Measurement-based Online Available Bandwidth Estimation employing Reinforcement Learning

Sukhpreet Kaur Khangura and Sami Akın
Institute of Communications Technology
Leibniz Universität Hannover

***Abstract*—An accurate and fast estimation of the available bandwidth in a network with varying cross-traffic is a challenging task. The accepted probing tools, based on the fluid-flow model of a bottleneck link with first-in, first-out multiplexing, estimate the available bandwidth by measuring packet dispersions. The estimation becomes more difficult if packet dispersions deviate from the assumptions of the fluid-flow model in the presence of non-fluid bursty cross-traffic, multiple bottleneck links, and inaccurate time-stamping. This motivates us to explore the use of machine learning tools for available bandwidth estimation. Hence, we consider reinforcement learning and implement the single-state multi-armed bandit technique, which follows the ε-greedy algorithm to find the available bandwidth. Our measurements and tests reveal that our proposed method identifies the available bandwidth with high precision. Furthermore, our method converges to the available bandwidth under a variety of notoriously difficult conditions, such as heavy traffic burstiness, different cross-traffic intensities, multiple bottleneck links, and in networks where the tight link and the bottleneck link are not same. Compared to the piece-wise linear network a model-based direct probing technique that employs a Kalman filter, our method shows more accurate estimates and faster convergence in certain network scenarios and does not require measurement noise statistics.**

***Index Terms*—Available bandwidth estimation, network measurements, reinforcement learning, multi-hop networks.**

I. Introduction

Real-time available bandwidth estimation in a communication network has been of interest to researchers due to its significant impact on delay sensitive Internet applications. For example, an accurate available bandwidth estimation in real-time streaming multimedia applications is fundamental to provide certain Quality-of-Service (QoS) guarantees to end users. Furthermore, available bandwidth estimates are used to select the best route, to monitor and detect the congestion, and to balance the traffic across a network to avoid stops, lags or buffering in the streaming content. Herein, the term *available bandwidth* refers to the residual capacity that remains available for data transmission after cross-traffic is served. Formally, given a link with capacity C and a cross-traffic with long-term average rate λ, where $\lambda \in [0, C]$, the available bandwidth of the link is defined as $A = C - \lambda$ [1]. Here, the end-to-end available bandwidth is determined by the *tight link*, i.e., the link with the minimal available bandwidth [2], and not by the bottleneck link, i.e., the link with the minimal capacity.

Researches have proposed several active probing techniques and corresponding theories for available bandwidth estimation [1]–[13]. These techniques use a sender that actively injects into a network synthetic probe traffic with known packet size, l, and a well-defined inter-packet gap, i.e., input gap g_{in}. As these probe packets traverse through the network, they get dispersed due to cross-traffic. At the receiver, this inter-packet dispersion, i.e., output gap g_{out}, is measured to estimate the available bandwidth in the network. These techniques have a common assumption that the cross-traffic in a network has *constant-rate*. Moreover, these techniques assume a fluid-flow model and neglect the impacts of the packet granularity of the cross-traffic, i.e., the cross-traffic is assumed to be composed of infinitely small packets. Following the constant-rate fluid-flow cross-traffic assumption, a single tight link is modeled as a lossless first-in, first-out (FIFO) multiplexer of the probe traffic and the cross-traffic. Herein, the relation between g_{out} and g_{in} is given as [1]

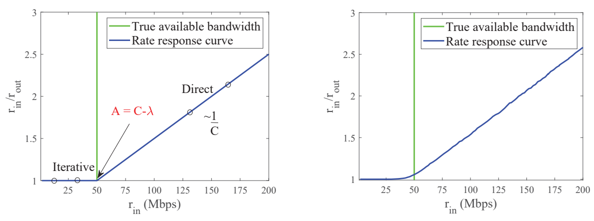
$$g_{\text{out}} = \max\left\{g_{\text{in}}, \frac{g_{\text{in}}\lambda + l}{C}\right\}, \tag{1}$$

where λ is the constant cross-traffic rate. Above, $g_{\text{in}}\lambda$ represents the amount of the cross-traffic that enters between two probe packets having an input gap of g_{in}, and causes them to be further apart. Reordering (1), we particularly obtain the characteristic *gap response curve* as

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

In practice, the output gaps g_{out} are highly distorted due to deviation 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. Therefore, the state-of-the-art bandwidth estimation methods average several output gap samples g_{out} in order to alleviate the observed variability of the samples of g_{out}. The samples can be collected by repeatedly sending *packet pairs* [14], or *packet trains* [5], [15], which consist of n consecutive packets, and hence, $n-1$ input gaps. At the receiver, the consecutive output gaps are formulated as $g^j_{\text{out}} = t^{j+1}_{\text{out}} - t^j_{\text{out}}$ for $j = 1 \ldots n{-}1$, where t^j_{out} is the time when the j^{th} packet arrives at the receiver. Then, the output rate of a packet train with n packets is given as

$$r_{\text{out}} = \frac{(n-1)l}{t^n_{\text{out}} - t^1_{\text{out}}}. \tag{3}$$



(a) Fluid-flow model (b) Single tight link

Fig. 1. Rate response curves of (a) the fluid-flow model assuming constant rate cross-traffic and (b) a single tight link with an exponential cross-traffic.

