Neural Networks for Measurement-based Bandwidth Estimation

Sukhpreet Kaur Khangura, Markus Fidler, Bodo Rosenhahn
Department of Electrical Engineering and Computer Science, Leibniz Universität Hannover

***Abstract*—The dispersion that arises when packets traverse a network carries information that can reveal relevant network characteristics. Using a fluid-flow model of a bottleneck link with first-in first-out multiplexing, accepted probing tools measure the packet dispersion to estimate the available bandwidth, i.e., the residual capacity that is left over by other traffic. Difficulties arise, however, if the dispersion is distorted compared to the model, e.g., by non-fluid traffic, multiple bottlenecks, clustering of packets due to interrupt coalescing, and inaccurate time-stamping in general. It is recognized that modeling these effects is cumbersome if not intractable. This motivates us to explore the use of machine learning in bandwidth estimation. We train a neural network using vectors of the packet dispersion that is characteristic of the available bandwidth. Our testing results reveal that even a shallow neural network identifies the available bandwidth with high precision. We also apply the neural network under a variety of notoriously difficult conditions that have not been included in the training, such as heavy traffic burstiness, and multiple bottleneck links. Compared to two state-of-the-art model-based techniques, the neural network approach shows improved performance. Further, the neural network can effectively control the estimation procedure in an iterative implementation.**

I. Introduction

The term *available bandwidth* refers to the residual capacity of a link or a network path that is left over after the existing traffic, also referred to as *cross traffic*, is served. Knowledge of the available bandwidth benefits rate-adaptive applications and facilitates, e.g., network monitoring, detection of congested links, and load balancing. The goal of bandwidth estimation is to infer the available bandwidth of a network path using external observations of data packets, only.

Formally, given a link with capacity C and cross traffic with long-term average rate λ, where $\lambda \in [0, C]$, the available bandwidth $A \in [0, C]$ is defined as $A = C - \lambda$ [1]. The end-to-end available bandwidth of a network path is determined by its *tight link*, that is the link that has the minimal available bandwidth [2]. The tight link may differ from the *bottleneck link*, i.e., the link with the minimal capacity.

To date, a number of accepted active probing techniques and corresponding theories for available bandwidth estimation exist, e.g., [1]–[13]. These techniques use a sender that actively injects artificial *probe traffic* with a defined packet size l and inter packet gap referred to as input gap g_{in} into the network. At the receiver, the output gap of the received probe g_{out} is measured to deduce the available bandwidth.

ISBN 978-3-903176-08-9

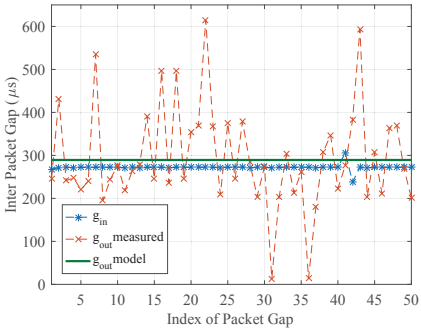


Fig. 1. Measurements of g_{in} and g_{out} compared to the fluid model. The network has a single tight link with capacity $C = 100$ Mbps and exponential cross traffic with rate $\lambda = 62.5$ Mbps. The packet size is $l = 1514$ byte.

A common assumption in bandwidth estimation is that the available bandwidth, respectively, the rate of the cross traffic does not change during a probe. Further, to simplify modeling, cross traffic is assumed to behave like fluid, i.e., effects that are due to the packet granularity of the cross traffic are neglected. Modeling a single tight link as a lossless First-In First-Out (FIFO) multiplexer of probe and cross traffic, the relation of g_{out} and g_{in} follows by an intuitive argument [1] as

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

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, so that the probe packets may be spaced further apart. Reordering Eq. (1) gives the characteristic *gap response curve*

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

The utility of Eq. (2) is that it shows a clear bend at $A = C - \lambda$ that enables estimating the available bandwidth using different techniques, see Sec. II. We note that the quotient of packet size and gap is frequently viewed as the data rate of the probe.

For an example, consider a tight link with capacity $C = 100$ Mbps and cross traffic with average rate $\lambda = 62.5$ Mbps. The packet size is $l = 1514$ byte, resulting in a transmission time l/C of about 120 μs. Given an input gap $g_{\text{in}} = 270$ μs, the output gap follows from Eq. (1) as $g_{\text{out}} = 290$ μs. Now, assume for the moment that C is known but λ is unknown. An active probing tool can send probes with, e.g., $g_{\text{in}} = 270$ μs to measure g_{out}. Noting that $g_{\text{out}}/g_{\text{in}} > 1$, Eq. (2) reveals the unknown $\lambda = (g_{\text{out}}C - l)/g_{\text{in}} = 62.5$ Mbps.

In practice, the observations of g_{out} are distorted for various reasons. For an example, we recorded a measurement trace of 50 pairs of g_{in} and g_{out} in the network testbed in Fig. 3 with a single tight link and the above parameters C, λ, and l, where l is the maximal size of Ethernet packets including the header. The results are shown in Fig. 1. Neglecting the cases where $g_{\mathrm{out}} < g_{\mathrm{in}}$ that are not possible in the model and ignoring large outliers, a range of samples g_{out} of about 360 μs remain that suggest concluding $\lambda \approx 90$ instead of 62.5 Mbps erroneously.

