Neural Network-based Available Bandwidth Estimation from TCP Sender-side Measurements

Sukhpreet Kaur Khangura
Institute of Communications Technology
Leibniz Universität
Hannover, Germany
khangura.sukhpreet@ikt.uni-hannover.de

***Abstract*—In this paper, we perform the bandwidth estimation from sender-side measurements using input and acknowledgment gaps. However, difficulties arise when these packet gaps get distorted. To deal with noise-afflicted packet gaps, we propose a neural network-based method for estimating the available bandwidth. We also apply the neural network under a variety of difficult conditions that have not been included in the training. We compare the performance of our proposed method with state-of-the-art model-based technique, where our neural network approach shows improved performance.**

***Index Terms*—Passive bandwidth estimation, neural network, multi-hop networks, lossy networks.**

I. Introduction

Transmission Control Protocol (TCP), the dominant transport layer protocol, is at the center of all Internet traffic. Though initially designed to provide a connection-oriented reliable data transport using a fair share of bandwidth, nowadays, it is an underlying protocol for video streaming applications such as MPEG-DASH, the core technology used by major Internet video providers such as YouTube and Netflix [1]. DASH splits the original video inside the server into segments of certain sizes typically in the order of a few MBs called "chunks" that are trans-coded into multiple quality levels. On the client-side, these are downloaded using HTTP GET requests selecting the quality level based on the estimated bandwidth of the previous chunk(s). Furthermore, the significance of bandwidth estimation for short-lived TCP flows arises from the benefits that TCP's new versions can gain to improve their throughput, e.g., HyStart [2] seeks to estimate the available bandwidth to find a safe exit point from a slow start before packets are lost. The recently proposed TCP's Bottleneck Bandwidth and Round-trip propagation time (BBR) [3] uses an estimate of the available bandwidth to determine its sending rate.

The term available bandwidth (A) refers to the residual capacity that is left over after the cross traffic of certain rate (λ) is served by a network path of capacity C, i.e., $A = C - \lambda$. In the context of data networks, such as the Internet, an end-to-end network path refers to a sequence of successive links that connects two end hosts via intermediary network nodes, e.g., routers, switches. The capacity of an end-to-end path is determined by the link with the minimum capacity known as *bottleneck link* and the link with minimum available bandwidth known as tight link determines available bandwidth. The tight may differ from the bottleneck link.

Throughout the years, the significance of bandwidth estimation has encouraged researchers, and several available-bandwidth measurement tools based on passive [4]–[6] and active [5], [7]–[18] measurements have been developed. Passive measurement tools are based on the non-intrusive monitoring of traffic and require direct access to the point of interest, i.e., intermediary nodes in the network path. In contrast, active tools are based on injecting artificial probe traffic with the initial known structure, e.g., with well-defined input gap g_{in} and the packet length l. The dispersion that arises when packets traverse a network, carries information that can reveal relevant network characteristics. Though active probing tools provide the flexibility over the input structure of probe traffic, TCP flows exhibit a rather chaotic traffic pattern. Furthermore, for certain applications, the intrusive active probing may decrease their performance and, hence satisfaction level of the users. For example, they can affect the response time of TCP connections [19]. Furthermore, on detecting congestion due to intrusive probing, TCP adjusts its Congestion Window (CWND), which affects its performance adversely in scenarios with long Round Trip Time (RTT). In the worst scenarios, due to high-intensity probing, significant delays, or even packet losses may occur which are modeled as infinite delays in [18] producing estimation bias.

