SABES: Statistical Available Bandwidth EStimation from passive TCP measurements

Francesco Ciaccia*†, Ivan Romero†, Oriol Arcas-Abella†, Diego Montero†‡, René Serral-Gracià*, Mario Nemirovsky†
*Universitat Politècnica de Catalunya (UPC), Barcelona, Spain {fciaccia, rserral}@ac.upc.edu
†Clevernet Inc., San Francisco, USA {fciaccia, oriol, iromero, mario}@clevernet.io
‡Universidad de Cuenca, Cuenca, Ecuador {diego.monterob}@ucuenca.edu.ec

***Abstract*—Estimating available network resources is fundamental when adapting the sending rate both at the application and transport layer. Traditional approaches either rely on active probing techniques or iteratively adapting the average sending rate, as is the case for modern TCP congestion control algorithms. In this paper, we propose a statistical method based on the inter-packet arrival time analysis of TCP acknowledgments to estimate a path available bandwidth. SABES first estimates the bottleneck link capacity exploiting the TCP flow slow start traffic patterns. Then, an heuristic based on the capacity estimation, provides an approximation of the end-to-end available bandwidth. Exhaustive experimentation on both simulations and real-world scenarios were conducted to validate our technique, and our results are promising. Furthermore, we train an artificial neural network to improve the estimation accuracy.**

***Index Terms*—Available bandwidth, network machine learning, passive probing**

I. Introduction

Nowadays, improving the Quality of Experience of final users is of paramount importance. To this end, Software-Defined Networking (SDN) opens new horizons for network devices to be smart, and to take advantage of the network deep contextual knowledge. Network stacks in endpoints and edge routers play an important role in managing the available network resources, but they usually lack visibility. Bandwidth adaptation, on the other hand, has always been one of the fundamentals research topics in both routing architectures and transport protocols. As of today, endpoints can infer the available bandwidth from their achieved TCP throughput; however, TCP throughput does not necessarily reflect the network available bandwidth, depending mostly on the congestion control strategy adopted. Another way to estimate it is to actively generate probes with specific traffic patterns as extensively documented in literature [1]–[6]. We detect some limitations in both these approaches. TCP throughput can be misleading when estimating the available bandwidth. Congestion control algorithms use specific metrics to interpret congestion and adapt their sending rate iteratively. Loss-based congestion controls react to packet loss. Such algorithms, combined with the big buffers of modern routers, can considerably degrade the performance of connections, causing bufferbloat [7]. Their behavior greatly affects the link utilization, finally impeding a correct estimation of the available network resources, as the network state is modified by their operation. More recent developments in TCP congestion control, such as Google's BBR [8], take another estimation strategy, much more promising and effective. However, its way of probing for available bandwidth still affects the network under analysis, consequently altering the available bandwidth estimation as detailed in Section II. On the other hand, many of the developed active probing methods are not suitable for real-world estimation. Their biggest limitation is that they usually need to be deployed in both endpoints of the connection, making them unsuitable for single-sided analysis, especially on routers. Finally, most methodologies assume a fluid cross traffic model, defeating their accuracy in scenarios with highly variable cross traffic patterns.

In this study, we propose a Statistical Approach to Available Bandwidth EStimation, – SABES –, a passive probing method based on the study of the inter-packet arrival time of TCP acknowledgments. Its main characteristics are:

- computationally inexpensive and therefore suited to real time analysis,
- does not need to be deployed at both endpoints,
- no assumptions about the cross-traffic model are made, is data-driven and based on an extensive set of simulations and real-world scenarios,
- uses a neural network to filter out misleading measures.

Possible applications of SABES include active congestion control strategies e.g., active queue management at an edge router or congestion control at the endpoint, and available bandwidth analysis. While it relies on accurate TCP segments timestamping, we demonstrate that our filtering technique makes it robust even in virtualized environments.

The rest of the paper is structured as follows: Section II discusses previous work in available bandwidth estimation both as part of TCP congestion controls and as active probing methodologies. Section III describes the statistical analysis carried on to build SABES model and the derived heuristic algorithm. Section IV shows results of the estimations obtained over large simulated datasets. Section V describes the approach taken to improve the accuracy of SABES by means of Deep Neural Networks. Section VI reports the results obtained by SABES in a real testbed. Section VII exposes future work and final remarks.