A. Challenges in Bandwidth Estimation

Relevant reasons for the distortions of g_{out} include deviations from the assumptions of the model, i.e., a lossless FIFO multiplexer with constant, fluid cross traffic as well as measurement inaccuracies, such as imprecise time-stamping:

Random cross traffic: Eq. (2) is deterministic and hence it does not define how to deal with the randomness of g_{out} that is caused by variable bit rate cross traffic. It is shown in [1] that the problem cannot be easily fixed by using the expected value $\mathsf{E}[g_{\mathrm{out}}]$ instead. In brief, this is due to the non-linearity of Eq. (2) and the fact that the location of the turning point $C - \lambda$ fluctuates if the rate of the cross traffic λ is variable. The result is a deviation that is maximal at $l/g_{\mathrm{in}} = C - \lambda$ and causes underestimation of the available bandwidth [1], [14].

Packet interference: Non-conformance with the fluid model arises due to the interaction of probes with packets of the cross traffic. In Fig. 1, two relevant examples are identified by frequent samples of g_{out} in the range of 240 and 360 μs, respectively. In contrast, the g_{out} of 290 μs, that is predicted by the fluid model, is observed rarely. To understand this effect, consider two probe packets with $g_{\mathrm{in}} = 270$ μs and note that the transmission time of a packet is 120 μs. The case $g_{\mathrm{out}} = 360$ μs occurs if two cross traffic packets are inserted between the two probe packets. Instead, if one of the two cross traffic packets is inserted in front of the probe, it delays the first probe packet, resulting in $g_{\mathrm{out}} = 240$ μs.

Packet loss: If probe packets are lost, the corresponding output gaps are void. This causes estimation bias, since packets that encounter congestion have a higher loss probability. There are few bandwidth estimation tools that consider loss, e.g., as an indication that the available bandwidth is exceeded [6].

Multiple tight links: An extensions of Eq. (1) for multiple links is derived in [3]. Yet, if cross traffic is non-fluid, the repeated packet interaction at each of the links distorts the probe gaps. Further, in case of random cross traffic, there may not be a single tight link, but the tight link may vary randomly. The consequence is an underestimation of the available bandwidth [14], [15] that is analyzed in [11], [13].

Measurement inaccuracies: Besides, there exist limitations of the accuracy due to the hardware of the hosts where measurements are taken. A possible clock offset between sender and receiver is dealt with by the use of probe gaps. A problem in high-speed networks is, however, interrupt coalescing [16], [17]. This technique avoids flooding a host with too many interrupts by grouping packets received in a short time together in a single interrupt, which distorts g_{out}.

B. State-of-the-Art Estimation Techniques

To alleviate the observed variability of the samples of g_{out}, state-of-the-art bandwidth estimation methods perform averaging of several g_{out} samples. These samples can be collected by repeated probes of two packets, so-called *packet pairs* [18], or by *packet trains* [5], [19] that consist of n consecutive packets and hence comprise $n - 1$ gaps. Further, to improve the available bandwidth estimates, statistical post-processing techniques are used, such as Kalman filtering [10], [20], majority decisions [6], averaging of the bandwidth estimates of repeated experiments [7], [8], or linear regression [4].

These techniques do, however, not overcome the basic assumptions of the deterministic fluid model in Eq. (1). While packet trains and statistical postprocessing help reduce the variability of available bandwidth estimates, they cannot resolve systematic deviations, such as the underestimation bias in case of random cross traffic and multiple tight links [1], [11], [13]. Further, it is difficult to tailor methods to specific hardware implementations that influence the measurement accuracy.

These fundamental limitations motivate us to explore the use of machine learning in available bandwidth estimation. The machine learning approach has been considered early in [21], [22] and receives increasing attention in the recent research [17], [23]. The works differ from each other with respect to their application: [21] considers the prediction of the available bandwidth from packet data traces that have been obtained in passive measurements. In contrast [17], [22], [23] use active probes to estimate the available bandwidth in NS-2 simulations [22], ultra-high speed 10 Gbps networks [17], and operational LTE networks [23], respectively.

Common to these active probing methods [17], [22], [23] is the use of packet chirps [7] that are probes of several packets sent at an increasing data rate. The rate increase is achieved either by a geometric reduction of the input gap [22], by concatenating several packet trains with increasing rates to a multi-rate probe [17], or by a linear increase of the packet size [23]. Chirps permit detecting the turning point of Eq. (2), that coincides with the available bandwidth, using a single probe. They are, however, susceptible to random fluctuations [12].

Other than chirps, [22] evaluates packet bursts that are probes of back-to-back packets and concludes that bursts are not adequate to estimate the available bandwidth. Also, [17] considers constant rate packet trains for an iterative search for the available bandwidth. Here, machine learning solves a classification problem to estimate whether the rate of a packet train exceeds the available bandwidth or not. Depending on the result, the rate of the next packet train is reduced or increased in a binary search as in [6] until the probe rate approaches the available bandwidth. The authors of [17] give, however, preference to chirp probes.

The feature vectors that are used for machine learning are generally measurements of g_{out} [22], [23] with the exception that [17] uses the Fourier transform of vectors of g_{in} and g_{out}.

Supervised learning is used and [17], [23] take advantage of today's availability of different software packages to compare the utility of state-of-the-art machine learning techniques in bandwidth estimation.

C. Contributions

In this work, we investigate how to benefit from machine learning, specifically neural networks, when using standard packet train probes for available bandwidth estimation. Compared to packet chirps, that are favored in the related works [17], [22], [23], packet trains have been reported to be more robust to random fluctuations. In fact, the implementation of a chirp as a multi-rate probe, that concatenates several packet trains with increasing rates, also benefits from this [17].

Different from multi-rate probes, packet trains are typically used in an iterative procedure that takes advantage of feedback to adapt the rate of the next packet train. Such a procedure is also proposed in [17], where machine learning is used to classify individual packet trains to control a binary search. The goal is to adapt the probe rate until it approaches the available bandwidth. In contrast, we use a feature vector that iteratively includes each additional packet train probe. This additional information enables estimating the available bandwidth directly, without the necessity that the probe rate converges to the available bandwidth. Instead of a binary search, our method chooses the probe rate next, that is expected to improve the bandwidth estimate most.