Due to the aforementioned problems of active probing, it is favorable if the available bandwidth can be estimated from passive measurements of existing network traffic. In our previous work [20], we propose a method that can infer the available bandwidth using sender-side measurements, i.e., input gaps g_{in} and acknowledgment gaps g_{ack} for short-lived TCP flows and with non-negligible Round Trip Time (RTT). However, the acknowledgment gaps get altered due to random cross traffic, multiple bottleneck links, and packet loss which increases the measurement noise resulting in an underestimation of the available bandwidth [15], [21]. In the context of the above discussion, in the present work, we try to investigate: *"Whether machine learning can be used to estimate the available bandwidth from noise affected sender-side TCP passive measurements? How to obtain reliable estimates when only a small number of samples are available due to short-lived flows? How to select relevant dimensions*

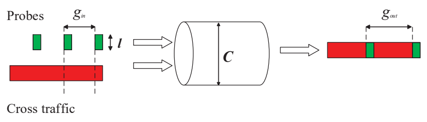


Fig. 1. FIFO multiplexer with capacity C and fluid constant rate traffic λ

of feature vectors, when the underlying density of input and acknowledgment gaps is usually unknown?". On the way to find answers to these questions, we made the following contributions: we propose a neural network-based method that works with small amount of data and can deal with distorted acknowledgment gaps to obtain reliable available bandwidth estimates using sender-side TCP passive measurements. We evaluate our method in controlled experiments in a network testbed. We specifically target topologies where the assumptions of the deterministic fluid model in Eq. 1 are not satisfied, such as bursty cross traffic, multiple tight links, and packet losses. We compare our neural network-based method with a reference implementation of a state-of-the-art model-based technique from our previous work [20].

The remainder of this paper is structured as follows: In Sec. II, we introduce the reference implementation of model-based estimation technique from our previous work [20]. We present our neural network-based method, describe the training, and show testing results in Sec. III. In Sec. IV, we consider the estimation of available bandwidth for networks with multiple tight links. The evaluation of our method for lossy networks is presented in Sec. V. In Sec. VI, we give brief conclusions.

II. Model-based Reference Implementation

A common assumption in bandwidth estimation is that the available bandwidth does not change during a probe measurement, i.e., the cross traffic is assumed to be of a constant rate λ. Furthermore, to simplify the modeling, the effects due to cross traffic packet granularity are ignored, i.e., it is assumed that cross traffic behaves like a fluid. In the fluid-flow model, a network path is modeled as a single lossless tight link behaving as a First-In-First-Out (FIFO) multiplexer of probe and cross traffic and the relation between input g_{in} and output gaps g_{out} is expressed as [15]

$$g_{\text{out}} = \max\left\{g_{\text{in}}, \frac{g_{\text{in}}\lambda + l}{C}\right\}. \tag{1}$$

The reasoning is that during g_{in} an amount of $g_{\text{in}}\lambda$ of the fluid cross traffic is inserted between any two packets of the probe traffic as shown in Fig 1. Reordering Eq. (1) gives the characteristic *gap response curve*

$$\frac{g_{\text{out}}}{g_{\text{in}}} = \begin{cases} 1 & \text{if } \frac{l}{g_{\text{in}}} \le C - \lambda, \\ \frac{l}{g_{\text{in}}C} + \frac{\lambda}{C} & \text{if } \frac{l}{g_{\text{in}}} > C - \lambda. \end{cases} \tag{2}$$

The utility of Eq. (2) is that it shows a clear bend at $A = C - \lambda$ that enables estimating the available bandwidth. The techniques using information from packet gaps are based on Probe Gap Model (PGM). We note that the quotient of packet size and gap is frequently viewed as the data rate of the probe used by the Probe Rate Model (PRM) techniques. The methods for available bandwidth estimation that are based on the fluid model of Eq. (1) essentially fall into two different categories: iterative probing and direct probing. Iterative probing techniques search for the turning point of the gap response curve by sending repeated probes at increasing rates, as long as $g_{out}/g_{in} = 1$. Increasing the probing rate further saturates the available bandwidth causing an increase in One Way Delays (OWD) that can be detected by the receiver [5], [11]. Direct probing techniques estimate the upward line segment of the gap response curve for $l/g_{in} > C - \lambda$ [7], [8], [14]. The line is determined by C and λ.