II. State of the art in Available Bandwidth estimation in TCP and Active Probing

Google's BBR TCP congestion control operates on the Kleinrocks optimal operating point [9] in which the available bandwidth and the round trip time, RTT, are estimated in order to determine the bandwidth-delay product (BDP). The endpoint probes periodically to estimate the tight link available bandwidth by pacing packets at higher rates than the previous estimation. However, BBR has been proven to build long-term standing queues that can cause misleading BDP estimation [10]. This causes the algorithm to often overestimate the BDP while not being fair to other competing flows [11] SABES available bandwidth estimation strategy could be used as part of a congestion control scheme that preserves higher fairness characteristics such as those guaranteed by loss-based congestion controls. The SABES approach entails detecting moments when the TCP sending rate is higher than the available bandwidth, and analyzing the ACKs inter-packet distribution to detect its value at that time. Under normal TCP operating conditions, this type of behavior can provoke phases where the flow self-induced congestion is still not causing adverse events such as buffer-bloat or packet loss; to estimate the available bandwidth during those phases is beneficial. Such approach is also taken by TCP HyStart [12]: it is a heuristic to find a safe exit point for TCP slow start which infers the available bandwidth based on the clocking of TCP ACKs. HyStart mitigates the losses caused by slow start and it is used as the default slow start algorithm for the CUBIC congestion control; their heuristic includes packet trains being sent back-to-back during the slow start. In addition to what HyStart does, SABES filters out statistical anomalies in clocking of TCP ACKs. SABES can run in real-time during the whole lifetime of the TCP connection and find adequate moments to estimate available bandwidth. SABES is tested with TCP flows using CUBIC in Section VI.

The *probe-gap model* (PGM) and the *probe-rate model* (PRM) are extensively covered in the literature and either one or the other are used for most active probing techniques aimed at available bandwidth estimation. The idea behind PGM is to send packet pairs at a rate equal to the one of the lowest capacity link (narrow link) to then estimate the rate of the lowest available bandwidth link (tight link). As such, PGM assumes that capacity is known in advance. A few tools implementing PGM foresee an exploratory phase in which capacity is inferred. SABES does the same, exploiting some characteristics of TCP slow start to infer the narrow link capacity. Tools implementing PGM include Pathrate [1] and Spruce [2]. Based on PGM is [3], a work that takes a similar approach as SABES although with some very relevant differences: even though they use TCP acknowledgments to estimate available bandwidth, they take into account both the sending rate of the data packets and the receiving rate of the ACKs, following the probe-gap model proposal. SABES looks exclusively to the ACKs clocking and its distribution to determine the available bandwidth, making it suitable for receiver-side-only TCP based estimation; to the authors knowledge, no other tools implements such an approach. Also, SABES does not need information provided by the connection socket or by the operating system, making it suitable for being deployed in routers. PRM methods are also called *iterative* methods, as they gradually decrease the time interval between the probes or trains of probes to detect the probe-gap curve bending point, thus they do not take assumptions on the link capacity. While methods like these have been proven to be more accurate than PGM-based methods, they generally require more time and generate more probing traffic. During this period of time, cross traffic can considerably vary, causing incorrect estimations. Examples of methods implementing a PRM approach are TOPP [4], pathChirp [5], and BART [6]. pathChirp was one of the first tool developed that used the concept of packet *chirps* which consist of packet trains sent with an exponentially decreasing inter-packet time. This approach aimes at generating self-induced congestion in the tight link queue, causing increasing queueing delays. A similar effect is obtained with TCP flows, especially the ones using loss-based congestion controls such as CUBIC, and SABES takes advantage of that.