Since we define $g_{out}^j = t_{out}^{j+1} - t_{out}^j$, we can rewrite (3) as

$$r_{\text{out}} = \frac{l}{\frac{1}{n-1}\sum_{j=1}^{n-1} g_{\text{out}}^j}. \tag{4}$$

Notice that the denominator in (4) converges to the mean of the output gaps with the increasing packet train size. Herein, assuming a deterministic fluid-flow model, i.e., $g_{\text{out}}^j = g_{\text{out}}$ for $\forall j$, we can see that

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

Similarly, defining the input rate as $r_{\text{in}} = l/g_{\text{in}}$, and inserting $r_{\text{in}} = l/g_{\text{in}}$ and $r_{\text{out}} = l/g_{\text{out}}$ into (2), we obtain the equivalent representation of the *rate response curve* as

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

which mathematically describes the clear bend in the rate response curve at $r_{in} = A$ as seen in Fig. 1(a).

A. State-of-the-Art Estimation Techniques

We can classify the active available bandwidth estimation methods which are based on the constant-rate fluid-flow model as *iterative probing* or *direct probing* techniques. Iterative probing techniques search for the turning point in the rate response curve by sending repeated probes at increasing rates in the region defined by $r_{\text{in}}/r_{\text{out}} = 1$. When r_{in} reaches $C - \lambda$, the available bandwidth saturates and increasing the probe rate to a value beyond the available bandwidth results in self-induced congestion and causes $r_{\text{in}}/r_{\text{out}} > 1$. As a consequence, a queue builds up at the multiplexer and increases one-way delays that can be detected by the receiver. This technique is implemented, for instance, by Pathload [6] and Pathchirp [7]. On the other hand, direct probing techniques estimate the upward segment of the rate response curve, i.e., the part where $r_{\text{in}} > C - \lambda$. This segment of the rate response curve is a function of C and λ. If we know C a priori, we need a single probe $r_{\text{in}} = C$ that yields a measurement, r_{out}, to estimate $\lambda = C(C/r_{\text{out}} - 1)$ from (5). We can see the implementation of this approach in [8]. If we do not have the knowledge of C in advance, we need at least two different probing rates $r_{\text{in}} > C - \lambda$ to obtain the two unknown parameters, C and λ. We can see this technique in, for instance, TOPP [3], DietTOPP [4], and BART [10].

In real-time available bandwidth estimation, noisy measurement data, multiple bottleneck links, and inaccurate time-stamping degrade the estimation quality. Furthermore, the stochastic nature of cross-traffic leads to deviations from the fluid-flow model. In order to improve available bandwidth estimation and reduce the impacts of the randomness in cross-traffic, the state-of-the-art estimation methods use statistical post-processing techniques, such as a Kalman filter [10], [17], majority rule [6], averaging repeated measurements [7], [8], and linear regression [4]. While packet trains and statistical post-processing techniques help to reduce the variability in available bandwidth estimation, they do not take care of the deviations from the deterministic fluid-flow model. For instance, looking at the experimental results[1] in Fig. 1(b), we can see that unlike the deterministic fluid-flow model in Fig. 1(a), the sharp bend around $r_{\text{in}} = C - \lambda$ that marks the available bandwidth is not clearly apparent. This elastic deviation from the fluid-flow model leads to biased estimates. Furthermore, it is difficult to tailor methods to specific hardware implementations that influence the measurement accuracy. Therefore, the fundamental limitations of the model-based state-of-the-art bandwidth estimation methods make researchers explore the use of machine learning techniques in bandwidth estimation.

Machine learning techniques have taken attention initially in [18], [19] and recently in [20], [21]. We see that the authors in [18] use network traffic data collected by passive measurements, while the authors in [19]–[21] use active probes to estimate the available bandwidth in NS-2 simulations [19], ultra-high speed 10 Gbps networks [20], and operational LTE networks [21]. Moreover, the authors in [19]–[21] use packet chirps [7], i.e., the probes of several packets sent at increasing rates. They achieve the rate increase by a geometric reduction of the input gap [19], by concatenating several packet trains with increasing rates to a multi-rate probe [20], and by linearly increasing the packet size [21]. The packet chirps, with a single probe, lead to the detection of the turning point, which is the actual available bandwidth. Nevertheless, the chirps are susceptible to random noise [12]. Furthermore, the authors in [19] study the packet bursts which are known as back-to-back packet probes and conclude that the packet bursts are not enough to estimate the available bandwidth. Also, the authors in [20] consider constant-rate packet trains in an iterative manner to attain the available bandwidth. Here, machine learning solves a classification problem to estimate whether the rate of a packet train exceeds the available bandwidth. Depending on the result, the rate of the next packet train is reduced or increased in a binary search [6] until the probe rate

[1]We obtained the results in Fig. 1(b) from the testbed shown in Fig. 2. The network is set with a single tight link of capacity $C = 100$ Mbps, and access links are of capacity $C = 1$ Gbps. The cross-traffic is discrete with a packet length of $l = 1514$ bytes. We further set the cross-traffic to moderate burstiness with exponentially distributed packet inter-arrival times and an average rate of $\lambda = 50$ Mbps. Particularly, we run the experiment 1000 times and average the results for smoothness in the presentation.