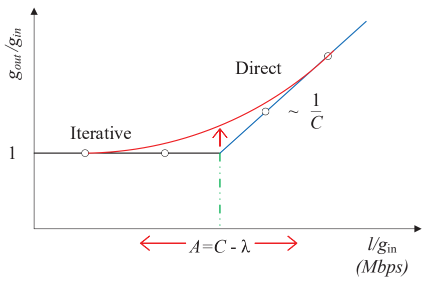


Fig. 2. Gap response curve. The bend marks the available bandwidth. The deviation from fluid-flow model results in deviation that is maximal at $l/g_{in} = C - \lambda$.

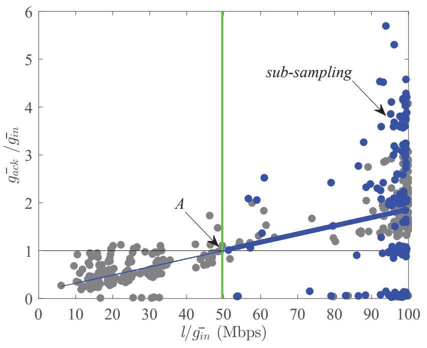


Fig. 3. Gap response curve. The ack-gap enables available bandwidth estimation using sender-side measurements only.

A. Direct Probing

Since the input rate in TCP cannot be controlled, we construct a method that uses direct probing technique together with a threshold test from DietTOPP [8] to select relevant input g_{in} and acknowledgment g_{ack} packet gaps [20]. The ack-gap response curve shows the same characteristic slope as the gap response curve with one difference: the average input gap $\bar{g}_{\text{in}}^{j,k} = g_{\text{in}}^{j,k}/(k-j-1)$ and average ack gap $\bar{g}_{\text{ack}}^{j,k} = g_{\text{ack}}^{j,k}/(k-j-1)$ for j and k acknowledged packets, take the place of g_{in} and g_{out}, respectively. However, in TCP, as typically only every other packet is acknowledged, the acknowledgment process effectively performs a sub-sampling. Furthermore, we show in [20] that combining several gaps

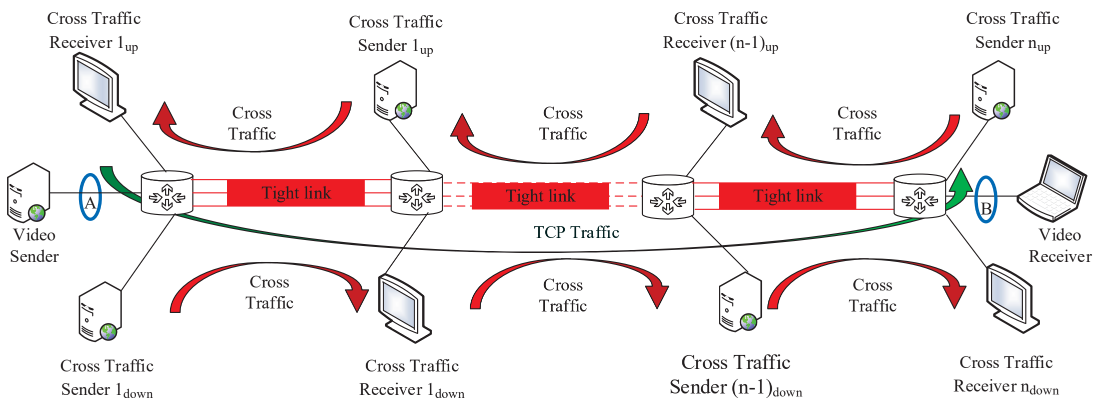

Cross Traffic Receiver 1_{up}
Cross Traffic Sender 1_{up}
Cross Traffic Receiver $(n-1)_{up}$
Cross Traffic Sender n_{up}
Cross Traffic
Cross Traffic
Cross Traffic
Video Sender
A
Tight link
Tight link
Tight link
B
Video Receiver
TCP Traffic
Cross Traffic
Cross Traffic
Cross Traffic
Cross Traffic Sender 1_{down}
Cross Traffic Receiver 1_{down}
Cross Traffic Sender $(n-1)_{down}$
Cross Traffic Receiver n_{down}

