Data-Driven Network Path Simulation with iBox

SACHIN ASHOK*, University of Illinois at Urbana-Champaign, USA
SHUBHAM TIWARI, Microsoft Research India, India
NAGARAJAN NATARAJAN, Microsoft Research India, India
VENKATA N. PADMANABHAN, Microsoft Research India, India
SUNDARARAJAN SELLAMANICKAM, Microsoft Research India, India

While network simulation is widely used for evaluating network protocols and applications, ensuring realism remains a key challenge. There has been much work on simulating network *mechanisms* faithfully (e.g., links, buffers, etc.), but less attention on the critical task of *configuring* the simulator to reflect reality.

We present iBox ("Internet in a Box"), which enables data-driven network *path* simulation, using input/output packet traces gathered at the sender/receiver in the target network to create a model of the end-to-end behaviour of a network path. Our work builds on recent work in this direction [7, 40] and makes three contributions: (1) estimation of a lightweight non-reactive cross-traffic model, (2) estimation of a more powerful reactive cross-traffic model based on Bayesian optimization, and (3) evaluation of iBox in the context of congestion control variants in an Internet research testbed and also controlled experiments with known ground truth.

CCS Concepts: • **Networks** → **Network simulations**; **Network measurement**; *Network performance modeling*; *Network experimentation*; *Network performance analysis.*

Additional Key Words and Phrases: data-driven simulation; cross-traffic estimation; bayesian optimization

ACM Reference Format:
Sachin Ashok, Shubham Tiwari, Nagarajan Natarajan, Venkata N. Padmanabhan, and Sundararajan Sellamanickam. 2022. Data-Driven Network Path Simulation with iBox. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 1, Article 6 (March 2022), 26 pages. https://doi.org/10.1145/3508026

1 INTRODUCTION

Network simulation (or emulation [1]) is widely used for evaluating network protocols and applications. Typically, a test network is rigged up by configuring a simulator by hand, including setting the type of network link (e.g., wired or wireless), link bandwidth, propagation delay, depth of network buffers, and the cross-traffic. While such an approach provides the developer with full control, it suffers from a lack of realism since the handpicked settings might not match the conditions of the target network, e.g., the Internet. Indeed, not only is it difficult to accurately recreate absolute performance matching real networks using such handpicked settings, even relative ordering in

*The work was done while the author was a Research Fellow at Microsoft Research India.
[1]We will not make a distinction between the two except where necessary.

Authors' addresses: Sachin Ashok, sachina3@illinois.edu, University of Illinois at Urbana-Champaign, Champaign, USA; Shubham Tiwari, t-shutiwari@microsoft.com, Microsoft Research India, Bengaluru, India; Nagarajan Natarajan, nagarajan.natarajan@microsoft.com, Microsoft Research India, Bengaluru, India; Venkata N. Padmanabhan, padmanab@microsoft.com, Microsoft Research India, Bengaluru, India; Sundararajan Sellamanickam, ssrajan@microsoft.com, Microsoft Research India, Bengaluru, India.

2476-1249/2022/3-ART6 $15.00
https://doi.org/10.1145/3508026

performance between protocols could be difficult to recreate. For instance, the relative ordering between TCP Cubic and TCP BBR in terms of metrics such as throughput depends on the network conditions (e.g., the buffer depth) [12], which can vary and hence is difficult to set appropriately by hand. Therefore, we believe that a data-driven alternative that can recreate the relative ordering between protocols as well as achieve an absolute match with respect to real world performance is quite desirable.

There has been recent work on configuring network simulators based on empirical measurements of end-to-end network paths between pairs of senders and receivers [40], including a short workshop paper by us [7]. Our present work, iBox– short for Internet in a Box – builds on this prior work, focusing on simulating the end-to-end behaviour of a network *path* (e.g., the delay and loss suffered by packets). iBox is not directly concerned with the internals of the target network; the details (such as topology) only matter to the extent it has a bearing on the end-to-end behaviour, which in turn could impact an application/protocol under test.

iBox makes three significant contributions over prior work, which are outlined below and expanded on in Section 3. Central to these contributions is the modelling of cross-traffic (i.e., other traffic that interferes with the application/protocol under test), which, as we show, has a strong bearing on faithfulness of the network path simulation.

First, we present a lightweight estimation of cross-traffic as a *non-reactive* rate time series based on a simple network path model (NRCT, Section 4). "Non-reactive" means that the estimated cross-traffic, which is replayed during simulation, does not react to the protocol under test. While such non-reactiveness is a simplification, this approach nevertheless ensures that the dynamics of the protocol under test are played out appropriately, which is an advantage over past work that had simply replayed a recorded delay trace from training [39]. We detail the estimation method for NRCT, including careful modeling of the queue dynamics to determine when a valid estimate can be obtained.

Second, going beyond just estimating the cross-traffic rate as a time series, we also develop a *reactive* cross-traffic (RCT, Section 5) model to help ensure realism. Such cross-traffic reacts appropriately to the actions of the sender under test, just as it would in an actual network. For instance, an aggressive sender (e.g., TCP Cubic bulk transfer that always looks to send more) might cause the cross-traffic to back off, while a timid one (e.g., real-time streaming that is codec-limited) might lead to cross-traffic filling up the slack in the pipe and network buffers. Clearly, either not modeling cross-traffic (e.g., as in [40]) or replaying a cross-traffic trace in a non-reactive manner (e.g., as in [7]) would not yield a realistic recreation. Building a reactive model of cross-traffic, however, is quite challenging since we do not have any direct measurements of or information on the nature of the cross-traffic in a real network setting. Nevertheless, we show promising results with a Bayesian optimization based approach to learning a reactive cross-traffic model, seeded suitably based on hints from the simpler non-reactive cross-traffic inference.

Finally, we evaluate iBox (Section 6) in the context of congestion control variants using the Pantheon research testbed [31] data. Our evaluation goes beyond prior work [40] in that we train iBox with just data from one flavour of TCP (e.g., Cubic) but then are able to faithfully recreate network behaviour for a different, previously unseen protocol (e.g., Vegas, BBR). In effect, we are able to use iBox to accurately simulate a new protocol without requiring access to any data whatsoever for this new protocol, which we believe provides a powerful tool to network developers and researchers. Furthermore, in terms of cross-traffic modeling, our results indicate a trade-off: NRCT is much more efficient to estimate but RCT yields more accurate results. While the ready availability of data makes Pantheon convenient for our evaluation, iBox could be trained with similar packet traces from other desired target network settings too. iBox models (derived from Pantheon data currently) have been made available for public download [2].