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, and multiple tight links. For a reference, we implement two state-of-the-art model-based methods to use the same data set as our neural network-based approach.

The remainder of this paper is structured as follows. In Sec. II, we introduce the reference implementation of model-based estimation techniques. 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 and capacity for different tight link capacities. Our iterative neural network-based method that selects the probe rates itself is presented in Sec. V. In Sec. VI, we give brief conclusions.

II. Model-based Reference Implementations

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*. For each of the two categories, we implement a bandwidth estimation technique that is representative of the state-of-the-art. While available bandwidth estimation tools differ significantly regarding the selection and the amount of probe traffic, our implementations are tailored to use the same database so that they provide a reference for the neural network-based method.

To reduce the variability of the measurements, a common approach is the use of constant rate packet train probes. A packet train of n packets comprises $n-1$ gaps. At the receiver, the gaps are defined as $g_{\mathrm{out}}(j) = t_{\mathrm{out}}(j+1) - t_{\mathrm{out}}(j)$ for $j = 1 \ldots n-1$, where $t_{\mathrm{out}}(j)$ is the receive time-stamp of packet j. Considering the output rate of a packet train defined as

$$r_{\mathrm{out}} = \frac{(n-1)l}{t_{\mathrm{out}}(n) - t_{\mathrm{out}}(1)} \tag{3}$$

implies averaging of the output gaps since by definition of $g_{\mathrm{out}}(j)$, Eq. (3) can be rewritten as

$$r_{\mathrm{out}} = \frac{l}{\frac{1}{n-1}\sum_{j=1}^{n-1} g_{\mathrm{out}}(j)}.$$

In case of long packet trains, stationarity, and ergodicity, the denominator converges to the mean $\overline{g_{\mathrm{out}}}$. Further, for the deterministic fluid model, $g_{\mathrm{out}}(j) = g_{\mathrm{out}}$ for $j = 1 \ldots n-1$ so that $r_{\mathrm{out}} = l/g_{\mathrm{out}}$. Similarly, the input rate for a defined g_{in} is $r_{\mathrm{in}} = l/g_{\mathrm{in}}$ and by insertion into Eq. (2) the equivalent *rate response curve* of the fluid model is obtained as

$$\frac{r_{\mathrm{in}}}{r_{\mathrm{out}}} = \begin{cases} 1 & \text{if } r_{\mathrm{in}} \le C - \lambda, \\ \frac{r_{\mathrm{in}}+\lambda}{C} & \text{if } r_{\mathrm{in}} > C - \lambda. \end{cases} \tag{4}$$

The characteristic shape of Eq. (4) is shown in Fig. 2.

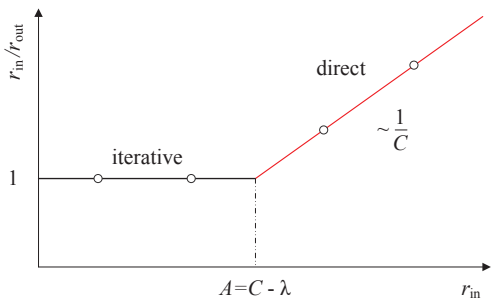


Fig. 2. Rate response curve. The turning point marks the available bandwidth.

A. Iterative probing

In brief, iterative probing techniques search for the turning point of the rate response curve by sending repeated probes at increasing rates, as long as $r_{\mathrm{in}} = r_{\mathrm{out}}$. When r_{in} reaches $C - \lambda$, the available bandwidth is saturated and increasing the probe rate r_{in} further results in self-induced congestion, so that $r_{\mathrm{in}} > r_{\mathrm{out}}$. This implies queueing at the tight link and hence increasing one way delays can be observed at the receiver.

Established iterative probing tools are, e.g., Pathload [6] and IGI/PTR [9]. Pathload adaptively varies the rates of successive packet trains r_{in} in a binary search until r_{in} converges to the available bandwidth. It uses feedback from the receiver that reports whether r_{in} exceeds the available bandwidth or not. The decision is made based on two statistical tests that detect increasing trends of the one way delay. For comparison, IGI/PTR tests whether $(r_{\mathrm{in}} - r_{\mathrm{out}})/r_{\mathrm{in}} > \Delta^{th}$, where the threshold value Δ^{th} is set to 0.1, to detect whether the probe rate exceeds the available bandwidth. Regarding the variability of the available bandwidth, Pathload reports an

available bandwidth range that is determined by the largest probe rate that did not cause self-induced congestion and the smallest rate that did cause congestion, respectively.

In our experiments, we use a dataset of equidistantly spaced r_{in} and corresponding r_{out}. We process these entries iteratively in increasing order of r_{in} and apply the threshold test of IGI/PTR [9] $(r_{\mathrm{in}} - r_{\mathrm{out}})/r_{\mathrm{in}} > \Delta^{th}$ to determine whether r_{in} exceeds the available bandwidth. We denote r_{in}^{th} the largest rate before the test detects that the available bandwidth is exceeded for the first time and report r_{in}^{th} as the available bandwidth estimate. We note that there may, however, exist $r_{\mathrm{in}} > r_{\mathrm{in}}^{th}$, where the test fails again. This may occur, for example, due to the burstiness of the cross traffic that causes fluctuations of the available bandwidth.

B. Direct probing

Instead of searching for the turning point of the rate response curve, direct probing techniques seek to estimate the parameters of the upward line segment for $r_{\mathrm{in}} > C - \lambda$. The line is determined by C and λ. If C is known, a single probe $r_{\mathrm{in}} = C$ yields a measurement of r_{out} that is sufficient to estimate $\lambda = C(C/r_{\mathrm{out}} - 1)$ from Eq. (4). Spruce [8] implements this approach. If C is also unknown, a minimum of two different probe rates $r_{\mathrm{in}} > C - \lambda$ are needed to estimate the two unknown parameters of the upward line segment of the rate response curve. This approach is taken, e.g., by TOPP [3], DietTOPP [4], and BART [10].