Fig. 4. Dumbbell topology set up in Emulab with multiple tight links of a capacity of 100 Mbps and a configurable delay and loss rate. Cross traffic in the downstream and upstream direction is single hop-persistent and used to load the tight links to enable controlled bandwidth estimation experiments.

to calculate $\overline{g}_{\text{in}}^{j,k}$ does not imply constant-rate packet train models. Since the individual input gaps from passive measurements are random, fewer samples pass the threshold test, i.e., $l/g_{\text{in}}^{i} > C - \lambda$ for all $i \in \{j, k-1\}$ and contribute to the regression line. Fig. 3 shows the estimate obtained from 1 MB chunk size in the presence of exponential random cross traffic of average rate $\lambda = 50$ Mbps in the tight link of capacity $C = 100$ Mbps. However, random cross traffic in the reverse path distorts the acknowledgment gaps as can be seen by $\overline{g}_{ack}/\overline{g}_{in} < 1$.

III. Neural Network-based Implementation

In this section, we present our neural network-based implementation to estimate the available bandwidth from distorted acknowledgment gaps g_{ack}. We describe the training data sets and show a comparison of available bandwidth estimates for a range of different network parameters.

A. Data-binning Implementation

We propose a neural network-based method that takes k-dimensional feature vector of the relative probability of samples of $(\overline{g}_{ack}, \overline{g}_{in})$ as input. We use a data binning approach, where we partition $l/\overline{g}_{in}$ and $\overline{g}_{ack}/\overline{g}_{in}$ into bins, with the bin edges specified by $(l/\overline{g}_{in})^{edge}$ and $(\overline{g}_{ack}/\overline{g}_{in})^{edge}$. We define bin edges as a vector of consecutive and non-overlapping intervals. The bin width is determined by the quotient of range of $l/\overline{g}_{in}$ and number of bins N_b in x-dimension, i.e., $\frac{max(l/\overline{g}_{in})-min(l/\overline{g}_{in})}{N_b}$, and similarly in y-dimension, as the quotient of range of $\overline{g}_{ack}/\overline{g}_{in}$ and N_b. Since the underlying density of input and acknowledgment gaps is usually unknown, selecting the number of bins is not a trivial task. Therefore, for our approach to avoid bins with zero samples, we choose the number of bins following Rice rule [22] as $N_b = 2N_s^{1/3}$, where N_s is the total number of samples of the gap response curve. We calculate the relative probability by counting the number of samples in each bin divided by the total number of samples N_s. To provide input to the neural network, we reshape a matrix representing the relative probabilities of samples into a k-dimensional vector.

B. Training Data: Exponential Cross Traffic, Single Tight Link

We generate different data sets for training and evaluation setting CUBIC and BBR as TCP congestion control protocols using a controlled network testbed at Leibniz Universität Hannover that is managed by the Emulab software [23]. We use a dumbbell topology with multiple tight links, as shown in Fig. 4. For short-lived TCP flows, we emulate the transmission of DASH video chunks of size of about 1 MB chosen based on the modal value of downloaded chunk sizes of an animated movie named “Sintel” by Blender Foundation from YouTube. The downstream cross traffic is used to load the bottleneck link so that a defined amount of bandwidth remains available, whereas uplink cross traffic interferes with TCP acknowledgments and alters their spacing. To transmit chunks of a defined size via TCP, we use the traffic generator iPerf [24]. Cross traffic of different types and intensities is generated using D-ITG [25]. The cross traffic packet size is set to 1514 B to consider the effect of packet granularity. The cross traffic in upstream and downstream is single hop-persistent, i.e., at each link fresh cross traffic is multiplexed. TCP traffic is path-persistent, i.e., it travels along the entire network path.