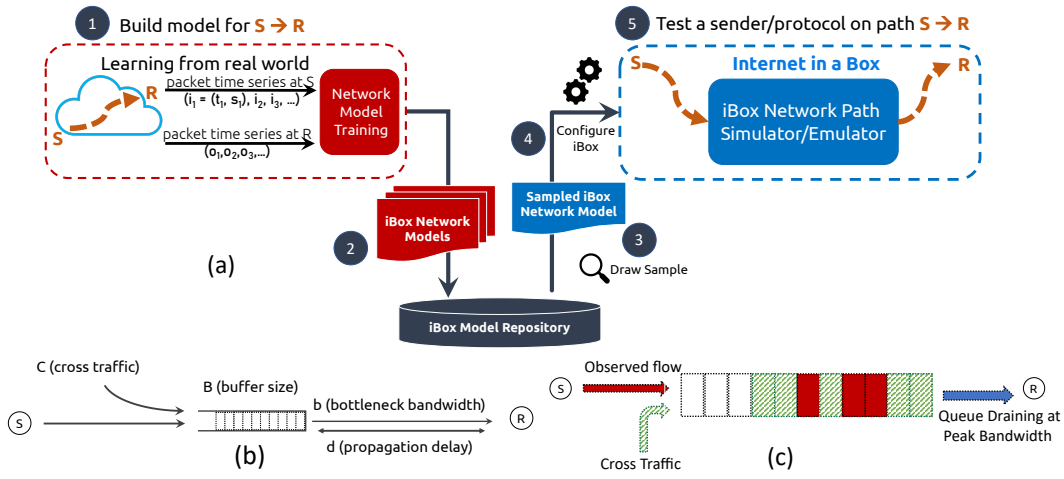

Fig. 1. (a) Framework for iBox training (left) and its use in simulation (right), (b) Simplified network path model in iBox, and (c) Modeling the bottleneck queue for estimating non-reactive cross-traffic (Section 4).

2 PROBLEM CONTEXT

Our focus is on a network path simulator that takes an input stream of packets at a sender host S and produces an output stream at the receiver host R. The goal is to mimic the behaviour of the target network, with each packet being delayed or dropped just as it would be in the target network. We are only concerned with the end-to-end behaviour of the network path, so the internals of the network are not of direct interest and only matter to the extent such details are necessary to recreate the end-to-end behaviour.

The training data comprises input/output packet traces gathered at senders and receivers of unicast network flows in the target network. For each packet, the trace contains the timestamp (recorded locally but assumed to be synchronized via a protocol such as NTP) and the size. Note that we consider each pair of sender host S and receiver host R separately. If there are multiple concurrent unicast flows between S and R, we consider these, and refer to these, collectively as a "single" flow between S and R for the purposes of modeling the queue dynamics (Section 4). The "target network" would, in general, comprise a heterogeneous mix of paths. For instance, if the target network were the Internet, it could include paths spanning different geographies, ISPs, network types (e.g., Ethernet, WiFi, cellular), etc. Clearly, it is neither feasible nor desirable to build a single omnibus simulation model for all such paths. Instead, we focus on building a model corresponding to each flow (with the collection of concurrent flows between a sender and receiver treated as a "single" flow, as noted above), tagged with corresponding metadata such as the geographic location, connectivity type, etc., and then letting the user sample from a repository of such models, as desired, when running simulations. For example, if the developer is only interested in simulating the network paths to WiFi nodes in North America, they could sample from just that subset.

Figure 1(a) summarizes the iBox framework as described here. Network models for a target network path between S and R are learnt by the iBox framework (step 1) and are added to a model repository (step 2). Users then sample network models from this repository (step 3), configure the simulator/emulator using this sampled network model (step 4), and test their sender/protocol in simulation/emulation (step 5).

Typical Use Case: A typical use case for an end-to-end network path simulator such as iBox is the evaluation of new network protocols or applications. The typical approach for such evaluation is A/B testing through "flighting", which refers to the treatment (B) being rolled out to a fraction of users and the results then compared with that for the control (A). iBox holds the promise of replacing flighting with realistic simulation, with both A and B being evaluated in the same simulation setup — with significant speed-up and without the risk of a regression negatively impacting users.

Certain characteristics of such A/B testing make iBox suitable. First, the evaluation of end-to-end protocols and applications in the predominant unicast setting (e.g., congestion control in transport protocols) is only concerned with end-to-end behaviour of a network path, not that of the network as a whole. Second, in many cases, the control and treatment operate in the backdrop of an unmodified network (e.g., a new video codec (B), which is possibly quite different from the existing codec (A), being evaluated on Internet paths where the nature of competing traffic remains unchanged), and so iBox is suitable for such a setting. However, in case the developer wishes to evaluate the test protocol in a setting where the nature of the network traffic has changed entirely from what it was when training data was gathered (e.g., training data was gathered when Cubic was the dominant network protocol while the developer is interested in a world where BBR dominates), the cross-traffic model learnt by iBox from the training setting would no longer be valid. For such a scenario, a pre-trained iBox model would not be a suitable choice and it would have to be re-learnt using updated data from the real-world.

3 OVERVIEW

iBox builds on recent work [7, 40] on configuring network simulators based on measurement data, with cross-traffic modeling as a novel contribution and differences in methodology (see Section 8). The idea is to start with a simplified and parameterized model of the network path and then find the parameter settings that best explain or fit the end-to-end measurements.

We use a simple, single-bottleneck-link, FIFO, drop-tail model (Figure 1(b)), which is characterized by the bottleneck bandwidth (b), buffer size (B), and the end-to-end propagation delay (d). In addition, cross-traffic is a key parameter, as is borne out by our results (Section 6). While our model is simple, our experimental results show that it nevertheless enables accurate recreation of end-to-end performance even in complex settings such as cellular networks (though we should point out that the Pantheon cellular data is only for static nodes and therefore may not exhibit the dynamic channel characteristics of a mobile node). More careful modeling of network characteristics such as variable link bandwidth is an important direction for our future work, which we touch on briefly in Section 9.

To train iBox we focus on just TCP Cubic data, which we believe is appropriate given the dominance of this protocol in the Internet and the consequent ease of obtaining training data. The iBox parameters b, B, and d are estimated directly using network domain knowledge, which is much more computationally efficient than obtaining these through a search/optimization procedure. However, we employ Bayesian optimization for inferring the reactive cross-traffic model, as noted below; but here again working with a popular protocol, such as TCP Cubic, enables fast training by leveraging the implementation of the protocol in the ns simulator to speed up the Bayesian optimization search.

The combination of the basic parameters (b, B, d) and cross-traffic obtained for each training flow is added to the iBox repository. At simulation time, we draw from this repository of realistic combinations to configure iBox accordingly. Unlike in typical network simulators, the parameters of iBox are *not* intended to be configured by hand.

3.1 Estimation of Basic Parameters and Cross-traffic in iBox

The first step is estimating the basic parameters of the iBox network model. The bottleneck bandwidth b is estimated based on the peak receive rate, propagation delay d based on checks for an empty bottleneck buffer, and the buffer size B based on checks for a full buffer. In Appendix A, we discuss our estimation of these basic parameters and also compare our approach with simpler alternatives (e.g., estimating the buffer size based on the maximum and minimum delays observed, without any checks for a full or empty buffer).

In **Section 4**, we detail an approach to estimating cross-traffic as a *non-reactive* time series. We formulate the conditions under which a valid estimate can be obtained, and perform interpolation otherwise. Here, "non-reactive" means that the cross-traffic does not react to the offered load, i.e., the estimated cross-traffic rate is just replayed during simulation.

In **Section 5**, we learn a novel *reactive* cross-traffic model, using Bayesian optimization, which reacts to the offered load, just as cross-traffic typically would in a network and unlike the non-reactive replay noted above. The reactive model is expressed as a mix of "basis" flows. We demonstrate the importance of learning such a reactive model of cross-traffic to accurately recreate the network path conditions, especially when the training data was obtained from say a protocol that is relatively aggressive (e.g., Cubic [18]) while the protocol being evaluated in simulation is much less so (e.g., LEDBAT [35], a background transfer protocol, or Vegas [11], a delay-sensitive protocol), with the result that the cross-traffic would behave rather differently in the two cases.