To implement the direct probing technique, we combine it with a threshold test to select relevant probe rates. Direct probing techniques require that $r_{\mathrm{in}} > C - \lambda$ where C and λ are unknown. We adapt a criterion from DietTOPP [4] to determine a minimum threshold $r_{\mathrm{in}}^{\min}$ that satisfies $r_{\mathrm{in}}^{\min} > C - \lambda$ and use only the probe rates $r_{\mathrm{in}} \geq r_{\mathrm{in}}^{\min}$. We use the maximal input rate in the measurement data denoted by $r_{\mathrm{in}}^{\max}$ and extract the corresponding output rate $r_{\mathrm{out}}^{\max}$. If $r_{\mathrm{in}}^{\max} > r_{\mathrm{out}}^{\max}$, it can be seen from Eq. (4) that both $r_{\mathrm{in}}^{\max} > C - \lambda$ as well as $r_{\mathrm{out}}^{\max} > C - \lambda$. Hence, we use $r_{\mathrm{in}}^{\min} = r_{\mathrm{out}}^{\max}$ as a threshold to filter out all $r_{\mathrm{in}} \leq r_{\mathrm{in}}^{\min}$. Once we have selected samples that certainly fulfill $r_{\mathrm{in}} > C - \lambda$, we use linear regression like [3], [4] to determine the upward segment of the rate response curve. The available bandwidth estimate is determined from Eq. (4) as the x-axis intercept where the regression line intersects with the horizontal line at 1, see Fig. 2.

If the assumptions of the fluid model do not hold, e.g., in case of random cross-traffic, the regression technique may occasionally fail. We filter out bandwidth estimates that can be classified as infeasible. This is the case if the slope of the regression line is so small that the intersection with 1 is on the negative r_{in} axis, implying the contradiction $A < 0$, or if the slope of the regression line is negative, implying $C < 0$.

III. Neural Network-based Method

In this section, we present our neural network-based implementation of bandwidth estimation, describe the training data sets, and show a comparison of available bandwidth estimates for a range of different network parameters.

A. Scale-invariant Implementation

We use a neural network that takes a k-dimensional vector of values $r_{\mathrm{in}}/r_{\mathrm{out}}$ as input. The corresponding r_{in} are equidistantly spaced with an increment δ_r. Hence, r_{in} is in $[\delta_r, 2\delta_r, \ldots, k\delta_r]$ that is fully defined by the parameters k and δ_r that determine the measurement resolution. Since the actual values of r_{in} do not provide additional information, they are not input to the neural network. Instead, the neural network refers to values of $r_{\mathrm{in}}/r_{\mathrm{out}}$ only by their index $i \in [1, k]$. The output of the neural network is the tuple of bottleneck capacity and available bandwidth that are also normalized with respect to δ_r, i.e., we use C/δ_r and A/δ_r, respectively. While C/δ_r and A/δ_r are not necessarily integer, they can be thought of as the index i_C and i_A where r_{in} saturates the bottleneck capacity or the available bandwidth, respectively. To obtain the actual capacity and the available bandwidth, the output of the neural network has to be multiplied by δ_r.

The normalization by δ_r achieves a neural network that is scale-invariant, since the division by δ_r replaces the units, e.g., Mbps or Gbps, by indices. Considering the fluid model in Eq. (4), the normalization of all quantities r_{in}, r_{out}, C, and λ by δ_r results in

$$\frac{r_{\mathrm{in}}}{r_{\mathrm{out}}} = \begin{cases} 1 & \text{if } i \leq i_C - i_\lambda, \\ \frac{i + i_\lambda}{i_C} & \text{if } i > i_C - i_\lambda. \end{cases} \tag{5}$$

where we used the indices $i = r_{\mathrm{in}}/\delta_r$, $i_C = C/\delta_r$, $i_\lambda = \lambda/\delta_r$, and $i_A = A/\delta_r = i_C - i_\lambda$. Eq. (5) confirms that the shape of $r_{\mathrm{in}}/r_{\mathrm{out}}$ is independent of the scale, e.g., sampling a 100 Mbps network in increments of $\delta_r = 10$ Mbps or a 1 Gbps network in increments of $\delta_r = 0.1$ Gbps reveals the same characteristic shape. The advantage of the scale-invariant representation is that the neural network requires less additional training. We note that the identity is derived under the assumptions of the fluid model and does not consider effects that are not scale-invariant such as the impact of the packet size or interrupt coalescing.

For implementation we use a $k = 20$-dimensional input vector of equidistantly sampled values of $r_{\mathrm{in}}/r_{\mathrm{out}}$. We decided for a shallow neural network consisting of one hidden layer with 40 neurons. Thus, the network comprises a 20-dimensional input vector, 40 hidden neurons and two output neurons. The output neutrons encode C/δ_r and A/δ_r.

We also explored the use of deeper networks with more hidden layers, convlayers and residual networks. However, in our setting different variants of networks did not improve the quality in our experiments. We belief, that the main reason is overfitting which is caused from the sparse amount of data used for training. When using more data or more complex settings these variants might become interesting again. Methods based on metric [24] or incremental learning [25] will also be explored in future works.