III. SABES Heuristic

The idea behind SABES is to evaluate the inter-packet arrival time of the acknowledgments of a TCP flow. The ACK number carried in a packet allows the computation of the actual inter-packet rate of the packets that generated each specific ACK pair. For each ACK pair with ACK number ack_{num} we define the the inter-ack bytes as $\delta_{acks} = ack_{num_i} - ack_{num_{i-1}}$ and the inter-packet time δ_t is then: $\delta_t = ts_i - ts_{i-1}$. ts is the ACK timestamp taken either on the egress interface of the receiver or the ingress interface of the sender. We finally define the inter-packet rate - IP_{rate} as:

$$IP_{rate} = \frac{ack_{num_i} - ack_{num_{i-1}}}{ts_i - ts_{i-1}} = \frac{\delta_{acks}}{\delta_t} \quad (1)$$

Duplicated and out of order ACKs-derived measurements are discarded. Figure 1 provides a graphical representation of the inter-packet gap/rate model using TCP acknowledgements. IP_{rate} can be measured both in sender and receiver. To be noted that in real systems, TCP stacks implement cumulative acknowledgements. Measuring the inter-ACK gap for every ACK-pair comprises the behavior of two or more original packet-pairs sent by the sender. Using the measurements obtained with Equation 1, we split the algorithm into two phases. The first phase tries to infer the connection narrow link capacity at the beginning of a TCP connection, while the second one tries to continuously estimate the available bandwidth.

A. Capacity Estimation

At the beginning of a TCP connection, while the sender congestion window is still growing, packets are sent in bursts with an inter-packet rate that approximates the sender NIC link speed; at that time, packets are said to be sent *back-to-back*

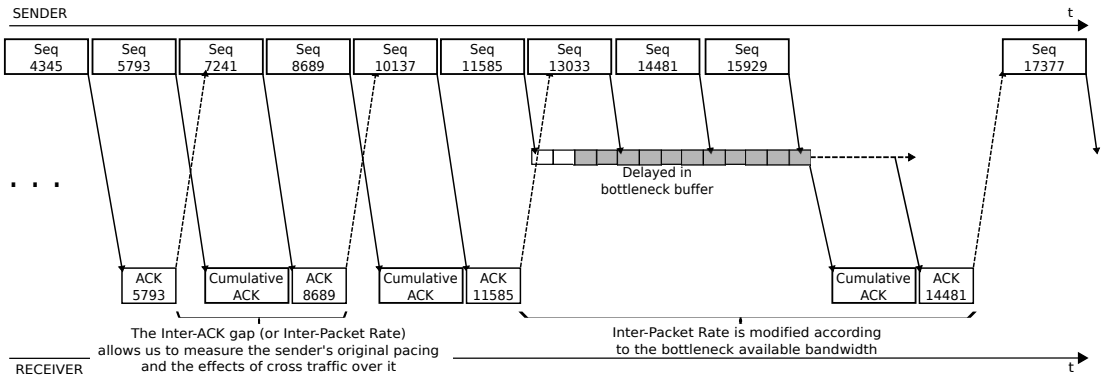
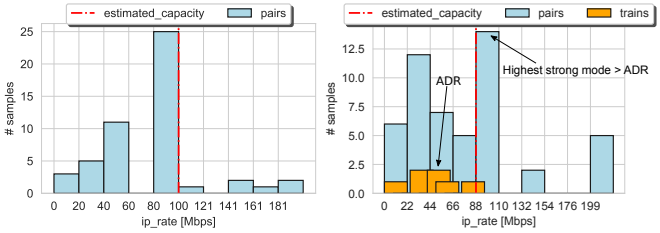


Fig. 1: The Inter-Packet Rate is the rate of the originally sent data packets that is possible to infer from their TCP acknowledgments spacing.