3.2 Using iBox for Simulation

The procedures noted above yield a set of parameters corresponding to each flow in the training dataset comprising (i) the b, B, and d parameters; and (ii) the cross-traffic estimate, whether as a non-reactive model (a C time series) or a reactive one (in terms of a set of basis flows). The combination of (i) and (ii) corresponding to each training flow is stored. At simulation time, we simply pick out one of these combinations to configure iBox.

3.3 Evaluation Methodology

Dataset: Much of our evaluation is based on data from the Pantheon testbed at Stanford [31]. This comprises data for 15+ flavours of congestion control protocols run between 12 client sites across the globe during a 3-year period from 2017 to 2020 (not all sites have data corresponding to the entire period). Each connection was of 30-second duration. The data used for our evaluation is summarized in Table 1. For the sake of consistency and easy visual comparison, the majority of the plots in the paper are for the Colombia Ethernet and the China Cellular datasets. Nevertheless, to showcase the diversity of the data analyzed, we also summarize results from the other datasets in Table 5 and, in select cases, we also present graphs.

In general, the only measurements available to us for the purpose of training iBox as well as testing it are from a "foreground" flow. These are the flows for which packet traces are available in the Pantheon dataset. The cross-traffic flows in the background are not directly observable by us, except in the controlled emulation experiments.

Foreground protocols used for evaluation: In our evaluation, we focus on foreground traces for TCP Cubic (for training) and foreground traces for TCP BBR, LEDBAT, and TCP Vegas (for testing). The former (Cubic) is the control (A), while the latter (Vegas, BBR or LEDBAT) is the treatment (B), which is previously unseen. TCP Cubic is a widely used transport protocol and is the default flavour of TCP on Linux, MacOS and Microsoft Windows [4]. Hence, it is relatively easy to obtain TCP Cubic packet traces, making it a good choice for the training protocol. The test protocols were chosen to exercise particular characteristics of their protocol design; TCP BBR is an up and

Type	Location	Period	# Cubic traces	# Vegas traces	# BBR traces
Ethernet	China	2 years	50	50	50
Ethernet	Colombia	2 years	50	50	50
Ethernet	Mexico	2 years	50	50	50
Ethernet	Nepal	1 week	10	10	NA
Cellular	China	2 years	51	49	50
Cellular	India	2 years	137	138	137
Cellular	US	1 year	81	80	80
WiFi	Nepal	2 years	43	43	NA

Table 1. Pantheon data sets used for evaluation.

coming protocol designed to combat buffer bloat [3], LEDBAT is a scavenger protocol intended for background applications (such as BitTorrent [5] and software updates [1]), while TCP Vegas is a delay-based protocol in contrast to a loss-based protocol like TCP Cubic. So, the Cubic traces are used to learn iBox models. The results for Vegas, BBR and LEDBAT obtained by running with these iBox models are compared with the ground truth in the data set to check for a statistical match. This evaluation methodology reflects how iBox will be used in practice; testing on a different and previously unseen (treatment) protocol is more challenging than testing on the control protocol itself (which is used for training iBox), e.g., as in [40].

Controlled experiments with known ground-truth: In addition, to evaluate the accuracy of cross-traffic estimation, we conduct controlled experiments using the NetEm [25] emulator, which enables us to control the amount of cross-traffic and facilitates a direct comparison of our estimate with the ground truth. We use a single-bottleneck topology, with the bottleneck bandwidth set in the 100 Kbps to 5 Mbps range, buffer size in the 10 to 150 MTU sized packets range, and propagation delay in the 30 to 150 ms range (network parameters estimated from the India Cellular traces). For the senders, we use Cubic, BBR, and Vegas implementations available in the Linux kernel.

4 NON-REACTIVE CROSS-TRAFFIC

Our non-reactive model of the cross-traffic is a time series, C, of the estimated rate of cross-traffic. A trace of per-packet delay for a flow from a sender to a receiver is used to estimate C. The estimation procedure is based on the simplified network model depicted in Figure 1(b), with a single bottleneck link fed by a FIFO, drop-tail queue.

Figure 1(c) depicts the queue in the assumed network model. Consider the change in the state of the queue over the course of an n-packet window [1] in the observed flow for which we have traces – from packet k to packet $k+n$. Let s_i be the size of the i^{th} packet in the measurement flow and τ_i the inter-packet spacing between the i^{th} and the $i+1^{th}$ packets, so the n-packet window spans a duration of $t_w = \sum_k^{k+n} \tau_i$. The corresponding change in the queue length, from $Q(k)$ to $Q(k+n)$, is impacted by 3 phenomena:

(1) The inflow of packets into the queue from the observed flow, totalling $\sum_k^{k+n} s_i$. We subtract out the contribution of dropped packets, if any, under the assumption that such packets were dropped at (or before) the bottleneck link and so did not enter the bottleneck buffer.
(2) The inflow of cross-traffic, which would be $C([k, k+n]) * t_w$, where $C([k, k+n])$ is the average rate of the cross-traffic in the $[k, k+n]$ packet window.
(3) The outflow of packets due to the draining of the queue, which would be $b * t_d$, where b is the bottleneck link bandwidth and t_d is the drain time (i.e., when the queue is non-empty and hence draining).

[1] n = 100 packets in our experiments

Therefore:

$$Q(k+n) - Q(k) = \sum_{k}^{k+n} s_i + C([k, k+n]) * t_w - b * t_d \Rightarrow$$

$$C([k, k+n]) = \frac{Q(k+n) - Q(k) - \sum_{k}^{k+n} s_i + b * t_d}{t_w} \Rightarrow$$

$$C([k, k+n]) = \frac{Q(k+n) - Q(k) - \sum_{k}^{k+n} s_i + b * t_d}{\sum_{k}^{k+n} \tau_i} \tag{1}$$

Q(k) can be estimated as the peak bandwidth times the measured queuing delay, akin to how the full buffer size is estimated in Section A.3.

Estimating the drain time, t_d, is key. We can be sure that the queue is non-empty and therefore draining in the intervening period between two packets, p and q of the observed flow (where packet p precedes packet q, or $p < q$), if we can determine that both packets were co-resident in the queue. The packets would be co-resident in the queue when the sender-side spacing between the packets, $\tau_{pq} = \sum_{p}^{q} \tau_i$, is less than the queuing delay experience by the leading packet, $d_p - d$ (the one-way delay experienced by packet p minus the estimated one-way propagation delay), i.e., when $\sum_{p}^{q} \tau_i < d_p - d$.

We find intervals when the non-empty queue condition is satisfied and estimate the cross-traffic during those periods using Equation (1) above. For other intervals, where we are unable to establish that the queue remained non-empty, we estimate the cross-traffic by interpolating the cross-traffic estimates from the neighbouring intervals for which we do have the estimates based on Equation (1).

Finally, while performing simulation with iBox, the estimated cross-traffic time series, C, is to be injected directly into the bottleneck buffer, as depicted in Figure 1 (c), avoiding any uncertainty in whether the cross-traffic packets actually make it to the bottleneck buffer. However, for the sake of simplicity of implementation, we connect a variable-bit traffic generator directly to the bottleneck buffer to send the estimated cross-traffic, which makes it likely that most or all of these packets end up in the bottleneck buffer.