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

We generate different data sets for training and for evaluation using a controlled network testbed. The testbed is

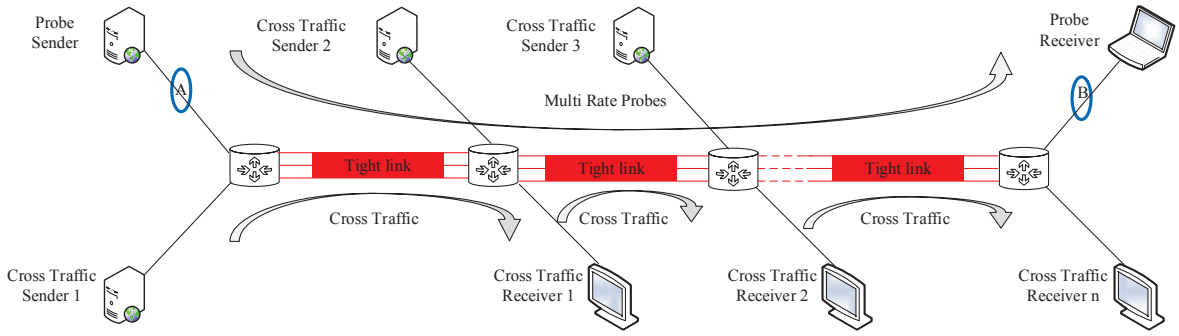


Fig. 3. Dumbbell topology set up using the Emulab and MoonGen software. A varying number of tight links with single hop-persistent cross traffic are configured. Probe-traffic is path-persistent to estimate the end-to-end available bandwidth from measurements at points A and B.

located at Leibniz Universität Hannover and comprises about 80 machines that are each connected by a minimum of 4 Ethernet links of 1 Gbps and 10 Gbps capacity via VLAN switches. The testbed is managed by the Emulab software [26] that configures the machines as hosts and routers and connects them using VLANs to implement the desired topology. We use a dumbbell topology with multiple tight links as shown in Fig. 3. To emulate the characteristics of the links, such as capacity, delay, and packet loss, additional machines are employed by Emulab. We use the MoonGen software [27] for emulation of link capacities that differ from the native Ethernet capacity. To achieve an accurate spacing of packets that matches the emulated capacity, MoonGen fills the gaps between packets by dummy frames that are discarded at the output of the link. We use the forward rate Lua script for the MoonGen API to achieve the desired forwarding rate for the transmission and reception ports of MoonGen.

Cross traffic of different types and intensities is generated using D-ITG [28]. The cross traffic is *single hop-persistent*, i.e., at each link fresh cross traffic is multiplexed. The probe traffic is *path-persistent*, i.e., it travels along the entire network path, to estimate the end-to-end available bandwidth. We use RUDE & CRUDE [29] to generate UDP probe streams. A probe stream consists of a series of k packet trains of n packets each. The k packet trains correspond to k different probe rates with a constant rate increment of δ_r between successive trains. The packet size of the probe traffic and the cross traffic is $l = 1514$ byte including the Ethernet header.

Packet timestamps at the probe sender and receiver are generated at points A and B, respectively, using libpcap at the hosts. We also use a specific endace DAG measurement card to obtain accurate reference timestamps. The timestamps are used to compute r_{in} and r_{out} for each packet train.

We generate two training data sets for a single tight link with exponential cross traffic. In data set (i) the capacity of the tight link and the access links is $C = 100$ Mbps. Exponential cross traffic with an average rate of $\lambda = 25$, 50, and 75 Mbps is used to generate different available bandwidths. In data set (ii) the capacity of the tight link is set to $C = 50$ Mbps and the exponential cross traffic has an average rate of $\lambda = 12.5$, 25, and 37.5 Mbps, respectively. In both cases the probe streams comprise packet trains of $n = 100$ packets sent at $k = 20$ different rates with rate increment $\delta_r = 5$ Mbps. For each configuration 100 repeated experiments are performed.

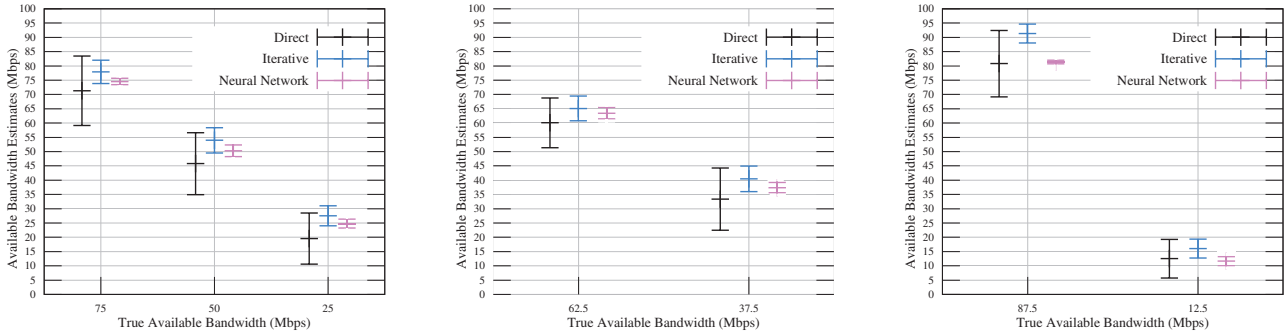
For training of the neural network, we first implement an autoencoder for each layer separately and then fine-tune the network using scaled conjugate gradient (scg). Given a regression network, we optimize the L2-error requiring approximately 1000 epochs until convergence is achieved. Training of the network (using Matlab) takes approximately 30 seconds. Due to the limited amount of training data (600 experiments overall in both training data sets), the shallow network with a small amount of hidden neurons allows training without much overfitting.