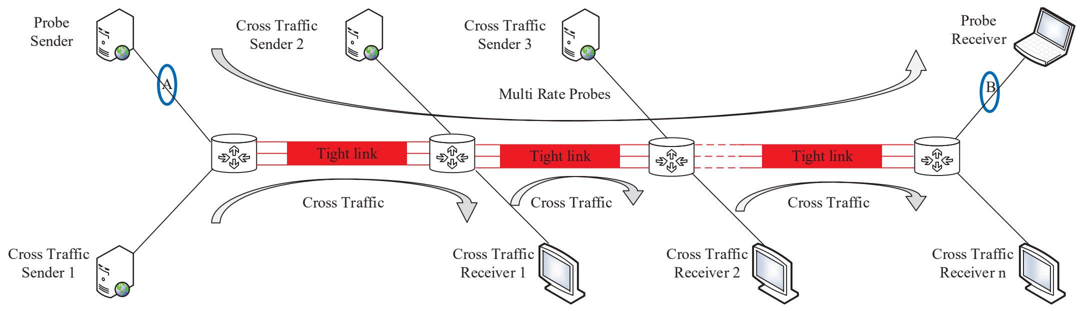


Fig. 2. Dumbbell topology set up using the Emulab and MoonGen software. A varying number of tight links with single hop-persistent cross-traffic are configured. Probe traffic is path-persistent to estimate the end-to-end available bandwidth from measurements at points A and B [16].

approaches the available bandwidth. The authors in [20] give, however, preference to the chirp probes.

The common aspect of the machine learning implementations in available bandwidth estimation is the use of output gap measurements [19], [21], the Fourier transforms of input and output gaps [20], or the $k \times 1$-dimensional vectors of $r_{\text{in}}/r_{\text{out}}$ [16] as labeled input features in training and testing phases. The major objective is to invoke supervised learning to extrapolate and generalize the available bandwidth estimates for the data sets not included in the training phase. However, from a practical point of view, it is not always feasible to create such training data sets which are representative of all cases because of the bursty nature of cross-traffic and multiple bottleneck links. As a consequence of this fundamental limit in supervised learning approaches [16], [18]–[21], we motivate ourselves to use reinforcement learning in available bandwidth estimation.

B. Contributions

In this paper, we propose a method to implement reinforcement learning in available bandwidth estimation, which converges the result faster and more accurately. We consider the set of input rates as the set of actions defined in reinforcement learning theory and define a reward metric as a function of input and output rates, which reaches the maximum when the input rate is equal to the available bandwidth. Our method is different from the existing machine learning approaches used in bandwidth estimation because it does not require a training phase. We evaluate our method in a controlled network testbed, where we specifically target topologies, including a bursty cross-traffic nature and multiple bottleneck links. We consider cross-traffic scenarios with different distributions and intensities. Furthermore, we compare our method with a fluid-flow model-based direct probing technique that employs a Kalman filter and show that our method converges faster and has less variations in bandwidth estimates. Moreover, we consider more difficult scenarios by setting the tight link different from the bottleneck link. Our method reliably performs real-time available bandwidth estimation in multi-hop networks with faster convergence and less variations, where the model-based direct probing technique underestimates the available bandwidth.

The remainder of this paper is organized as follows. We describe our experimental set up in Section II and present our reinforcement learning-based approach in Section III. We introduce the reference implementation of the state-of-the-art model-based direct probing technique in Section IV and show test results in Section V. We provide the conclusion in Section VI.

II. Experimental Setup

We set our controlled network testbed in Leibniz Universität Hannover, and it comprises 80 servers. We connect each server deploying minimum four network switches with 1 Gbps and 10 Gbps link capacities. We use the Emulab software [22] to manage the testbed, where we configure the servers as hosts and routers and connect them using virtual local area networks (VLANs) to implement the desired topology. We use a dumbbell topology with multiple tight links, as shown in Fig. 2. In order to emulate the characteristics of the links, such as capacity, delay, and packet loss, we employ additional servers in Emulab. We use the MoonGen software [23] for the emulation of link capacities that differ from the native physical Ethernet capacity. To achieve an accurate spacing of packets that matches the emulated capacity, we fill the gaps between packets with dummy frames by using MoonGen, which are later discarded at the output of the link. We use the "forward rate Lua script" for the MoonGen to achieve the desired forwarding rate at the transmission and reception ports of MoonGen.

We create cross-traffic models having different distributions by employing distributed internet traffic generator (D-ITG) [24]. Each cross-traffic is *single-hop-persistent*, i.e., at each link, fresh cross-traffic is multiplexed. The probe traffic that we deploy to estimate the end-to-end available bandwidth is *path-persistent*. Particularly, it travels the entire network path from the probe sender to the probe receiver. We use real-time user datagram protocol (UDP) data emitter and collector known as RUDE and CRUDE [25] respectively, in order to generate UDP probe streams. A probe stream consists of a series of k packet trains, each having n packets. These k different packet trains respectively correspond to k different probe rates that successively increase with an increment rate, δ_r. We set the packet length to $l = 1514$ bytes including

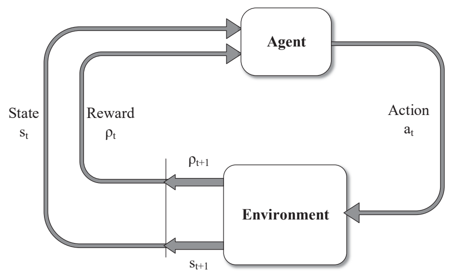


Fig. 3. An agent-environment interaction [26].

the Ethernet header, both in the probe traffic and the cross-traffic. We use "libpcap" to capture the packets at the probe sender and receiver, and the packet timestamps are generated at points A and B, respectively, as shown in Fig. 2. We also use a specific *Endace* data acquisition and generation (DAG) measurement card to obtain the accurate reference timestamps. We use the timestamps to compute r_{in} and r_{out} for each packet train.

III. Reinforcement Learning-based Method

In this section, we present our reinforcement learning-based method for available bandwidth estimation where we maximize a cumulative reward function by employing the exploration-exploitation mechanism and learn through environment observations without having a training phase. In the sequel, we start with the ε-greedy search algorithm and then discuss the reward function mechanism and the convergence speed of our method.