4.1 Evaluation of Non-reactive CT Model

We evaluate the non-reactive cross-traffic (NRCT) model, both in a controlled setting (emulation), which provides knowledge of the ground truth on cross-traffic, and using the Pantheon dataset [31], which provides measurements from a real-world setting.

4.1.1 Evaluation in controlled setting. For the evaluation in the controlled setting, we use a topology that includes a single bottleneck link, with the various network parameters (bandwidth, propagation delay, etc.) being drawn from the India cellular profiles seen in the Pantheon dataset. Each simulation run lasts 60 seconds and includes up to 5 cross-traffic flows, a mix of TCP Cubic and TCP Vegas connections (e.g., the 5 randomly selected cross-traffic flows may consist of 3 TCP Cubic flows and 2 TCP Vegas flows) that start and end at randomly chosen times.

Figure 2(a) shows the CDF (over 200 runs) of the normalized error, which is the ratio of the root mean square error (over 1 second windows) between the ground truth and the estimated cross-traffic and the bottleneck bandwidth. We see that the median error is under 0.04 (4% of the bottleneck bandwidth) and the 90th percentile is under 0.11 (11%), which points to the accuracy of iBox's non-reactive cross-traffic model. Figure 2(b) depicts an example (corresponding to the median in the CDF) of the ground truth of cross-traffic and the non-reactive cross-traffic model inferred by iBox. We see a close match between the two.

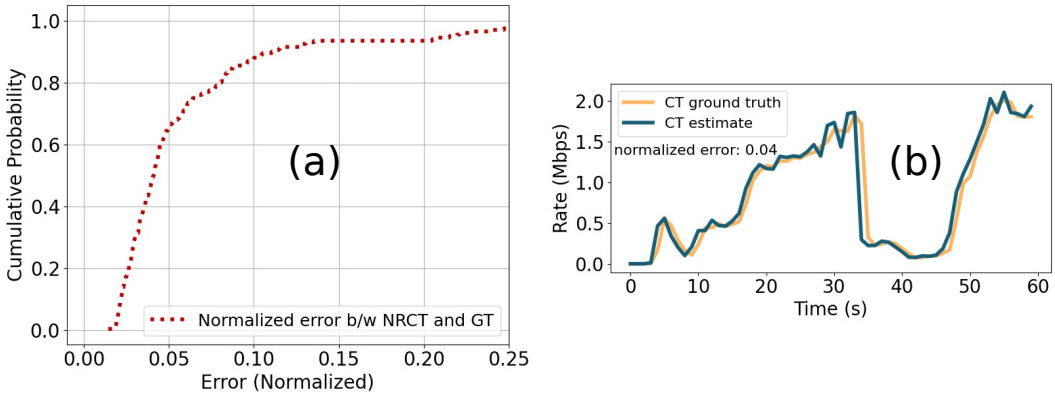

Fig. 2. **Controlled setting**: (a) Match between the non-reactive CT estimated in iBox and the ground truth (GT) cross-traffic; (b) Illustrative plot shows the estimated cross-traffic closely matches GT. Normalized error is the ratio of the RMSE between GT and the estimated cross-traffic and the bottleneck bandwidth.

4.1.2 Evaluation in real-world setting. To evaluate the non-reactive cross-traffic model in a real setting, the challenge is that the ground truth is not known. While not knowing the ground truth is often a practical constraint in real-world settings, it is still worth examining traces from such settings.

It turns out that the Pantheon dataset includes traces of multiple concurrent connections between a source-destination pair, so we can treat one as the measurement flow and the others as known cross-traffic flows. It is important to note, however, that these known flows only constitute a subset of, and hence a lower bound on, the cross-traffic; there could have been additional cross-traffic over and beyond this known subset. If iBox's estimate of cross-traffic is lower than the lower bound defined by the known cross-traffic, then clearly there has been an error in estimation ("underestimation error"). On the other hand, if iBox's estimate is greater than the lower bound ("additional CT estimated"), we cannot conclude on correctness or otherwise; in other words, "additional" does *not* mean "wrong". That said, we would not want the additional CT estimated to be unreasonably high. After all, a gross overestimate of cross-traffic would be quite useless, even if it has zero underestimation error. On the other hand, if both the underestimation error and the additional CT estimated are low, that would strongly point to an accurate estimate.

Figure 3(a) shows the CDF of the underestimation error and the additional CT estimated. To quantify the underestimation error, we consider the area between the estimated and lower bound cross-traffic curves when the former dips below the latter and normalize it by dividing by the area under the known subset of cross-traffic curve (both areas are computed for the entire experiment). We quantify the additional CT estimated similarly but by considering the area between the curves when the reverse happens, i.e., our estimate exceeds the known subset of cross-traffic.

We see from Figure 3(a) that the underestimation error is quite small, e.g., under 0.02 (2%) in over 90% of cases. Furthermore, this low level of known error is achieved while keeping the additional CT estimated within reasonable limits. For instance, additional CT is under 1 (100%) about half the time, i.e., the estimated cross-traffic includes the known cross-traffic plus an equal amount of additional cross-traffic. The additional CT is under 2.5 (250%) in 90% of the cases. These results are encouraging. Together with the high accuracy established in the controlled setting (where the ground truth on the entire CT is known), this supports the validity of the non-reactive CT (NRCT) model in iBox.

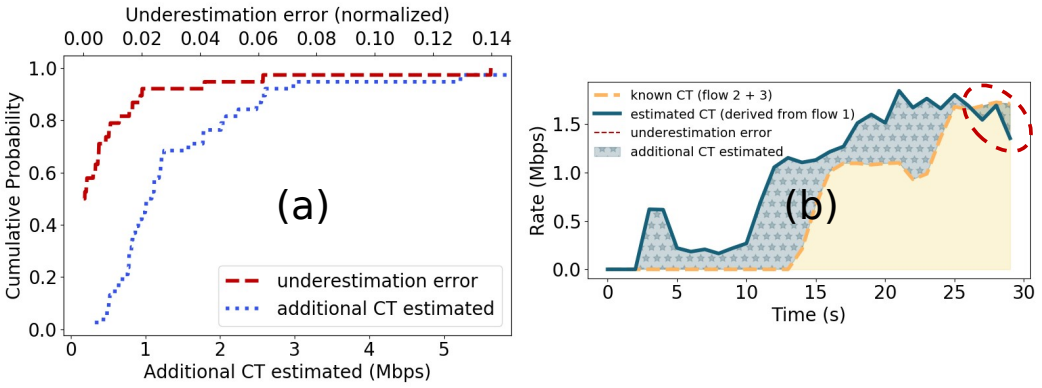

Fig. 3. **Real-world setting**: (a) CDF of "underestimation error" (top x-axis) and the "additional CT estimated" (bottom x-axis) in the Pantheon India Cellular data; (b) Illustrative plot shows how the estimated cross traffic tracks the known amount of cross traffic across time in the real-world setting.

Figure 3(b) illustrates an example, where the estimated cross-traffic is at least as high as the lower bound defined by the known subset of cross-traffic at almost all times, just as we desire. The only exception is the small region marked in red, where estimate dips below the known cross-traffic, i.e., there is an underestimation error.