C. Evaluation: Exponential Cross Traffic, Single Tight Link

We train the neural network using the two training data sets and generate additional data sets for testing. The test data is generated for the 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. We also consider other cross traffic rates of 12.5, 37.5, 62.5, and 87.5 Mbps that have not been included in the training data set (i) to see how well the neural network interpolates and extrapolates. We repeat each experiment 100 times so that we obtain 100 bandwidth estimates for each configuration. We compare the performance of the neural network-based method with the two model-based reference implementations of an iterative and a direct estimation method. All three methods generate available bandwidth estimates from the same measurement data.

1) Testing: The testing results of the neural network-based method are summarized in Fig. 4(a) compared to the results of the direct and the iterative method. We show the average of the available bandwidth estimates with error bars that depict the standard deviation of the estimates. The variability of the available bandwidth estimates is due to a number of reasons as discussed in Sec. I-A. Particularly, the exponential cross traffic deviates from the fluid model and causes random fluctuations of the measurements of r_{out}.

The variability of the available bandwidth estimates of the direct method is comparably large and the average under-



(a) Testing (b) Interpolation (c) Extrapolation

Fig. 4. Bandwidth estimates for different cross traffic rates that have been included in the training data set (testing), that fall into the range of the training data set (interpolation), and that fall outside the range of the training data set (extrapolation). The neural network-based method provides available bandwidth estimates that exhibit little variation and have an average that matches the true available bandwidth.

estimates the true available bandwidth. The iterative method shows less variability but tends to overestimate the available bandwidth. This is a consequence of the threshold test, where a lower threshold increases the responsiveness of the test but makes it more sensitive to random fluctuations. The neural network-based method improves the 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.

2) Interpolation: Next, we consider cross traffic of the same type, i.e., exponential, however, with a different rate that has not been included in the training data. First, we consider cross traffic rates of 37.5 and 62.5 Mbps that fall into the range of rates 25, 50, and 75 Mbps that have been used for training, hence the neural network has to interpolate. The results in Fig. 4(b) show that the available bandwidth estimates of the neural network-based method are consistent also in this case.

3) Extrapolation: Fig. 4(c) depicts available bandwidth estimates for cross traffic rates of 12.5 and 87.5 Mbps. These rates fall outside the range of rates that have been included in the training data set so that the neural network has to extrapolate. The results of the neural network-based method are nevertheless highly accurate, with a noticeable underestimation of 5 Mbps on average only in case of a true available bandwidth of 87.5 Mbps. A reason for the lower accuracy that is observed when the available bandwidth approaches the capacity is that fewer measurements are on the characteristic upward line segment, see Fig. 2 that is also used for estimation by the direct method.

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 investigate two cases that are known to be hard in bandwidth estimation. These are cross traffic with high burstiness, and networks with multiple tight links.

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 the cross traffic is $\lambda = 50$ Mbps in all cases. As before, the tight link capacity and the access links capacities are $C = 100$ Mbps.

The burstiness of the cross traffic can cause queueing at the tight link even if the probe rate is below the average available bandwidth, i.e., if $r_{in} < C - \lambda$. This effect is not captured by the fluid model. It causes a deviation from the ideal rate response curve as depicted in Fig. 2 that is maximal at $C - \lambda$ and blurs the bend that marks the available bandwidth. The result is an increase of the variability of available bandwidth estimates as well as an underestimation bias in both direct and iterative bandwidth estimation techniques [1], [14].

Fig. 5 shows the mean and the standard deviation of 100 repeated experiments using the direct and iterative probing techniques and the neural network-based method. The average of the estimates shows a slight underestimation bias compared to the true available bandwidth if the cross traffic burstiness is increased. 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 and iterative techniques also for the case of Pareto cross traffic.

2) Multiple Tight Links: To test the neural network with multiple tight links, we extend our network from single-hop to multi-hop as shown in Fig. 3. The path-persistent probe streams experience single hop-persistent exponential cross-traffic with average rate $\lambda = 50$ Mbps while traversing

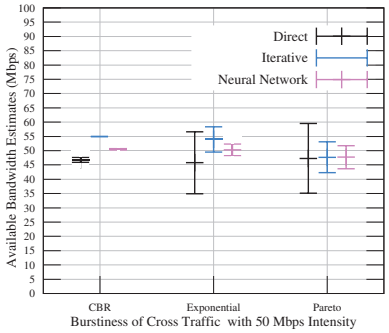


Fig. 5. Bandwidth estimates for different types of cross traffic burstiness. An increase of the burstiness causes a higher variability of the bandwidth estimates as well as an underestimation bias.

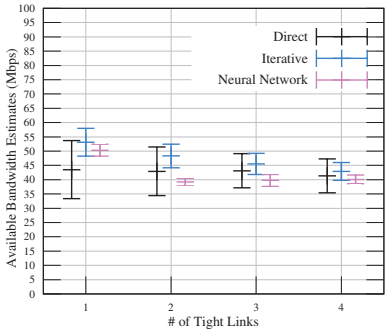


Fig. 6. Multiple tight links with capacity $C = 100$ Mbps in the presence of single hop-persistent exponential cross traffic with an average rate $\lambda = 50$ Mbps. All methods tend to underestimate the available bandwidth in case of multiple tight links.

multiple tight links of capacity $C = 100$ Mbps. The capacity of the access links is 1 Gbps.

In case of multiple tight links, the probe stream has a constant rate r_{in} with a defined input gap g_{in} only at the first link. For the following links, the input gaps have a random structure as they are the output gaps from the preceding links. At each additional link the probe stream interacts with new, bursty cross traffic. This causes lower probe output rates and results in underestimation of the available bandwidth in multi-hop networks [1], [13], [14].

In Fig. 6 we show the results from 100 repeated measurements for networks with 1 up to 4 tight links. The model-based methods, direct and iterative, as well as the neural network-based method underestimate the available bandwidth with increasing number of tight links. The reason is that the model as well as the training of the neural network consider only a single tight link. Training the neural network for multiple tight links is an interesting topic for future research. The estimates of the neural network show the least variability.