and cross traffic is less likely to interfere with such packets; cross-traffic interference is more likely when TCP slow start has grown considerably or during TCP congestion avoidance. However, if there is a link with lower capacity than the sender NIC, packets will be queued before entering this link and the inter-packet rate measured on the acknowledgements will be paced accordingly. We base our capacity estimation technique on the one proposed by *pathrate* in [1]. We study the IP_{rate} distribution of the first α ACKs. Valid values for α vary according to the link capacity, the TCP slow start algorithm, the connection RTT, and the amount of cross traffic in the tight link. We always assume that the TCP flow in analysis is a bulk data transfer and all packets are the size of TCP MSS. We study a worst-case scenario, looking for the minimum amount of packets for α that makes a correct capacity estimation possible. We find that for a 100Mbps narrow link, with 90% cross traffic from multiple Pareto sources, 140ms RTT, and multiple competing flows, that at least 10 acknowledgments are needed to estimate the capacity. To this end, we bind the formula to the NIC capacity C_n, which we assume we can know in advance, being 1Gbps for this specific case. We derive then:

$$\alpha = \frac{C_n}{2 \cdot MSS \cdot RTT} \cdot \epsilon \tag{2}$$

where ϵ is a constant derived from the data exploration of the previously cited worst case scenario. With such an α value we are able to complete the capacity estimation between 3 and 30 RTTs, with less cycles for larger RTTs. In this way, we can capture enough information to build the histogram statistic while not delaying the capacity estimation to a late moment in the connection lifetime. Being the first packets of a TCP connection sent back-to-back, we can follow the conclusions of [1] and analyze if the distribution highest mode represents more than $\beta\%$ of the total sample data. We choose $\beta = 30\%$ as shown in [1]; this applies to non-congested scenarios where a strong capacity mode is formed, as shown in Figure 2a. If the distribution is not found to have a single high mode, we proceed by studying the IP_{rate} generated by ACK-trains: we consider every N acknowledgements instead of each pair and perform the same computation described in Equation 1. This is equivalent to using packet trains as done in *pathrate*. We look at ACKs trains of size $N = 4$. In this scenario,



(a) Capacity estimation is obtained from single strong mode in packet pairs

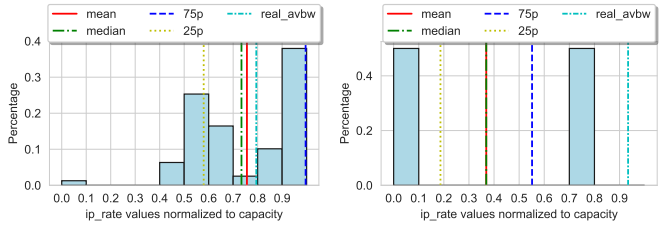
(b) Capacity estimation is obtained from packet trains dispersion

Fig. 2: Capacity estimation following SABES heuristic.

IP_{rate} converges closer to the flow throughput; in [1] a similar concept is identified as Asymptotic Dispersion Rate (ADR). We pick the ADR value as the main mode of the distribution obtained from ACKs trains. We then look for the first mode in the original set of modes obtained from ACK pairs and select the first mode with a value higher than the ADR as capacity value as show in Figure 2b. The accuracy of this technique is affected by the bin size selected. In our case, we fix the number of bins to 10 instead of selecting a single bin size.

B. Available Bandwidth Estimation

SABES main objective is to first identify specific moments in which the IP_{rate} measurements are valid to be used to estimate available bandwidth. Since we are not controlling how the probes are generated, the packet pacing applied by TCP is not always adequate to proceed with the estimation. For example, this happens when the flow has a very low throughput, as per the application generating the traffic itself (e.g. interactive flows or constant bit-rate flows) or because of very high level of concurrency on the machine generating traffic (e.g., more than hundreds of bulky flows). A sliding-window mechanism based on the connection Round-Trip Time (RTT) is used. When measuring in the sender the exponentially smoothed RTT $sRTT$ is used. In the receiver, the RTT used for the sliding window is measured during the three-way handshake. The dynamic sliding window $\mathcal{D}_i$ of the ACKs



(a) Mean is representative of the available bandwidth (b) Mean is not representative of the available bandwidth

Fig. 3: Example of IP_{rate} distribution that matches SABES criteria obtained with the dynamic sliding window.