5 REACTIVE CROSS-TRAFFIC

Even an accurate non-reactive model of cross-traffic does not ensure fidelity in simulation because the impact of the sender under test on the cross-traffic is not considered. To illustrate this, we consider a scenario where a sender application sends (foreground) traffic to a single-bottleneck topology, with the bottleneck set to 2 Mbps, buffer size set to 100 MTU sized packets, the propagation delay set to 50 ms, and is made to compete with TCP Cubic (background) cross-traffic which starts at t = 40s and ends at t = 300s. We consider two cases, A and B, where the sender application uses Cubic and LEDBAT as the congestion control algorithm, respectively. Figure 4 shows that the non-reactive cross-traffic estimated in the presence of a TCP Cubic foreground flow (Case A) is very accurate (i.e., it is a close match to the ground truth, so much so that the purple and green curves in the figure coincide) but is much less than the cross-traffic seen in the presence of a LEDBAT foreground flow at test time (Case B). This is not surprising since LEDBAT, being a timid background transfer protocol, cedes much more ground to cross-traffic than does Cubic at training time. So, a non-reactive model learnt with a Cubic measurement flow cannot simply be replayed in the presence of a very different protocol such as LEDBAT. To address this challenge, we present a novel approach to learning a reactive cross-traffic model, while using the non-reactive model as the starting point.

Estimating a model of cross-traffic is challenging because we do not get to make any direct observations of the cross-traffic. Furthermore, even if we are able to estimate the aggregate volume of cross-traffic over time (as in the non-reactive cross-traffic model in Section 4.1), we would still not know critical details such as the number, nature, and duration of the cross-traffic flows.

5.1 Outline

Our approach in iBox is to learn to express cross-traffic in terms of flows of "basis" protocols. The high-level idea is as follows. As in the previous sections, we model the target network path via a

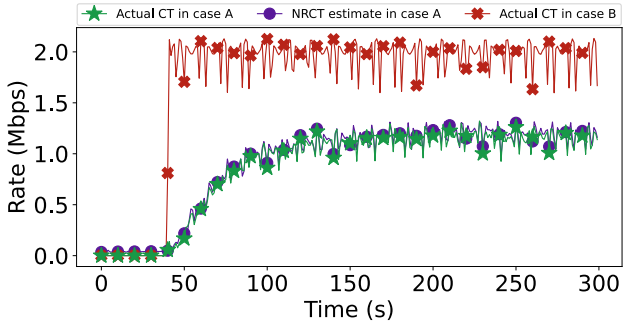

Fig. 4. Well-matched ground truth and estimated CT in the presence of a TCP Cubic flow (Case A) but poorly matched with the ground truth CT in the presence of a LEDBAT flow (Case B).

single-bottleneck bandwidth network model, with the parameters bandwidth, propagation delay and buffer size estimated from the data. In lieu of using a non-reactive estimated *cross-traffic rate time series* as in Section 4.1, we use a configuration of concurrent flows — the protocol (drawn from a set of basis protocols such as TCP Cubic), and the start and end times for each flow — to best re-create the given data by subjecting the sender protocol to *reactive* competing basis flows (which serve as a proxy for the unobserved cross-traffic) in the simulation environment.

The key challenges are in defining *what* "best" re-creation means, and *how* to achieve it. In Section 5.2, we describe a principled, efficient, optimization methodology that searches for the configuration of concurrent competing flows in order that the resulting simulation accurately matches, according to an appropriately defined metric, the recorded delay and packet loss traces for a given sender flow. In the subsequent subsections, we study the effectiveness of the proposed reactive cross-traffic flow estimation on Pantheon data. Figure 5 illustrates the iBox pipeline detailing all the stages from the parsing of raw network traces to finding cross-traffic parameters via the RCT model.

5.2 Bayesian Optimization based Search

The search for the set of basis flows to approximate the effect of the actual cross-traffic is rather expensive. After all, we need to infer the number of cross-traffic flows (up to an upper bound of k flows) and their start (τ_s) and end (τ_e) times, each of which dimensions spans a large space of choices. Furthermore, in general, the choice of each cross-traffic flow type C_{type} (for example Cubic or Vegas) is drawn from a set $\mathcal{P}$ of basis protocols.

Fitness function f: Let f denote a fitness function that captures how accurately the simulated trace (flow) matches a given ground-truth trace. There are several ways of defining f. A natural choice for f is the mean squared error between the delay time series of the ground-truth trace $\mathcal{T}$ and that obtained from simulation (in the presence of reactive cross-traffic model with given parameters) $\mathcal{T}'$. We consider the following, more robust, definition of f. We first compute 1-second average delay values in the trace, and represent a T-seconds long delay trace in $\mathcal{T}$ by T numbers, $\bar{d}_1, \bar{d}_2, \ldots, \bar{d}_T$ (i.e., averaged delays). We then define f as the root mean squared error (RMSE) in this representation, i.e.

$$f(\mathcal{T}, \mathcal{T}') := \sqrt{\frac{1}{T}\sum_{j=1}^{T}\left(\bar{d}_j - \bar{d}'_j\right)^2}. \tag{2}$$

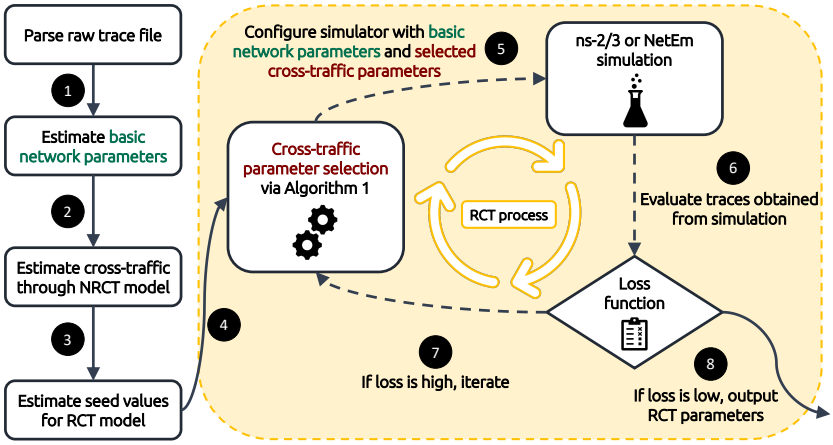

Fig. 5. **iBox pipeline**: Raw network traces are parsed from which basic network parameters (e.g., peak bandwidth) are estimated (steps 1 & 2). Next, cross-traffic is estimated using the NRCT model (step 3). Further, for more accurate cross-traffic estimation, the RCT model is used. The NRCT model's estimates help seed the RCT model's search process (step 4). The simulator is configured using the basic network parameters as well as the cross-traffic parameters as chosen by Algorithm 1. A sender application (same as which generated the raw network trace) sends traffic and is subjected to network conditions including cross-traffic competition (step 5). The resulting trace is compared against the ground-truth trace (step 6). If the two traces are not similar as per the loss function, the search continues (step 7) and new cross-traffic parameters are explored. Else, the RCT model has found the right cross-traffic parameters (step 8). The yellow region highlights the RCT process (steps 5 $\rightarrow$ 8).