Since the PCs in our Emulab testbed are connected via physical Ethernet links of 1 Gbps and 10 Gbps, respectively, we use the token bucket filter [26] to emulate a 100 Mbps tight links. Besides, a delay node is used at the tight link to emulate a wide-area link to investigate the effect of different RTTs on the bandwidth estimation. The delay node can also be configured to create packet loss with a defined probability in the downstream and upstream direction. The access links are configured to have 100 Mbps capacity. We note that the emulation has limited accuracy and hence contributes additional noise to the measurements. We disable the segmentation offloading by the network interface card using ethtool [27]. Hence, the TCP/IP stack is responsible for segmenting chunks into datagrams of

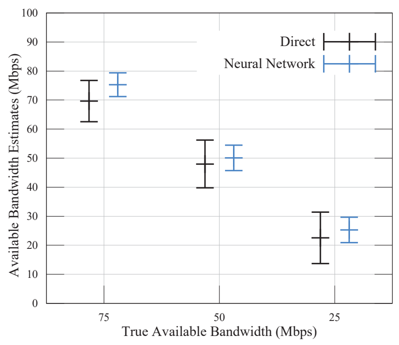


Fig. 5. Available bandwidth estimates obtained from ack-gaps for different exponential cross traffic rates that have been included in the training data set.

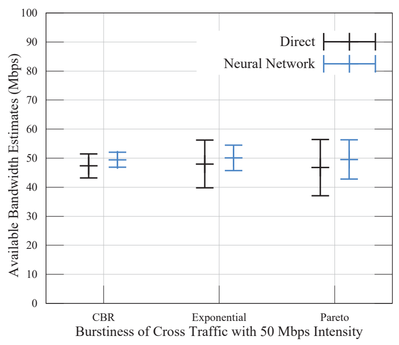


Fig. 6. Bandwidth estimates for different types of cross traffic burstiness. An increase of the burstiness causes a higher variability of the bandwidth estimates.

1500 B size, that is the maximum transmission unit carried by the Ethernet links. Including Ethernet header and trailer, the packet size is 1514 B. Packet time-stamps at the video sender and receiver are generated at points A and B, respectively, using libpcap at the hosts. We also use a specific Endace Data Acquisition and Generation (DAG) measurement card to obtain accurate reference time-stamps.

The first training data set (i) is generated for a single tight link of capacity $C = 100$ Mbps. The capacity of the access links is also configured to have $C = 100$ Mbps. The exponential cross traffic with an average rate of $\lambda = 25$, 50, and 75 Mbps is used to generate different available bandwidths. TCP short-lived flow is a chunk of 1 MB size with RTT 2 ms. For each configuration, 100 repeated experiments are performed. For training of the neural network, we first implement an autoencoder for each layer separately and then fine-tune the network using scaled conjugate gradient (scg). Given a regression network, we optimize the L2-error requiring approximately 1000 epochs until convergence is achieved. Due to the limited amount of training data (300 experiments overall in the first training data set), the shallow network with a small number of hidden neurons allows training without much over-fitting. Both the methods generate available bandwidth estimates from the same measurement data.

C. Evaluation: Exponential Cross Traffic, Single Tight Link

We train the neural network using the first training data set (i) and generate additional data sets for testing using same network configuration as the training data set (i), i.e., using exponential cross traffic of 25, 50, and 75 Mbps at a single tight link of 100 Mbps capacity. The testing results of the neural network-based method compared to the results of the direct method are summarized in Fig. 5. We show the average of 100 available bandwidth estimates with error bars depicting the standard deviation of the estimates.

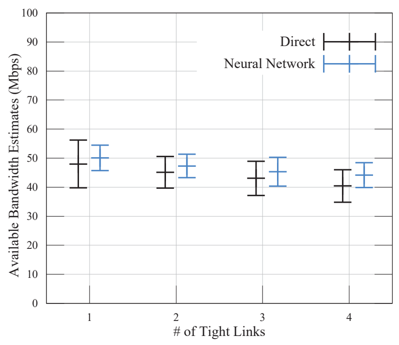
The variability of the available bandwidth estimates of the direct method is comparably large and the average underestimates the true available bandwidth. Though variability of estimates could be due to a number of reasons [28], particularly in this case, the exponential cross traffic deviates from the fluid model and causes random fluctuations of the measurements of g_{ack}. The neural network-based method improves bandwidth estimates significantly. The average matches the true available bandwidth, and the variability is low. The good performance of the neural network is not unexpected as it has been trained for the same network parameters.