IP_{rate} is obtained as defined in Eq. 3:

$$\mathcal{D}_i = \{p_k.ip_{rate} | ts_{p_k} \in \mathcal{W}_i\} \quad (3)$$

$$\mathcal{W}_i = [ts_{p_i} - \min(1sec, sRTT_i), ts_{p_i}] \quad (4)$$

where ts is the packet timestamp, p_i is the last received ACK and $sRTT_i$ is the smoothed RTT computed up to the last RTT measurement. The maximum window horizon is set to 1 second behind ts_{p_i} as defined in Eq. 4. The result is filtered to remove statistical outliers that can derive from noisy measurements, using a simple inter-quartile range rule over the window in analysis. We discard duplicated ACKs. We finally filter any inter-packet time bigger than twice the RTT, as they are usually associated with retransmission timeouts or slowly growing congestion windows. We use the samples selected through this dynamic sliding window mechanism - $\mathcal{D}_i$ - to build a statistical distribution of the inter-packet rate in form of an histogram. We studied this histogram for a big simulation data-set and derive an heuristic to infer when the IP_{rate} follows specific patterns. Given $\mathcal{D}_i$ we define its inter-quartile range as $\mathcal{I}(\mathcal{D}_i) = q_{75}(\mathcal{D}_i) - q_{25}(\mathcal{D}_i)$ and its mean-median distance as $\mathcal{M}(\mathcal{D}_i) = |q_{50}(\mathcal{D}_i) - \bar{x}(\mathcal{D}_i)|$, where $q_y()$ and $\bar{x}()$ are the y-quantile and mean operator, respectively. From extensive data-analysis of both simulations and real-world traces we detect that when $\mathcal{M}(\mathcal{D}_i)$ is small and $\mathcal{I}(\mathcal{D}_i)$ is big, $\bar{x}(\mathcal{D}_i)$ is representative of the network available bandwidth. We finally define two parameters ω and γ as the mean-median maximum distance and the minimum distribution spread, respectively. Both are derived from the capacity $\mathcal{C}$ estimated in III-A; we experimentally find that $\omega = 0.1 \cdot \mathcal{C}$ and $\gamma = 3 \cdot \omega$ allow to identify distributions that meet our criteria. The set $\mathcal{B}$ of all available bandwidth estimations obtained from a single TCP flow is defined as:

$$\mathcal{B} = \{\bar{x}(\mathcal{D}_i) \,|\, \mathcal{M}(\mathcal{D}_i) \leq \omega \quad and \quad \mathcal{I}(\mathcal{D}_i) \geq \gamma\} \quad (5)$$

In Figure 3a is shown an example of valid IP_{rate} sliding histogram following the heuristic criteria. The samples are normalized according to the estimated link capacity. In this case, the mean value actually approximates the available bandwidth with an error of 3.5%.

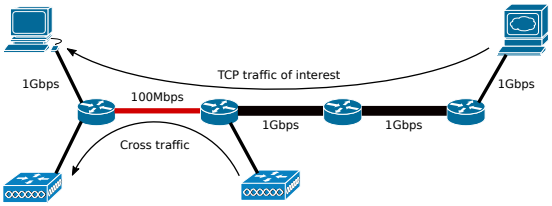


Fig. 4: Simulated topology with one single 100 Mbps bottleneck and hop-persistent cross-traffic flowing in the same direction of the main TCP data flow.

IV. Heuristic Evaluation

A. Simulation environment

To validate our model, we simulated a network scenario in ns-3 [13]. The simulated dumbell topology consists of multiple intermediate routers and one single bottleneck link at 100 Mbps, as shown in Figure 4. All the other links have 1 Gbps capacity. We simulate scenarios with different values of end-to-end latency, varying between 20ms and 140ms. Cross-traffic is traverses only the bottleneck link (is said to be one-hop persistent). The TCP traffic of interest goes through all topology up to the receiver. Cross-traffic and traffic of interest are generated from different sources and received in different sinks. The cross-traffic generators simulated are both constant bit-rate and on-off Pareto sources. In both cases, cross-traffic is generated so to use 20%, 50%, and finally 90% of the bottleneck capacity. We generate tests with one single cross-traffic flow, up to 30 concurrent cross-traffic sources; cross-traffic flows duration is continuos until the end of the TCP transfers. The TCP traffic of interest flows are bulk data transfers of 1MB, 5MB and 50MB. We test generating only one TCP flow, with 20 and up to 50 concurrent TCP flows generated from the same host. TCP congestion control algorithm is the loss-based TCP-Reno (default in ns-3).