A. ε-greedy Algorithm

Let us consider a finite-state Markov decision process (MDP) with an agent and an environment, as shown in Fig. 3. Let us further consider that there are a set of states, $\mathcal{S}$, a set of actions, $\mathcal{A}$, and a set of rewards, $\mathcal{R}$. Here, we assume that there exists a bijective function between the sets of actions and states and the set of rewards. Particularly, there exists a one-to-one correspondence between the action-state pairs and the rewards. At time t, the agent in state $s_t \in \mathcal{S}$ chooses an action $a_t \in \mathcal{A}(s_t)$, and the environment returns a reward, $\rho_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and changes the agent's state to $s_{t+1} \in \mathcal{S}$. Here, $\mathcal{A}(s_t)$ refers to the set of actions that the agent chooses when it is in state s_t. Particularly, $\mathcal{A}(s_t)$ is a subset of $\mathcal{A}$. In a stochastic environment, the reward values following an action in one state can be samples from a distribution with a mean and variance. In this case, the reward of the action in that state can be the average of rewards received until the last time the action is chosen, and the agent is in that state. Under these conditions, given that the agent is in state s_t, the ε-greedy algorithm chooses with probability $1-\varepsilon$ the action $a_t \in \mathcal{A}(s_t)$ that performs the best with respect to reward returns, and selects uniformly one action among the others with probability ε. Particularly, the algorithm guides the agent with the best action observed while exploring with probability ε among the other actions to find a better action. Here, ε indicates how greedy the agent is, and the optimal value of ε is important especially in noisy environments because the agent needs to explore more in order to find the action that performs best on average. When ε is smaller, the agent converges to a reward value slowly and stabilizes on an action. However, although it is more stable in the long run, yet there is a risk that the reward value is not the maximum reward the agent could have. On the other hand, when ε is larger, it takes shorter to converge to a reward value, but there will be too much variations in the long-run even if the measurements are not very noisy. For more details, we refer interested readers to [26].

In our experiment, we consider that the network is stationary, i.e., the network statistics remain constant for the time interval during which we make our measurements and estimate the average available bandwidth. Therefore, we treat available bandwidth estimation as a single-state MDP multi-armed bandit problem. Herein, the set of input probe rates, i.e., $r_{in} \in \{\delta_r, 2\delta_r, \dots, k\delta_r\}$ in our setting corresponds to the set of actions, $\mathcal{A}$. We further define a reward parameter, which is a function of the r_{in} and r_{out}, and reaches the maximum when the input probe rate, r_{in}, is equal to the available bandwidth in the network. We provide the details of the reward function in the sequel.

Following the selection of one probe rate among k input rates, its associated reward is received. Because the reward values are perturbed due to noisy measurements, we rely on the corresponding average rewards after a probing rate is selected. Particularly, we calculate the action-value function $\mathcal{Q}_t(r_{in})$ that estimates the value for choosing r_{in} at time step t by calculating the average rewards received up to time $t-1$ as

$$\mathcal{Q}_t(r_{in}) = \frac{\sum_{j=1}^{t-1} \rho_j i_j(r_{in})}{\sum_{j=1}^{t-1} i_j(r_{in})}, \tag{6}$$

where $i_j(r_{in})$ is the indicator function, which is set to 1 whenever the input rate r_{in} is chosen up to time $t-1$ and is 0 otherwise. Here, the ε-greedy algorithm at time t chooses the input probe rate that has the maximum average reward up to time $t-1$ with probability $1-\varepsilon$. Specifically, the algorithm sets the input rate at time t, i.e., r_{in_t}, as

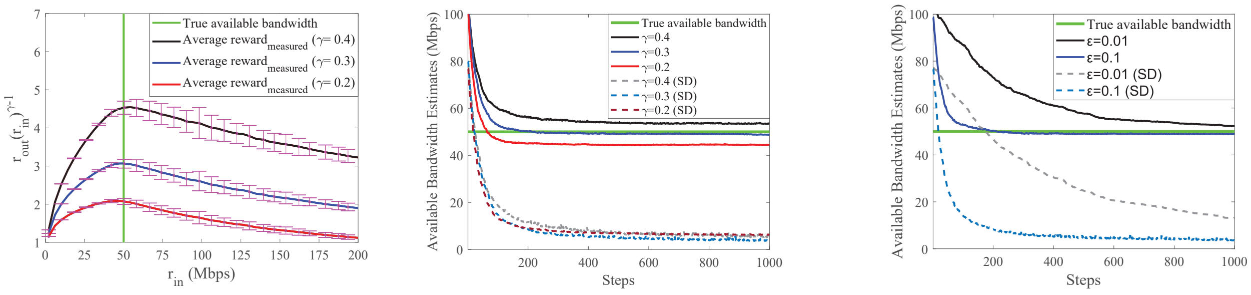
$$r_{in_t} = \arg\max_{r_{in} \in \mathcal{A}} \{\mathcal{Q}_t(r_{in})\}.$$

It uniformly chooses any rate among the others with probability ε and sets the input rate.

B. Choice of Reward Function

In real-time available bandwidth estimation, the major challenge is to define a function that produces a credible reward even in the presence of noisy measurements due to non-fluid traffic, multiple bottlenecks, and inaccurate time stamping. To combat the effect of noise, we define a reward metric which is a function of the r_{out} and $r_{in}^{\gamma-1}$, where γ is the convergence parameter satisfying $0 < \gamma < 1 - \frac{\lambda}{C}$. Formally, we define the reward function as

$$\rho = r_{out}(r_{in})^{\gamma-1}. \tag{7}$$



(a) Reward distribution

(b) Effect of γ on convergence speed

(c) Effect of ϵ on convergence speed

Fig. 4. (a) Reward distribution with average measured rewards and error bars depicting their SDs, and (b) & (c) average available bandwidth estimates and their SDs for different values of γ and ε, the two parameters that affect the convergence speed of reinforcement learning-based method.