D. Network Parameter Variation Beyond the Training Data

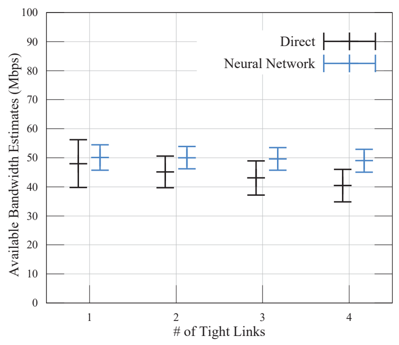
We investigate the sensitivity of the neural network with respect to a variation of network parameters that differ substantially from the training data set. Specifically, we consider two cases that are known to be hard in bandwidth estimation. These are cross traffic with high burstiness, and networks with multiple tight links. In this section, we evaluate the variability of the cross traffic. Multiple tight links are discussed in the following section.

1) Burstiness of Cross Traffic: To evaluate how the neural network-based method performs in the presence of cross traffic with an unknown burstiness, we consider three different types of cross traffic: Constant Bit Rate (CBR) that has no burstiness as assumed by the probe rate model, moderate burstiness due to exponential packet inter-arrival times, and heavy burstiness due to Pareto inter-arrival times with infinite variance, caused by a shape parameter of $\alpha = 1.5$. The average rate of cross traffic is $\lambda = 50$ Mbps in all cases. As before, the tight link and access links capacities are $C = 100$ Mbps.

Fig. 6 shows the mean and the standard deviation of 100 repeated experiments using the direct and the neural network-based method. The average of the estimates shows a slight underestimation bias for direct method due to the queueing effect caused by bursty traffic at the tight link even if the TCP flow rate is below the average available bandwidth, i.e., if $l/g_{in} < C - \lambda$. More pronounced is the effect of the cross traffic burstiness on the standard deviation of the bandwidth estimates. While for CBR cross traffic the estimates are close to deterministic, the variability of the estimates increases significantly if the cross traffic is bursty. The neural network that has been trained for exponential cross traffic only, performs almost perfectly in case of CBR cross traffic and shows good results with less variability compared to the direct technique also for the case of Pareto cross traffic.



(a) Neural network trained for a single tight link



(b) Neural network trained for multiple tight links

Fig. 7. The training of the neural network for multiple tight links improves the bandwidth estimates significantly.

IV. Multiple Tight Links

The multiple tight links networks pose a well-known challenge in available bandwidth estimation. In the case of multiple tight links, the chaotic input gaps g_{in} of TCP get further distorted in the following links as they are the output gaps of the preceding links. At each additional link, the TCP stream interacts with new, bursty cross traffic and it's output rate reduces. This results in an underestimation of the available bandwidth in multi-hop networks [15], [18], [21], [29]. To test the neural network with multiple tight links, we extend our network from single-hop to multi-hop, as shown in Fig. 4. The path-persistent TCP flow experiences single hop-persistent exponential cross traffic with average rate $\lambda = 50$ Mbps in upstream and downstream, while traversing multiple tight links of capacity $C = 100$ Mbps. The capacity of the access links is 100 Mbps.

In Fig. 7a, we show the results from 100 repeated measurements for networks with 1 up to 4 tight links. The direct as well as the neural network-based method, underestimate the available bandwidth with an increasing number of tight links. The reason for underestimation in case of model-based techniques is the cross traffic burstiness, which potentially reduces the output probe rate at each of the tight links. Though the estimates of our neural network show the least variability, the neural network underestimates the available bandwidth. This is not surprising, as it is trained only for a single tight link.