IV. Variation of the Tight Link Capacity

So far, we used test data sets that cover a tight link capacity of $C = 100$ Mbps sampled with equidistantly spaced probe rates r_{in} with an increment of $\delta_r = 5$ Mbps. Since our implementation of the neural network-based method is scale-invariant (within the limits of the fluid model), we expect that the method can perform bandwidth estimation also, e.g., in case of a tight link with $C = 50$ Mbps sampled at increments of $\delta_r = 2.5$ Mbps. If the capacity is, however, unknown, the increment δ_r cannot be adequately scaled and the measurement data will differ fundamentally. For this reason we include the training data set (ii) that is obtained for a single tight link with $C = 50$ Mbps sampled at increments of $\delta_r = 5$ Mbps. We test the neural network with data sets for $C = 50$, 100, and 200 Mbps.

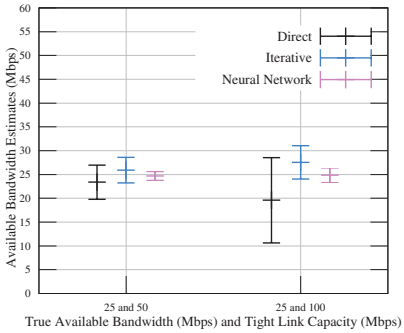


Fig. 7. Available bandwidth estimates for tight links with different capacities of $C = 50$ and 100 Mbps, respectively, and $A = 25$ Mbps available bandwidth.

A. Estimation of Available Bandwidth and Capacity

We perform testing using measurement data obtained for probe rates r_{in} with increments of $\delta_r = 5$ Mbps. The network has a single tight link with unknown available bandwidth A and unknown capacity C. In the evaluation we consider $C = 50$ and 100 Mbps and exponential cross trafic with rates $\lambda = 0.25\,C$, $0.5\,C$, and $0.75\,C$, respectively. We use the neural network to estimate both A and C.

In Fig. 7 we compare the available bandwidth estimates for $A = C - \lambda = 25$ Mbps. The results confirm that the neural network estimates the available bandwidth correctly, regardless of the capacity of the tight link. We omit further results for reasons of space and note that the neural network also estimates the capacity, i.e., 50 or 100 Mbps, with little error.

B. Capacity and Parameter Scaling

Next, we consider a proportional scaling of the network and probing parameters. In detail, the network has a single tight link with capacity $C = 50$, 100, or 200 Mbps with exponential cross traffic with rate $\lambda = 0.25\,C$, $0.5\,C$, or $0.75\,C$. The probing is performed at rate increments of $\delta_r = 2.5$, 5, and 10 Mbps, respectively. We note that only the case $C = 100$ Mbps and $\delta_r = 5$ Mbps is included in the training data set whereas the others are not. The available bandwidth estimates for $\lambda = 0.5\,C$ are presented in Fig. 8. The results confirm the utility of the scale-invariant implementation of the neural network-based method that achieves precise estimates

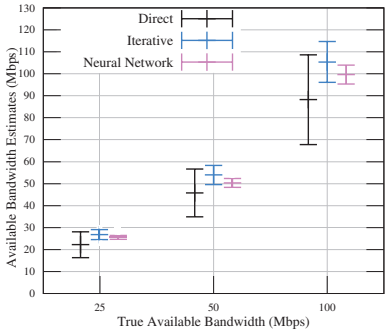


Fig. 8. Parameter scaling. Available bandwidth estimates for tight links with different capacities of $C = 50$, 100, and 200 Mbps, respectively, and $A = 0.5\,C$ available bandwidth.

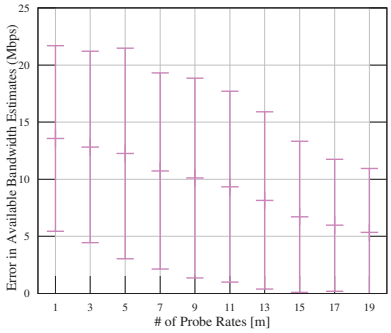


Fig. 9. Error of the available bandwidth estimates obtained for a set of m randomly selected probe rates.

in all cases. We omit showing results of the capacity estimation that was generally successful with little estimation error.

V. Iterative Neural Network-based Method

State-of-the-art iterative probing methods perform a search for the available bandwidth by varying the probe rate r_{in} until r_{in} converges to the available bandwidth. Pathload [6] uses statistical tests to determine whether r_{in} exceeds the available bandwidth or not and performs a binary search to adapt r_{in} iteratively. The recent method [17] adopts Pathload's binary search algorithm but uses machine learning instead of statistical tests to determine whether r_{in} exceeds the available bandwidth or not.

We propose an iterative neural network-based method that differs from [17] in several respects. Most importantly our method (a) determines the next probe rate by a neural network, that is trained to select the probe rate that improves the bandwidth estimate most, instead of using the binary search algorithm, and (b) it includes the information of all previous probe rates to estimate the available bandwidth instead of considering only the current probe rate. Our implementation comprises two parts. First, we train the neural network to cope with input vectors that are not fully populated. Second, we create another neural network that recommends the most beneficial probe rates.

A. Partly Populated Input Vectors

An iterative method will only use a limited set of probe rates. Correspondingly, we mark the entries of the input vector that have not been measured as invalid by setting $r_{\text{in}}/r_{\text{out}} = 0$. To obtain a neural network that can deal with such partly populated input vectors, we perform training using the training data sets (i) and (ii) where we repeatedly erase a random number of entries at random positions. When testing the neural network we erase entries in the same way.