Search objective: Given a ground-truth trace $\mathcal{T}$ of length T time units (i.e., the trace of a flow for which we have actual measurements), the reactive cross-traffic model estimation problem is as follows. For a given setting of RCT model parameters, let $\mathcal{T}'(.)$ denote the trace resulting from simulating a test flow in the presence of the competing cross-traffic generated by the parameter setting. The objective then is to find *optimal* RCT model parameters such that the trace $\mathcal{T}'(.)$ (in simulation) matches the ground-truth $\mathcal{T}$ (as measured) as closely as possible per the chosen fitness function f. Formally, given basis protocols $\mathcal{P}$ and trace $\mathcal{T}$, we seek:

$$\begin{aligned} \min_{C_{\text{type}}, r, \tau_s} \quad & f\Big(\mathcal{T}, \mathcal{T}'\big(\tau_s, \tau_e, C_{\text{type}}\big)\Big) \\ s.t. \quad & 0 \le \tau_s^{(i)} \le T, \quad 0 \le r^{(i)} \le 1 \quad 1 \le i \le k, \\ & \tau_e^{(i)} = \tau_s^{(i)} + (T - \tau_s^{(i)}) * r^{(i)}, \quad 1 \le i \le k, \\ & C_{\text{type}}^{(i)} \in \mathcal{P}, \quad 1 \le i \le k. \end{aligned} \tag{3}$$

This is a challenging black-box optimization problem for two reasons: (i) note that even though we know f, we cannot directly optimize it because the objective (in (3)) is to find the optimal cross-traffic *parameters*, i.e., C_{type}, τ_s, τ_e for the simulation; to even compute f, as defined in (2), we need to perform a simulation and obtain the time series $\mathcal{T}'$, which is an expensive operation, and (ii) the search space is combinatorial with both categorical and real-valued parameters.

Global black-box optimization has received a lot of attention from machine learning and optimization communities; Bayesian Optimization (BO) is a widely-used family of techniques for global optimization of arbitrary functions. We leverage a recently-developed sequential-model

based optimization (SMBO) technique for hyperparameter optimization [10] that operates within the BO framework; this technique is particularly suitable for our setting where evaluating the objective is expensive and the search space consists of both categorical and real-valued parameters.

The high-level procedure for solving the search problem (3) using SMBO is given in Algorithm 1. The objective in (3) is a complex black-box function of $C_{\text{type}}, r, \tau_s$ that is expensive to compute since it requires running a simulation; so, the optimization algorithm approximates the objective with a surrogate model which is easier to evaluate. Algorithm 1 optimizes this surrogate model over iterations [1], where it picks a point that optimizes the surrogate (or some transformation), which in turn becomes the candidate for where the objective (3) should be evaluated next. Bayesian optimization algorithms differ in (a) the criterion they optimize to obtain the next candidate, and (b) how they model the objective function given the history of observations, denoted by $\mathcal{H}$ in the Algorithm. In our implementation, we use the Tree-structured Parzen Estimator approach (referred to as TPE in the Algorithm) from [19] as the surrogate model (details are outside the scope of this work, see [9, 10]).

Algorithm 1: Reactive cross traffic estimation

Result: Solution to Problem (3)
Input: $k, f, \mathcal{T}, \mathcal{P}$, MaxIter, $\mathbf{x}_0 := \left\{C_{\text{type}}^{(i)}, \tau_s^{(i)}, \tau_e^{(i)}\right\}_{i=1}^{k}$;
$\mathcal{T}' \leftarrow$ Run ns with initial cross-traffic parameters $\mathbf{x}_0$;
Initialize $\mathcal{H} \leftarrow \left(\mathbf{x}_0, f(\mathcal{T}, \mathcal{T}')\right)$;
while $1 \leq j \leq$ MaxIter **do**
  $\mathbf{x}_j := \left\{C_{\text{type}}^{(i)}, \tau_s^{(i)}, \tau_e^{(i)}\right\}_{i=1}^{k} \leftarrow TPE(\mathcal{H})$;
  $\mathcal{T}' \leftarrow$ Run ns with cross-traffic parameters $\mathbf{x}_j$;
  Compute $f_j \leftarrow f(\mathcal{T}, \mathcal{T}')$;
  $\mathcal{H} \leftarrow \mathcal{H} \cup \left(\mathbf{x}_j, f_j\right)$;
end
return $\mathbf{x}_{j^*} \in \mathcal{H}$ where $j^* = \arg\min_{j \in \mathcal{H}} f_j$;

Choice of k: Note that the problem formulation allows *up to* k cross-traffic flows, including 0, because $\tau_s^{(i)} = \tau_e^{(i)}$ is admissible, which amounts to one less flow. So, it suffices to set a maximum value for k in the optimization; we use $k = 3$ by default but explore other settings of k in Section 6.3.

Basis protocol for cross-traffic model: While our framework is general, in this paper, by default we pick TCP Cubic alone as the default basis protocol, i.e., we seek to model the cross-traffic as a set of zero or more Cubic flows, starting and ending at various times. Cubic is the most widely used transport protocol in the Internet, and the default flavour of TCP on both Linux and Microsoft Windows. Therefore, we expect that expressing the cross-traffic in terms of a set of Cubic flows would result in a good fit, while limiting the search space. However, in Sections 6.4 and 6.5, we evaluate RCT with additional basis protocols (BBR and LEDBAT), besides Cubic.

Seeding the Bayesian Optimization: To aid the optimization efficiency, we utilize the non-reactive cross-traffic model (which, as discussed in Section 4.1, can be computed quite efficiently), to seed the search procedure (this initial setting is denoted by $\mathbf{x}_0$ in Algorithm 1). Specifically: (1) We first estimate the volume of the non-reactive cross-traffic given a trace $\mathcal{T}$ using the ideas described in Section 4.1. (2) We then employ a simple change point detection technique on the

[1] MaxIter (maximum number of iterations) = 100 in our experiments.

estimated cross-traffic volume series to find the times when there is a sudden increase or decrease in cross-traffic volume (by at least a threshold δ), signaling the start (or end) of new (or ongoing) cross-traffic flows. We start with a conservative δ (i.e., indicating a change point) and get the initial values $\tau_s^{(1)}, \tau_e^{(1)}$ for the first (presumed) cross-traffic flow; then we progressively use smaller values of δ to get estimates of further presumed cross-traffic flows $\tau_s^{(i)}, \tau_e^{(i)}$, until either δ is too small or $i = k$, i.e., we have exhausted the budget of cross-traffic flows.

5.3 Impact of Bayesian Optimization

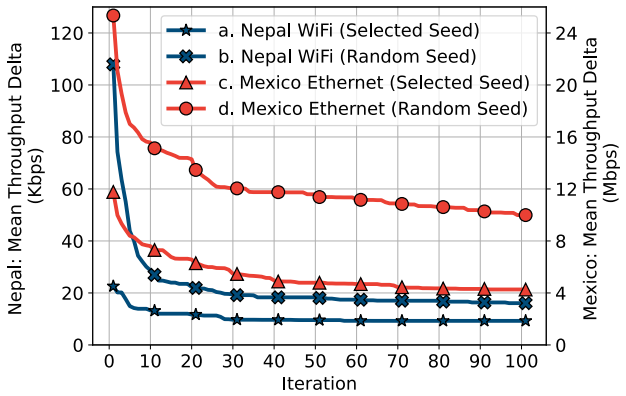

Fig. 6. Convergence of Algorithm 1, with(out) seeding where MaxIter = 100.