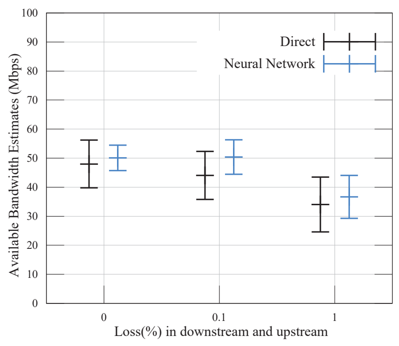
1) Training for Multiple Tight Links: Since the neural network, that is trained only for a single tight link, underestimates the available bandwidth in case of multiple tight links, we consider training the neural network for the latter. To generate the training data, a chunk of 1 MB data is transmitted via TCP through a network with two, three, and four tight links, respectively, each with single-hop persistent exponential cross traffic with an average rate $\lambda = 50$ Mbps in uplink and downlink. The capacity of the access links and that of the tight links is 100 Mbps. The neural network is trained with this additional training set (ii) for multiple tight links along with training set (i) for a single tight link.

Fig. 7b compares the results of the direct method with the neural network. As can be seen clearly, the results have improved significantly with the neural network after it has been trained for multiple tight links. The mean value matches the true available bandwidth, and the estimates have less variability.

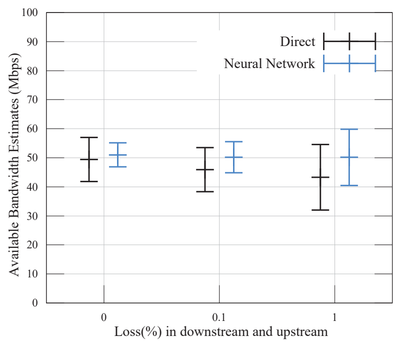
V. Evaluation of Loss

The fluid flow models that are used in available bandwidth estimation assume lossless systems and few methods for bandwidth estimation consider loss [5], [18], [20]. Though the loss of acknowledgments is less of an issue as it is typically resolved by the next cumulative acknowledgment, but it perturbs packet sequence because of retransmission and distorts the input and acknowledgment gaps further [20].

In Fig. 8a, and Fig. 8b we show the comparison results of the neural network-based and direct method for exponential cross traffic with an average rate $\lambda = 50$ Mbps using CUBIC and BBR, respectively. Loss rates of 0%, 0.1%, and 1% are evaluated. For TCP CUBIC in Fig. 8a, though the estimates of neural network-based method show less variability, both the methods underestimate the available bandwidth with increasing loss rate. This is due to the fact that TCP CUBIC is a loss-based congestion control algorithm and adjust its congestion window when packet loss occurs, which affects the distribution of input and acknowledgment gaps. Hence neural network trained for a lossless network doesn't perform well for lossy networks. In the case of the direct method with higher packet loss, few samples remain that pass the threshold test for the regression step to estimate the upward segment of the gap response curve. In contrast to traditional loss-based congestion control algorithms like CUBIC, BBR is designed to respond to actual congestion, rather than packet loss. This can be seen from the results of the direct method in Fig. 8b which are better in case of BBR as compared to CUBIC. In addition to that, though the variation in estimate increases with increase in loss rate, the neural network trained for TCP BBR lossless link can estimate the bandwidth correctly. However, for both TCP CUBIC and BBR after training for a lossy network, the results for neural network improve. Due to limited space, we have not shown these results.



(a) Bandwidth estimates for TCP CUBIC



(b) Bandwidth estimates for TCP BBR

Fig. 8. Bandwidth estimates for TCP (a) CUBIC and (b) BBR for a lossy network with a neural network trained for a lossless network.