The reward function in (7) reaches the maximum when r_{in} is equal to the available bandwidth as long as the convergence parameter, γ, is in the aforementioned defined range. We set the exploration rate $\varepsilon = 0.1$ and show the measured reward function for convergence parameter $\gamma \in [0.2, 0.3, 0.4]$ as a function of r_{in} averaged over 1000 repeated measurements in Fig. 4(a), taken in the network with a single tight link of capacity $C = 100$ Mbps in the presence of exponential cross-traffic with an average rate of $\lambda = 50$ Mbps. The cross-traffic packet size is $l = 1514$ bytes, and access links are of capacity 1 Gbps. The error bars depict the standard deviation (SD) from the average reward values, which increases when probing rate reaches beyond available bandwidth, i.e., $r_{in} > C - \lambda$ due to building up of queues at the multiplexer. As seen in Fig. 4(a), the reward function is maximized when r_{in} is equal to the available bandwidth, which is 50 Mbps. We also note that with decreasing γ, the reward function also decreases, which leads to slower convergence because the impact of noise is more belligerent with decreasing reward function when differentiating the maximum reward from the others.

C. Convergence Speed

Our proposed reinforcement learning-based method is a continuous process. Once the convergence is reached, it produces a stable value of available bandwidth estimate. However, the speed at which it converges depends upon the choice of two parameters, i.e., γ and ε.

1) Choice of γ: In a network with unknown C and λ, it is not trivial to determine γ, which depends on these unknowns by definition. To analyze the effects of γ on the convergence speed, we plot the average available bandwidth estimates and their standard deviations (SDs) over 1000 steps for $\gamma \in [0.2, 0.3, 0.4]$, as shown in Fig. 4(b). At each step, our method chooses one of the input rates among k input rates following the ε-greedy algorithm and provides a single available bandwidth estimate. As the number of steps increases, every input rate is sampled enough number of times leading to the convergence of the input rate with maximum reward value to the available bandwidth. Furthermore, we use standard deviation (SD) as a metric to measure the precision of the bandwidth estimates:

$$SD = \sqrt{\frac{1}{K_r - 1}\sum_{i=1}^{K_r}(\hat{A}_i - \bar{A}_i)^2}, \tag{8}$$

where $\hat{A}$ and $\bar{A}$ are the estimated and the true values of the available bandwidth, respectively and K_r is the number of repeated experiments over which we obtain the average available bandwidth estimates and their SDs around the true available bandwidth. We set $K_r = 1000$ and the exploration rate to $\varepsilon = 0.1$ unless otherwise stated. As seen in Fig. 4(b), the convergence is faster when $\gamma = 0.3$, i.e., the method detects the available bandwidth after 200 steps. On the other hand, it takes more than 1000 steps on average for the method to converge to the available bandwidth when $\gamma = 0.2$ and $\gamma = 0.4$. We plot the graphs until 1000 steps for clarity in the comparison of different γ values. One can run the experiment for more steps and can easily observe that as long as the convergence parameter satisfies $0 < \gamma < 1 - \frac{\lambda}{C}$, the method will converge. However, the convergence speed depends not only on γ but on the ε as well.

2) Choice of ε: The choice of ε dictates the exploration-exploitation trade-off in reinforcement learning-based methods. Hence, in order to understand the impact of ε, we plot the average available bandwidth estimates and their SDs for $\varepsilon \in [0.01, 0.1]$ with the convergence parameter set to $\gamma = 0.3$. The larger exploration rate, $\varepsilon = 0.1$, leads the method to explore more and find the available bandwidth faster when compared to the smaller exploration rate, $\varepsilon = 0.01$. However, although the method converges more quickly with larger ε, yet the method performs better with a smaller ε in the long run when the noise variance is low. This is because the method with large ε is able to reach an acceptable range of r_{in} that involves the available bandwidth. However, it leads to more variations in the available bandwidth estimation in the long-run since it tests other values very often. On the other hand, the method reaches a smaller range of r_{in} that maximizes the reward function very slowly when ε is smaller, but the variation around the available bandwidth is much smaller in the long-run.

IV. Model-based Reference Implementation

We compare our method with the piece-wise linear network model-based direct probing technique that employs a Kalman filter, which is provided in [10]. While available bandwidth estimation tools differ significantly regarding the selection and the amount of probe traffic. We test both techniques using the same database for the sets of input rates and output rates in order to provide a solid reference point. In the following section, we briefly describe the direct probing technique. For more information, we refer interested readers to [10].

A. Direct probing

In order to implement the direct probing technique in our testbed, we combine the active probing with a Kalman filter. Unlike in [10], to increase the convergence speed of the filter, we use a multi-rate probe stream of the k packet trains that correspond to k input rates r_{in} as in [27] to probe the network path with several rates in each experiment. We define the inter-packet strain as [10]

$$\xi = \frac{r_{in}}{r_{out}} - 1 \quad \text{for } r_{\text{in}} > C - \lambda. \tag{9}$$

After inserting ξ into (5) for $r_{in} > C - \lambda$, we rewrite (5) as

$$\xi = r_{\text{in}} \frac{1}{C} + \frac{\lambda - C}{C}. \tag{10}$$

By defining $\alpha = \frac{1}{C}$ and $\beta = \frac{\lambda - C}{C}$ we obtain the packet strain parameter as

$$\xi = \begin{cases} 0, & \text{if } r_{in} \leq C - \lambda, \\ \alpha r_{in} + \beta, & \text{if } r_{in} > C - \lambda. \end{cases} \tag{11}$$