We evaluate the effectiveness of the Bayesian optimization procedure described in Algorithm 1. Figure 6 shows drop in the mean discrepancy in throughput (i.e., the difference between the ground truth throughput and the throughput measured with the presumed reactive cross-traffic model) as the Bayesian Optimization proceeds. The mean discrepancy is computed over 43 Cubic traces (Nepal WiFi) and 50 Cubic traces (Mexico Ethernet). For instance, in the Nepal WiFi case, while the throughput discrepancy starts out at more than 100 Kbps in the "random seed" case (relative to an average throughput of 570 Kbps for these connections), it drops sharply to 20 Kbps after about 40 iterations. When the optimization is seeded with estimates obtained from the NRCT model as described above, there is a significant reduction in the throughput discrepancy even as the optimization begins, and it drops to about 10 Kbps in 40 iterations (i.e., a discrepancy of 10/570 = 1.76% relative to the mean), as compared to 20 Kbps in the "random seed" case. The importance of seed selection in enabling a convergence to a better local optimum is well established. We see similar trends play out in the Mexico Ethernet case too (where the mean throughput was 67 Mbps, to put the throughput discrepancy (5.97%) in context). These results underscore the promise of the Bayesian Optimization approach together with our seeding technique for learning a reactive cross-traffic model.

5.4 Comparison with Cross-Traffic Ground Truth

We compare the inferred reactive cross-traffic model with the ground truth in a controlled setting where cross-traffic is known and therefore we are able to perform a microscopic comparison. Figure 7 shows two examples where the ground truth comprised 3 TCP Cubic cross-traffic flows, starting and ending at various times. In Figure 7(left), the reactive cross-traffic model inferred is quite accurate, both in terms of the number of cross-traffic flows inferred (3) and their start and end

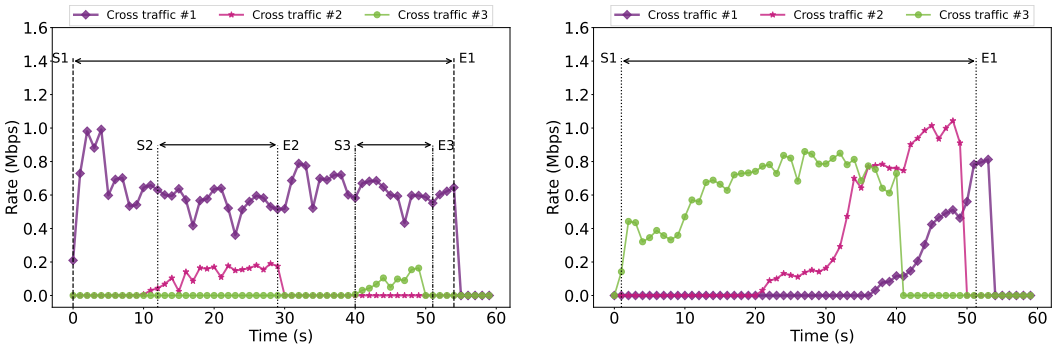

Fig. 7. Illustrative examples showing the result of Bayesian optimization (Algorithm 1) after 100 iterations, with TCP Cubic alone as a basis protocol. Vertical lines indicate the start and end times of the inferred Cubic cross-traffic flows, which closely matches the ground truth on the left. However, on the right, the algorithm decides to use only 1 long running Cubic flow to model an ensemble of 3 actual cross-traffic flows.

times. However, in Figure 7(right), the seeding procedure discussed early in the section identifies 2 cross traffic flows with (start,end) times as (0,53) and (32,40). Bayesian optimization, however, collapses these into a single flow in (0,51). The overlapping nature of the cross-traffic flows in the ground truth and the fact that one of them (flow 3) ends when another (flow 1) picks up suggests that it is hard to tease apart the individual flows. Arguably such teasing apart is less important so long as the overall impact of cross-traffic is captured adequately.

6 EXPERIMENTAL EVALUATION

We compare our reactive cross-traffic estimation technique against the non-reactive model discussed in Section 4 and a baseline where no cross traffic is used. For this study, we use Pantheon China Cellular and Colombia Ethernet data. In the former case (China), we build iBox models based on 51 TCP Cubic foreground traces from July 2017 to June 2018, while in the latter case (Colombia), we build models based on 50 TCP Cubic foreground traces from June 2017 to Feb 2020. That is, we use the measurements derived from these Cubic traces to drive Bayesian optimization and learn a cross-traffic model in terms of the basis flows.

Thereafter, we test using two different foreground protocols — TCP BBR (for China) and TCP Vegas (for Colombia) — using iBox-based simulation under 3 conditions: "No CT" (only basic network parameters but no modeling of cross-traffic), "NRCT" (cross-traffic model in the form of a UDP stream replayed per the non-reactive model from Section 4), and "RCT" (the reactive model where we learn the start and stop times for up to $k = 3$ presumed TCP Cubic cross-traffic flows; we explore other values of k in Section 6.3 and additional basis protocols in Sections 6.4 and 6.5).

We compare these iBox-based results with the "ground truth" obtained from actually running the test protocol on real network paths (i.e., BBR in China and Vegas in Colombia), which is also available from the Pantheon data set. Note that, in general, the cross-traffic encountered during the "ground truth" runs of the test protocols would be different from that during the runs of Cubic, which we use to derive our cross-traffic model. Nevertheless, if the latter model implanted in iBox can help obtain results matching the "ground truth", that would underscore the value of the iBox cross-traffic model.

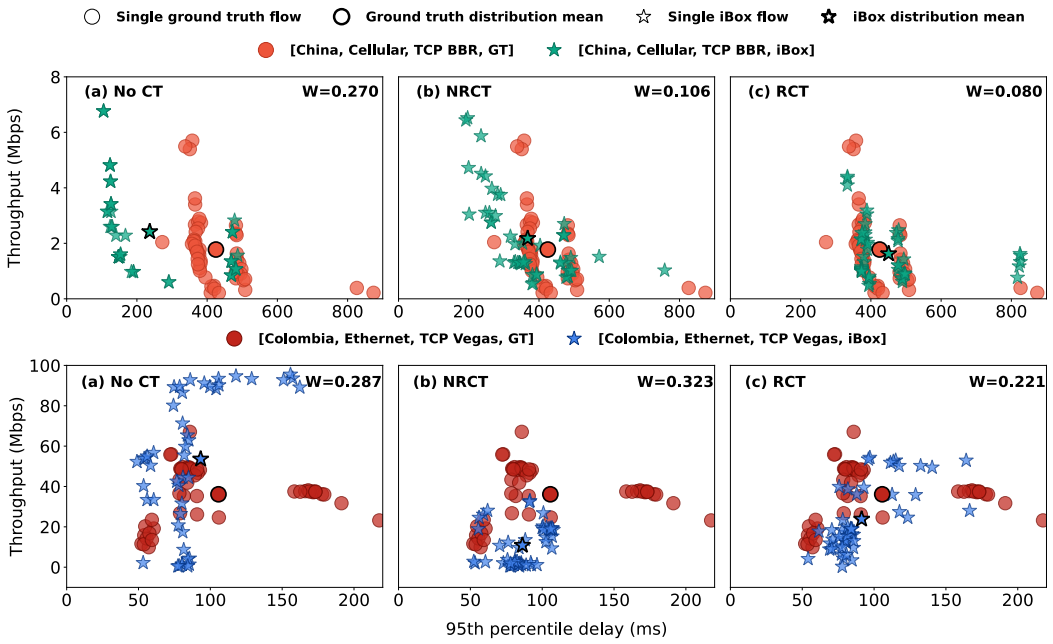

Fig. 8. Distribution of P95 delay and throughput for TCP BBR and Vegas simulations of iBox models trained with China Cellular (top) and Colombia Ethernet (bottom) data, respectively, with (a) no cross-traffic (No CT), (b) non-reactive cross-traffic (NRCT), and (c) reactive cross-traffic (RCT). Each dot represents an individual connection, with circles corresponding to the ground truth and stars to the iBox simulation. (Figure format patterned after [40].) 2D-Wasserstein distance (W), a measure of distance of the 2D (delay, throughput) distribution vis-a-vis the ground truth, is also shown.