B. Heuristic simulation results

We show qualitative results for the heuristic in a single TCP flow with 20% Pareto cross-traffic scenario in Figure 5. Blue dots represent SABES estimations. The heuristic provides correct estimations based purely on acknowledgments-derived measurements in moments where the actual TCP throughput is far from matching the real available bandwidth. As a matter of fact in this scenario the loss-based TCP congestion control is never converging to an optimal link utilization. As a matter of fact, this flow TCP throughput diverges from the real available bandwidth with an error of 30Mbps in its congestion avoidance phase. On the other hand SABES heuristic is able to detect when the IP_{rate} measurement is representative of the available bandwidth and the mean absolute error of the estimation set $\mathcal{B}$ is 7 Mbps.

We show cumulative results with the boxplot in Figure 6a. Estimations are computed over a validation data-set consisting of highly competitive simulations scenarios, with 50 TCP concurrent flows and 30 cross-traffic Pareto sources, using up to 50% and then 90% of the bottleneck capacity. We test this in a topology with an end-to-end RTT of 20ms and then 140ms.

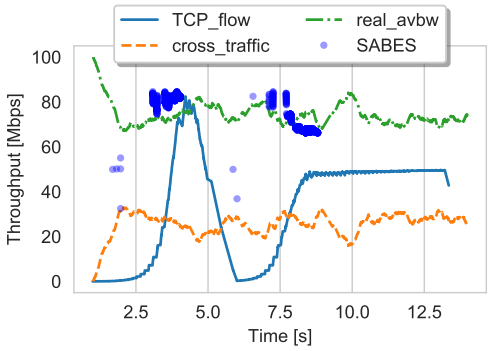
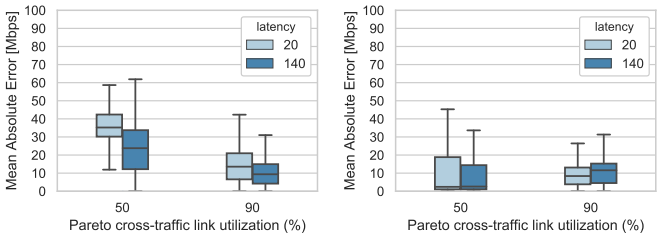


Fig. 5: Heuristic application to a single TCP flow, single bottleneck scenario. The mean absolute error of $\mathcal{B}$ is 7.4Mbps.

Bottleneck capacity is 100Mbps. We show the distribution of the mean absolute error of the estimation in Figure 6a. Results improve in the 90% cross-traffic scenario in respect to the 50%. We deduce that the better estimation obtained in the 90% cross-traffic scenario is due to the average TCP throughput being closer to the actual available bandwidth; in these cases the TCP congestion control is able to converge to a fairly precise value, improving the IP_{rate} statistic. On the other hand the estimations accuracy is independent on the RTT. Results obtained provide good confidence margins in the estimation with a computationally lightweight process implementable as part of a real-time system. SABES can be deployed both close to the sender or the receiver and its estimations are obtained exclusively from passive TCP analysis. However, the results distribution for the 50% cross-traffic case shows a median error value of approximately 30Mbps, which is rather high. In Section V, we investigate the problem and propose an AI-enhanced solution, improving estimation results considerably.

V. Neural Network Evaluation

The heuristic described in Section III tries to identify distributions shaped like the one in Figure 3a, looking for a mean value which is representative of the available bandwidth. However, there are distributions that match the heuristic criteria but that do not contain any significant mode close to the available bandwidth, such as the one shown in Figure 3b. The mean of such a distribution gives a 50% error in the estimation and no actual values are present in the bin closest to the distribution mean. The histogram shapes of Figure 3a and Figure 3b are notably different. To distinguish between them we train an artificial neural network. ANNs have proven to be effective as classifiers when the input features are non-linear and with high dimensionality, such as the sliding histograms produced by SABES heuristic. We train a neural network to classify good and bad histogram distributions; the generalization is obtained by normalizing the data to the estimated capacity and the total number of samples obtained in the sliding window. In this way, the neural network input data is independent of the link capacity and the sample size. We fix the histogram bins number to 10. This translates in a lower resolution when representing IP_{rate} distributions of high capacity links (i.e. more than 500Mbps), although we opt for this compromise to keep the neural network of reasonable size.