Following the assumptions of the fluid-flow model, we can see that the expected value of ξ is zero in the absence of congestion, and it grows proportional to the probe traffic when the probing rate exceeds the available bandwidth. As seen in (11), the model is piece-wise linear due to the sharp bend at $r_{in} = C - \lambda$ which inhibits the direct application of the Kalman filter. In order to overcome the problem, we feed only the measurements that satisfy $r_{in} > \hat{A}$ to the filter, where $\hat{A}$ is the recent estimate of the available bandwidth. Since the direct probing technique seeks to estimate the upward line segment of the rate response curve which is determined by two parameters C and λ, we can express the state of the system with a state vector containing two unknown parameters as

$$x_t = \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix}. \tag{12}$$

Assuming that the network statistics remain constant during the observation period t, the transition matrix A becomes an identity matrix. Hence, we define the system state as [10]

$$x_t = x_{t-1} + w_{t-1}, \tag{13}$$

where w_{t-1} is the process noise and denotes the deviations from the fluid-flow model.

We define the measurement model as

$$z_t = H_t x_t + v_t, \tag{14}$$

where z_t is a $k \times 1$ dimensional vector of measured strains as

$$z_t = \left[z_t^1, z_t^2, .., z_t^k \right]^T, \tag{15}$$

where $\{\}^T$ is the transpose operator. The k packet trains corresponding to k different rates, increase the variance of measurement noise of a probe stream. In order to combat this effect, we compute the strain in (15) and the corresponding measurement noise v_t with covariance $R = \mathbb{E}\{v_t v_t^T\}$, which defines the precision of the strain measurement and is crucial for the tracking ability of the Kalman filter, for each packet train individually. Similarly, the observation matrix H_t consisting of different input rates, i.e., r_{in} corresponding to k packet trains is defined as

$$H_t = \begin{bmatrix} r_{in_t}^1 & 1 \\ .... & \\ r_{in_t}^k & 1 \end{bmatrix}.$$

Furthermore, the 2×2 covariance matrix of the process noise, $Q = \mathbb{E}\{w_t w_t^T\}$, describes the deviation of the system from the fluid-flow model, and it is treated as an adjustable parameter as in [10]. Q, being a symmetric matrix, provides three degrees of freedom for tuning; however, we use it in a simple form as $Q = \Lambda I$, where I is the 2×2 identity matrix. We choose $\Lambda = 10^{-2}$ in our settings because it allows faster convergence and less variations in available bandwidth estimates.

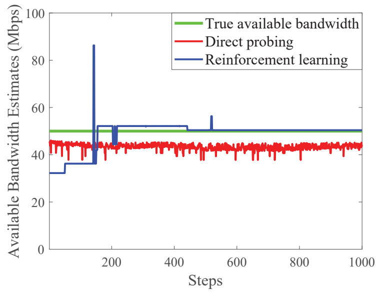
V. Experimental Evaluation

We evaluate the performance of our technique and compare it with the performance of the direct probing technique in a controlled network testbed described in Section II. We use the same setting we have in Section III unless otherwise stated. In Fig. 5(a), Fig. 5(b), Fig. 5(c), we show randomly selected the first three repeated experiments and available bandwidth estimation results employing the direct probing technique and the reinforcement learning-based method. As seen in the randomly selected experiments, our method outperforms the other method. However, in order to have a better view from a statistical perspective, we perform $K_r = 1000$ experiments and compare the average estimation performances and their SDs around the actual available bandwidth values in the sequel.

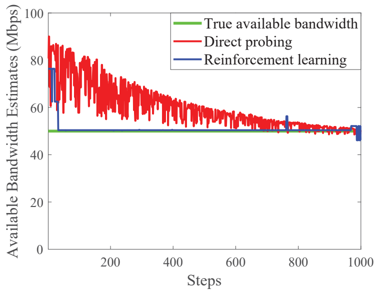
1) Cross traffic Burstiness: In order to evaluate how our method performs in the presence of cross-traffic with an unknown burstiness, we consider three types of cross-traffic:

1) No burstiness with constant bit rate,
2) Moderate burstiness due to exponential packet inter-arrival times,
3) Heavy burstiness due to Pareto inter-arrival times with infinite variance, defined by a shape parameter $\alpha = 1.5$.

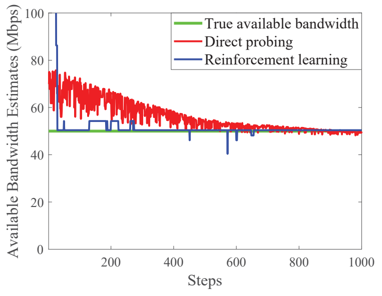
Recall that the burstiness of the cross-traffic can cause queueing at the tight link even if the probe rate is below the available bandwidth, i.e., if $r_{in} < C - \lambda$, which leads to deviations from the ideal rate response curve. We show these deviations in Fig. 1(b), and we observe the maximum deviation when



