C. Reinsch, *Singular Value Decomposition and Least Squares Solutions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1971, pp. 134–151.

[20] A. Mestres, A. Rodríguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, G. Estrada, K. Maruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. D. Evans, F. Maino, J. C. Walrand, and A. Cabellos-Aparicio, "Knowledge-defined networking," *Computer Communication Review*, vol. 47, pp. 2–10, 2017.