(a) Heuristic only. (b) Neural Network.

Fig. 6: Mean absolute error results of the estimations applying SABES in a simulated environment with 100Mbps bottleneck for different values of cross-traffic link utilization and latency.

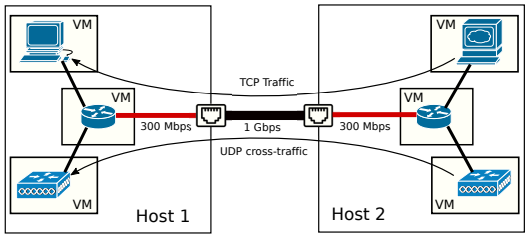


Fig. 7: Real testbed topology with one single 300Mbps bottleneck.

We run SABES heuristic and get the complete normalized sliding histograms of the simulations described in Section IV-A. We obtain over 140000 sliding histograms used as training data-set for the neural network. We run a supervised-learning training process labeling all the histograms *good* whose mean value provides an available bandwidth estimation with an error lower than $\pm 10\%$ of the bottleneck capacity, and as *bad* the other histograms. We identify that distributions that overestimate or underestimate have a similar shape, as far as the estimation error is small. The neural network used is a deep neural network, consisting of two hidden layers. The network inputs are 10 neurons, one per normalized bin of the sliding histogram. The network output are two neurons, one per class - good and bad. Following [14] we find the minimum number of neurons per each hidden layer that avoids over-fitting. Given the neural network number of inputs N_i and outputs N_o, we use as upper boundary for the number N_h of the hidden layers neuron, the result of : $N_h = \frac{N_s}{(\sigma * (N_i + N_o))}$ where $2 \leq \sigma \leq 10$. The activation function for the input and hidden layers neurons is a rectified linear unit (ReLU) while the output layer activation is a normalized exponential function, also known as softmax. After training, we run again SABES but with the additional filter provided by the neural network. The validation data-set is the same used for the evaluation of the heuristic alone. The results obtained are shown in Figure 6b. The boxplot shows the improved accuracy provided by the usage of the neural network. The 50th percentile of the estimations error is below 10Mbps in all the scenarios proposed, compared to the 30Mbps obtained when using the heuristic alone. While training a neural network can be computationally expensive, we quantify that adding the neural network evaluation increases the computation execution time only by 6% in respect to the heuristic alone.

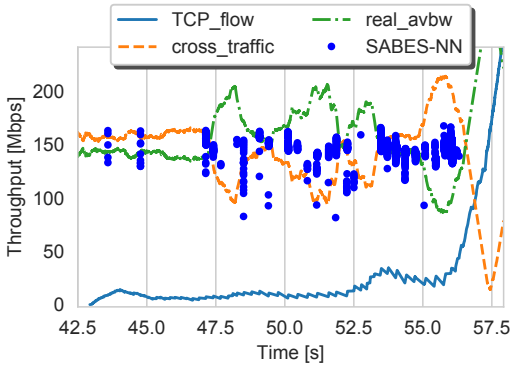


Fig. 8: SABES-NN running in a real testbed. Estimations mean absolute error is 8% of the bottleneck capacity.