(a) $K_r(1)$

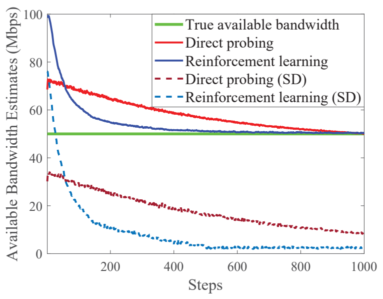


(b) $K_r(2)$

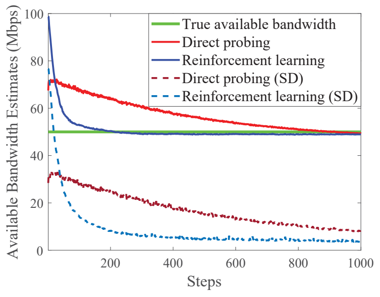


(c) $K_r(3)$

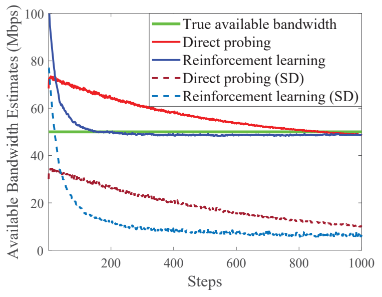
Fig. 5. Available bandwidth estimates for randomly selected first three repeated experiments.



(a) Constant Bit Rate

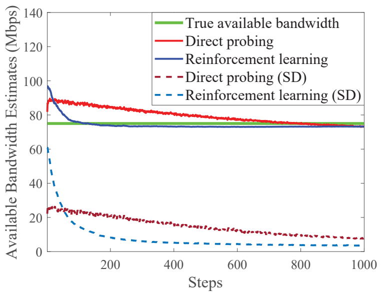


(b) Exponential

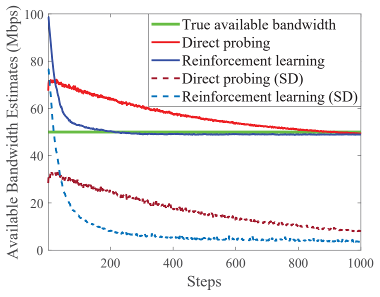


(c) Pareto

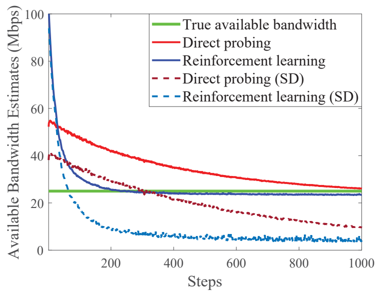
Fig. 6. Average available bandwidth estimates and their SDs for different types of cross-traffic burstiness.



(a) $\lambda = 25\ Mbps$



(b) $\lambda = 50\ Mbps$



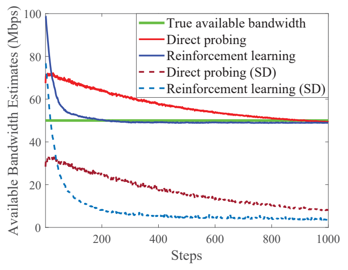
(c) $\lambda = 75\ Mbps$

Fig. 7. Average available bandwidth estimates and their SDs for different exponential cross-traffic rates $\lambda \in 25, 50, 75$ Mbps.

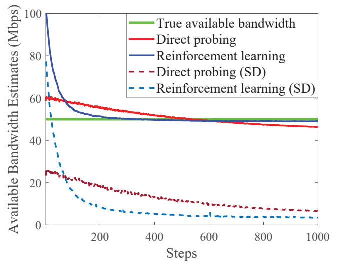
$r_{in} = C - \lambda$. Moreover, the strong deviation blurs the bend that marks the available bandwidth and causes an estimation bias. As seen in Fig. 6(a), Fig. 6(b) and Fig. 6(c), the bandwidth estimates of the reinforcement learning-based method are more accurate with low SDs when compared to the direct probing technique. We can see the significant improvement in the convergence speed when we employ the reinforcement learning-based method irrespective of the cross-traffic burstiness. Particularly, the reinforcement learning-based method is robust to the deviations from the fluid-flow model.

2) Cross Traffic Intensity: To evaluate the impacts of cross-traffic intensity on available bandwidth estimation, we deploy exponential cross-traffic with average rates $\lambda = 25$, 50, and 75 Mbps, and depict the average of the available bandwidth estimates and their SDs around the true available bandwidth in Fig. 7(a), Fig. 7(b) and Fig. 7(c), respectively. While the SDs increase in the direct probing technique with the increasing cross-traffic, the SDs in the reinforcement learning-based method remains almost unchanged in all cases. Moreover, the reinforcement learning-based method converges to the available bandwidth faster than the direct probing technique does.

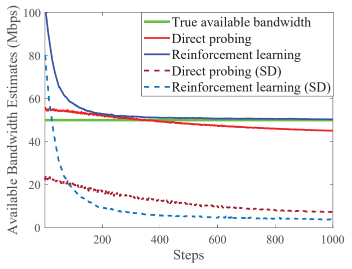
3) Multiple Tight Links: We extend our testbed from the single-hop network to the multi-hop network, as shown in Fig. 2, to test the reinforcement learning-based method in multiple tight links. While traversing the entire network path



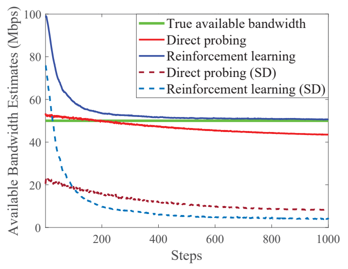
(a) 1-hop



(b) 2-hop



(c) 3-hop



(d) 4-hop

Fig. 8. Average available bandwidth estimates and their SDs for 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.