VI. CONCLUSION

We investigated how neural networks can be used to benefit passive measurement-based available bandwidth estimation. We proposed a method that apply data binning technique on gap response curve obtained by using short-lived TCP sender-side measurements. Our method uses k-dimensional feature vector of the relative probability of samples of input and acknowledgment packet gaps as input to a neural network to estimate the available bandwidth. We conducted a comprehensive measurement study in a controlled network testbed. Our results showed that neural networks can significantly improve available bandwidth estimates by reducing bias and variability. This holds true also for network configurations that have not been included in the training data set, such as different types and intensities of cross-traffic, multiple tight links and packet loss.

REFERENCES

[1] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *ACM Conference on Multimedia Systems*, 2011, pp. 157–168.

[2] S. Ha and I. Rhee, "Taming the elephants: New TCP slow start," *Computer Networks*, vol. 55, no. 9, pp. 2092–2110, 2011.

[3] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," 2016.

[4] M. Zangrilli and B. B. Lowekamp, "Applying principles of active available bandwidth algorithms to passive tcp traces," in *International Workshop on Passive and Active Network Measurement*. Springer, 2005, pp. 333–336.

[5] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *Passive and Active Measurement Workshop*, 2002.

[6] C. L. T. Man, G. Hasegawa, and M. Murata, "A merged inline measurement method for capacity and available bandwidth," in *International Workshop on Passive and Active Network Measurement*. Springer, 2005, pp. 341–344.

[7] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *IEEE Globecom*, 2000, pp. 415–420.

[8] A. Johnsson, B. Melander, and M. Björkman, "Diettopp: A first implementation and evaluation of a simplified bandwidth measurement method," in *Swedish National Computer Networking Workshop*, 2004.

[9] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *IEEE INFOCOM*, 2001, pp. 905–914.

[10] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 537–549, 2003.

[11] 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.

[12] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *ACM Internet Measurement Conference*, 2003, pp. 39–44.

[13] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 879–894, 2003.

[14] 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 *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2006, pp. 73–84.

[15] X. Liu, K. Ravindran, and D. Loguinov, "A queueing-theoretic foundation of available bandwidth estimation: single-hop analysis," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 918–931, 2007.

[16] ——, "A stochastic foundation of available bandwidth estimation: Multi-hop analysis," *IEEE/ACM Transaction on Networking*, vol. 16, no. 1, pp. 130–143, 2008.

[17] J. Liebeherr, M. Fidler, and S. Valaee, "A system theoretic approach to bandwidth estimation," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1040–1053, 2010.

[18] R. Lübben, M. Fidler, and J. Liebeherr, "Stochastic bandwidth estimation in networks with random service," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 484–497, 2014.

[19] A. Shriram and J. Kaur, "Empirical evaluation of techniques for measuring available bandwidth," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 2162–2170.

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

[21] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls on end-to-end available bandwidth estimation," in *ACM Internet Measurement Conference*, 2004, pp. 272–277.

[22] D. M. Lane, D. Scott, M. Hebl, R. Guerra, D. Osherson, and H. Zimmer, *An Introduction to Statistics*. Citeseer, 2017.

[23] D. S. Anderson, M. Hibler, L. Stoller, T. Stack, and J. Lepreau, "Automatic online validation of network configuration in the emulab network testbed," in *IEEE International Conference on Autonomic Computing*, 2006, pp. 134–142.

[24] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf: The TCP/UDP bandwidth measurement tool," *https://iperf.fr/*, 2005.

[25] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-ITG distributed internet traffic generator," in *Quantitative Evaluation of Systems*, 2004, pp. 316–317.

[26] K. Wagner, "Short evaluation of linuxs Token-Bucket-Filter (TBF) queuing discipline," 2001.

[27] "ethtool," ://linux.die.net/man/8/ethtool, accessed: 05-01-2017.

[28] S. K. Khangura, M. Fidler, and B. Rosenhahn, "Machine learning for measurement-based bandwidth estimation," *Manuscript accepted for publication in Elsevier's Computer Communications Journal*, 2019.

[29] L. Lao, C. Dovrolis, and M. Sanadidi, "The probe gap model can underestimate the available bandwidth of multihop paths," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 29–34, 2006.