VI. Evaluation in real testbed

We finally tested SABES in a real testbed. The testbed topology is a dumbell as shown in Figure 7. It is composed of six virtual machines, two for TCP traffic generation, two for UDP cross-traffic generation, and finally two acting as intermediate routers and shapers. The VMs are deployed in two different physical hosts. The two hosts are connected through a 1Gbps link, but we apply a traffic shaper of 300Mbps to the egress interfaces of both intermediate routers. We also add an artificial delay of 20ms between the two physical hosts using the Linux Traffic Control module NetEm. All VMs run Debian Linux; Host 1 is running KVM as hypervisor, while Host 2 has VMWare ESXi. TCP traffic is generated as large file transfers of 100 MB over HTTP. UDP cross-traffic is generated with the D-ITG tool [15]. We chose this tool as it allows to generate traffic following a Pareto distribution of choice as done in simulation. In Figure 8, the results of SABES-NN estimations are shown for a single test. The test consists of three concurrent HTTP/TCP file transfers of 100 MB competing for a bottleneck link of 300Mbps with one UDP cross-traffic flow. The throughput of only one of the three TCP flows is shown. SABES effectively detects moments where is possible to infer the real available bandwidth, even though the flow throughput is far lower. The mean absolute error of the $\mathcal{B}$ estimation set is less than 10% of the bottleneck capacity.

VII. Conclusions

In this study we presented SABES, a method to estimate network available bandwidth using passive measurements obtained from TCP traffic. SABES is computationally inexpensive, taking advantage of simple statistical analysis of TCP traffic. It can be deployed just on one side of the TCP connection, being it either close to the sender or the receiver.SABES implements an heuristic that detects IP_{rate} distributions whose mean value approximates the network available bandwidth. SABES model was validated in simulations and in a real-world virtualized scenario. We show that SABES heuristic provides a fair estimation of the available bandwidth of our validation data-set with a median mean absolute error of 30% of the bottleneck capacity. We then improved the heuristic by detecting patterns in the distributions that could be learned by an Artificial Neural Network. The ANN acts as discriminator of histograms that are better suited for the estimation. SABES-NN improves the estimation results over the validation set reducing the median mean absolute error down to 10%. SABES relies on a good initial estimation of the bottleneck capacity. Future work includes finding an improved technique for the estimation, taking advantage of other signal processing techniques such as Kernel Density Estimation. Finally, SABES is a good candidate for an estimator that is part of a TCP congestion control algorithm or other types of traffic control systems such as the one described in [16].

Acknowledgment

This work was supported by the grant 2015DI023 as part of the Industrial PhD grants of AGAUR and Generalitat de Catalunya. Project co-financed by the Spanish Ministry of Ciencia Innovacion y Universidades with reference RTC-2017-6655-7 (FEDER).

References

[1] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *Proceedings IEEE INFOCOM 2001*. IEEE, 2001.

[2] J. Strauss, D. Katabi, F. Kaashoek, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*.

[3] S. K. Khangura and M. Fidler, "Available bandwidth estimation from passive tcp measurements using the probe gap model," in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 2017.

[4] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Globecom'00-IEEE. Global Telecommunications Conference. Conference Record (Cat. No. 00CH37137)*. IEEE.

[5] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathchirp: Efficient available bandwidth estimation for network paths," in *Passive and active measurement workshop*, 2003.

[6] S. Ekelin, M. Nilsson, E. Hartikainen, A. Johnsson, J.-E. Mangs, B. Melander, and M. Bjorkman, "Real-time measurement of end-to-end available bandwidth using kalman filtering," in *2006 ieee/ifip network operations and management symposium noms 2006*. IEEE, 2006.

[7] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the internet," *Queue*, 2011.

[8] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," 2016.

[9] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications."

[10] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of BBR congestion control," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. IEEE, 2017.

[11] S. Ma, J. Jiang, W. Wang, and B. Li, "Fairness of congestion-based congestion control: Experimental evaluation and analysis," *arXiv preprint arXiv:1706.09115*, 2017.

[12] S. Ha and I. Rhee, "Taming the elephants: New tcp slow start," *Computer Networks*, 2011.

[13] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010.

[14] "How to choose the number of hidden layers and nodes in a feedforward neural network?" Cross Validated. [Online]. Available: https://stats.stackexchange.com/q/136542

[15] S. Avallone, S. Guadagno, D. Emma, A. Pescapè, and G. Ventre, "D-itg distributed internet traffic generator," in *First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings.* IEEE, 2004.

[16] F. Ciaccia, O. Arcas-Abella, D. Montero, I. Romero, R. Milito, R. Serral-Gracia, and M. Nemirovsky, "Improving tcp performance and reducing self-induced congestion with receive window modulation," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2019.