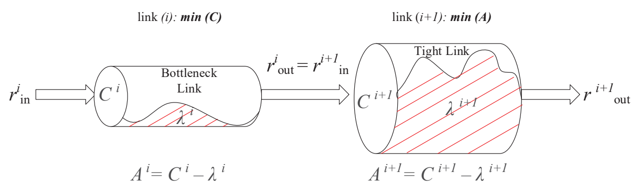


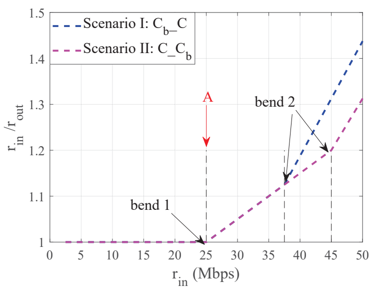
Fig. 9. A two-hop network where the tight link differs from the bottleneck link [30].

with the tight link capacity $C = 100$ Mbps and the access links with capacity 1 Gbps, the path-persistent probe streams experience single hop-persistent cross-traffic with exponential packet inter-arrival times and average rate $\lambda = 50$ Mbps. We show in Fig. 8 that the reinforcement learning-based method provides more accurate available bandwidth estimates, whereas the other method fails to converge to the available bandwidth. This can be explained by the fact that in the 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. In the following links, the input gaps have a random structure as they are the output gaps from the preceding links [1], [13], [28]. For the direct method, the inter-packet strain, ξ, does not grow linearly with the cross-traffic in multi-hop networks [29], which causes the underestimation of the available bandwidth.

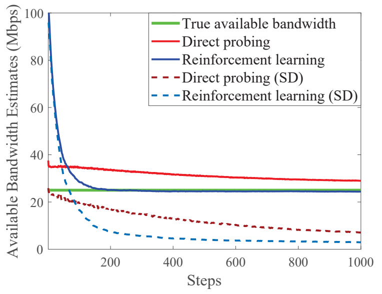
4) *Tight Link but not Bottleneck Link:* The available bandwidth estimation in multi-hop networks becomes more difficult if the tight link of a network is not the bottleneck link of the same network. As shown in Fig. 9, link i is the *bottleneck link*, and link $i+1$ represents the *tight link*. We investigate the available bandwidth estimation by considering two different scenarios. In Scenario I, the bottleneck link appears before the tight link. In Scenario II, the bottleneck link comes after the tight link. The existence of separate tight and bottleneck links has an impact on the shape of the rate response curve. As shown in Fig. 10(a), the curves are piece-wise linear. The two bends indicate the presence of two congestible links. We set the tight link capacity to $C = 100$ Mbps and the bottleneck capacity to $C_b = 50$ Mbps in both scenarios. We model the cross-traffic with constant bit rate $\lambda = 75$ Mbps and $\lambda_b = 12.5$ Mbps in the tight link and the bottleneck link respectively. We show the available bandwidth estimates and the corresponding SDs in Scenario I and II respectively in Fig. 10(b) and Fig. 10(c). The reinforcement learning-based method results in more accurate and faster estimates than the direct probing technique does. However, we observe an estimation bias in the available bandwidth estimates of the direct probing technique in both scenarios, i.e., no accurate convergence to the actual available bandwidth because of the congestion measure, ξ, that grows faster when the congestion occurs at both the tight and bottleneck links when the probing rates are larger than $C_b - \lambda_b = 37.5$ Mbps. The effect is more noticeable in Scenario II.

VI. Conclusion

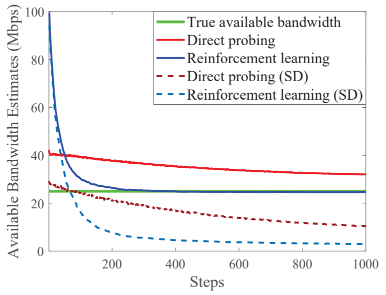
We have investigated how reinforcement learning can be utilized in measurement-based online available bandwidth estimation. We have proposed a method that runs ε-greedy algorithm to find the available bandwidth by maximizing the designated reward function. We have conducted a comprehensive measurement study in a controlled network testbed to analyze our proposed method and compare it with the piece-wise linear network model-based direct probing technique that employs a Kalman filter. Our results have shown that the reinforcement learning-based method can significantly



(a) Rate response curves [30]



(b) Available bandwidth estimates for scenario I



(c) Available bandwidth estimates for scenario II

Fig. 10. (a) Rate response curves with two bends indicating two congestible links and average available bandwidth estimates and their SDs with the tight link (b) succeeding and (c) preceding the bottleneck link.

improve the available bandwidth estimates by reducing bias and variability. The convergence of the reinforcement learning-based method is faster when compared to the direct probing technique. We have shown that our method provides better estimates in network configurations which deviate from the constant rate fluid-flow assumptions when there is cross-traffic with heavy burstiness and different intensities as well. We have further tested our method in network scenarios with multiple tight links, and in multi-hop networks where the tight link and the bottleneck link are different. We have shown that even though the additional links affect the convergence speed, the reinforcement learning-based method results in accurate available bandwidth estimates with less variability, whereas the other method does not.

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

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

[16] S. K. Khangura, M. Fidler, and B. Rosenhahn, "Neural networks for measurement-based bandwidth estimation," *IFIP Networking Conference*, 2018.

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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/

[26] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[27] M. Sedighizad, B. Seyfe, and K. Navaie, "MR-BART: multi-rate available bandwidth estimation in real-time," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 731–742, 2012.

[28] 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.

[29] E. Bergfeldt, "Available-bandwidth estimation in packet-switched communication networks," Ph.D. dissertation, Linköping University Electronic Press, 2010.

[30] S. K. Khangura, M. Fidler, and B. Rosenhahn, "Machine learning for measurement-based bandwidth estimation," *Computer Communications*, vol. 144, pp. 18 – 30, 2019.