In Fig. 9 we show the absolute error of the available bandwidth estimates that are obtained by the neural network if $m \in [1, k]$ randomly selected entries of the k-dimensional input vector are given. The bars show the average error and the standard deviation of the error. The data set used for testing is the same as the one used for Fig. 4(a) previously, i.e., $C = 100$ Mbps and $A \in [25, 50, 75]$ Mbps. We show the combined results for all values of A. The average error shows a clear improvement with increasing m whereas the standard deviation first grows slightly up to $m = 5$ before it eventually starts to improve. The reason is that for $m = 1$ the information is not sufficient to identify the two unknown parameters capacity and available bandwidth. Hence, the neural network first reports conservative estimates in the middle range. For comparison, by guessing 50 Mbps in all cases the average error is 16.6 Mbps for the given test data set. With increasing m the neural network starts to distinguish the range of $A \in [25, 50, 75]$ Mbps but tends to frequent misclassifications that can cause large errors. These misclassifications are mostly resolved when increasing m further. We observe the same trend also for the error of the capacity estimates that shows a high correlation with the error of the available bandwidth estimates. Hence, we omit showing the results.

B. Recommender Network for Probe Rate Selection

When adding entries to the partly populated input vector of the neural network, the average estimation error improves. The amount of the improvement depends, however, on the position of the a priori unknown entry that is added, as well as on the m entries that are already given, i.e., their position and value. We use a second neural network that learns this interrelation. Using this knowledge, the neural network acts as a recommender that given a partly populated input vector selects the next probe rate, i.e., the next entry, that is expected to improve the accuracy of the bandwidth estimate most. The recommender network takes the $k = 20$-dimensional input vector of values $r_{\text{in}}/r_{\text{out}}$, has 80 hidden neurons, and generates a k-dimensional output vector of estimation errors that apply if the entry $r_{\text{in}}/r_{\text{out}}$ is added at the respective position. Given the output vector, the rate r_{in} that minimizes the estimation error is selected for probing next.

Fig. 10 shows how the recommender network improves the error of the bandwidth estimates compared to the random selection of probe rates in Fig. 9. Starting at 5 selected probe rates, the average estimation error as well as the standard deviation of the error are small and adding further probe

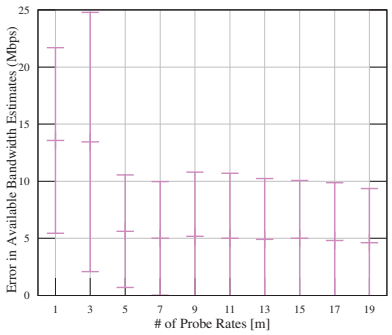


Fig. 10. Error of the available bandwidth estimates obtained for a set of m recommended probe rates.

rates improves the estimate only marginally. The reason is that certain probe rates, e.g., those on the horizontal line at $r_{\text{in}}/r_{\text{out}} = 1$ in Fig. 2, provide little additional information. We conclude that the recommender can effectively control the selection of probe rates to avoid those rates that contribute little. In this way, the recommender can save a considerable amount of probe traffic.

VI. Conclusion

We investigated how neural networks can be used to benefit measurement-based available bandwidth estimation. We proposed a method that is motivated by the characteristic rate response curve of a network. Our method takes a vector of ratios of equidistantly spaced probe rates at the sender and at the receiver $r_{\text{in}}/r_{\text{out}}$ as input to a neural network to estimate the available bandwidth and the bottleneck capacity. We use ratios of data rates and a suitable normalization to achieve an implementation that is scale-invariant with respect to the network capacity. 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 different bottleneck capacities. To reduce the amount of probe traffic, we implemented an iterative method that varies the probe rate adaptively. The selection of probe rates is performed by a neural network that acts as a recommender. The recommender effectively selects the probe rates that reduce the estimation error most quickly.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

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

[9] 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.

[10] 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.

[11] X. Liu, K. Ravindran, and D. Loguinov, "A stochastic foundation of available bandwidth estimation: Multi-hop analysis," *IEEE/ACM Transaction on Networking*, vol. 16, no. 1, pp. 130–143, 2008.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] R. Prasad, M. Jain, and C. Dovrolis, "Effects of interrupt coalescence on network measurements," in *Passive and Active Measurement Workshop*, 2004, pp. 247–256.

[17] Q. Yin and J. Kaur, "Can machine learning benefit bandwidth estimation at ultra-high speeds?" in *Passive and Active Measurement Conference*, 2016, pp. 397–411.

[18] S. Keshav, "A control-theoretic approach to flow control," in *Proc. ACM SIGCOMM*, Sep. 1991, pp. 3–15.

[19] V. Paxson, "End-to-end internet packet dynamics," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 277–292, 1999.

[20] Z. Bozakov and M. Bredel, "Online estimation of available bandwidth and fair share using Kalman filtering," in *IFIP Networking*, 2009.

[21] A. Eswaradass, X.-H. Sun, and M. Wu, "A neural network based predictive mechanism for available bandwidth," in *Parallel and Distributed Processing Symposium*, 2005.

[22] L.-J. Chen, "A machine learning-based approach for estimating available bandwidth," in *TENCON*, 2007, pp. 1–4.

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

[24] A. Kuznetsova, S. J. Hwang, B. Rosenhahn, and L. Sigal, "Exploiting view-specific appearance similarities across classes for zero-shot pose prediction: A metric learning approach," *Conference on Artificial Intelligence (AAAI)*, Feb. 2016.

[25] ——, "Expanding object detector's horizon: Incremental learning framework for object detection in videos," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[26] 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.

[27] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *ACM Internet Measurement Conference*, Oct. 2015.

[28] 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.

[29] J. Laine, S. Saaristo, and R. Prior, "Real-time udp data emitter (rude) and collector for rude (crude)," 2000. [Online]. Available: https://sourceforge.net/projects/rude/