p95 Delay	GT	No CT	NRCT	RCT
mean	425.03	236.51 (44%)	367.60 (14%)	**450.46** (6%)
p75	482.03	468.45 (3%)	470.06 (2.5%)	**478.46** (0.7%)
p50	379.37	147.72 (61%)	353.11 (7%)	**390.31** (2.9%)
p25	368.24	126.11 (66%)	279.11 (24%)	**373.29** (1.4%)

Table 2. Results for TCP BBR (all numbers are in ms) on Pantheon (China Cellular) traces.

Throughput	GT	No CT	NRCT	RCT
mean	1.78	2.42 (36%)	2.17 (22%)	**1.62** (9%)
p75	2.32	3.15 (36%)	2.87 (24%)	**2.14** (7.8%)
p50	1.63	**1.60** (1.8%)	1.36 (17%)	1.21 (26%)
p25	0.90	1.35 (50%)	1.26 (40%)	**0.98** (8.9%)

Table 3. Results for TCP BBR (all numbers are in Mbps) on Pantheon (China Cellular) traces.

6.1 China Cellular Data

For the China data, Tables 2 and 3 show that not modeling any cross-traffic produces a poor match with respect to the ground truth, with a significant underestimate of p95 delay and a slight

overestimate of throughput, which is not unexpected given the absence of competition from cross-traffic. Introducing NRCT yields some improvement on account of increased competition offered to the BBR test flow.

With RCT, there is an excellent match in p95 delay with respect to the ground truth over the entire distribution. Note that unlike NRCT, RCT would tend to push and fill the buffer (because it comprises TCP Cubic flows), thereby offering greater competition to the BBR test flow. It is also noteworthy that the TCP Cubic based RCT model works well though the nature of the actual cross-traffic is unknown. It also suggests that iBox with such RCT can accurately recreate performance even in absolute terms.

We see a similar benefit of RCT if we consider throughput even though less pronounced as shown in Table 3. Figure 8 (top) shows the combination of p95 delay and throughput for these 3 settings of cross traffic. There is progressive improvement in the match, both visually and in terms of 2D Wasserstein distance, a widely used measure of distance between two probability distributions — in this case, between the 2D (delay, throughput) distribution corresponding to the iBox recreation of the BBR flows and the ground truth, as we go from No CT to NRCT and on to RCT. This metric (normalized to the range [0,1]) measures the minimum cost incurred in morphing one distribution to the other [34].

6.2 Colombia Ethernet Data

We observe a similar trend in the case of the Colombia (Ethernet) data from Figure 8 (bottom). First, we observe that in the No CT case there is a gross overestimation of throughput (e.g., >30% flows attain more than 80 Mbps throughput in the No CT simulation whereas almost all ground truth flows are under 60 Mbps); in fact, the overestimation is much more severe than in the China case. The reason again is the lack of any competition from cross traffic, especially with plentiful bandwidth on offer for the sender in this Ethernet setting. As competition is offered through NRCT and RCT (middle and right plots, respectively), the match improves as expected and in particular, we observe that the RCT case achieves the best match both in terms of throughput as well as the delay distribution. Note that a non-trivial fraction of simulated calls in the RCT case achieve a p95 delay of >120 ms which is characteristic of the ground truth (red circles) presumably because cross traffic flows fill the buffers and increase the queuing delay.

6.3 Impact of k (upper bound on the number of reactive flows considered)

Thus far, the maximum number, k, of reactive cross-traffic flows in the RCT model was fixed at 3. In Figure 9, we sweep through $k = 1$ to 5 in the [China, Cellular, BBR] case from Figure 8 (top). While there is an improved match (both visually and in terms of the Wasserstein distance) between the results obtained from iBox and the ground truth as k goes up to 3, there is no benefit from larger values of k. We see a similar trend in the [Colombia, Ethernet, Vegas] case too (not shown). This supports our choice of $k = 3$ by default, as larger k would only increase the complexity of Bayesian optimization (Section 5.2) without any benefit.

6.4 Impact of multiple basis protocols in Pantheon setting

Next, we consider expanding the set of basis protocols for RCT from just Cubic alone to also include BBR and LEDBAT. Therefore, in addition to searching through the space of cross-traffic flow count and start/end times, the Bayesian optimization also searches over the multiple protocols in the basis set — {Cubic, BBR, LEDBAT} — and combinations of these.

However, as we see in the throughput versus 95th percentile delay plots in Figure 10, expanding the basis set does not yield a visible improvement in the match with respect to the ground truth compared to when the basis set was {Cubic} alone. Indeed, even the 2D Wasserstein distance between

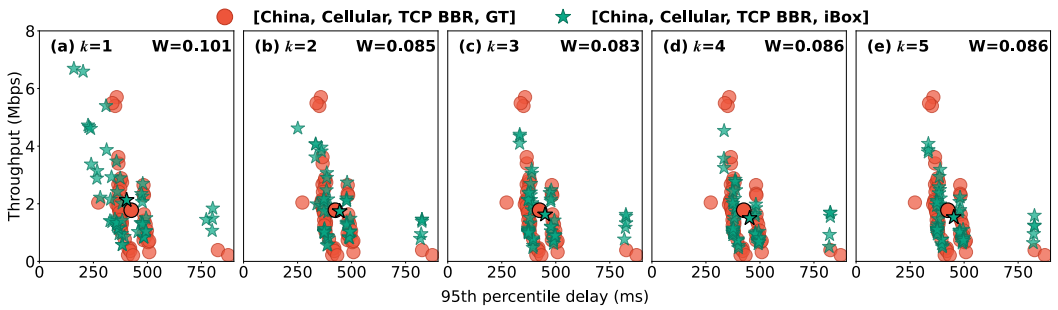

Fig. 9. Exploration of the impact of k (the maximum number of RCT flows assumed) corresponding to the [China, Cellular, TCP BBR] case (Figure 8(top), where the RCT model had used $k = 3$).

the ground truth and the simulation shows little improvement: 0.084 with {Cubic}, 0.076 with {Cubic, BBR}, and 0.083 with {Cubic, BBR, LEDBAT}. We believe the reason is that the dominance of Cubic on the Internet (at least, as it was when the Pantheon data set was collected) means that Cubic alone is adequate for modeling the cross-traffic.

6.5 Impact of multiple basis protocols in a controlled setting

While the above results shows that Cubic, with its dominance on the Internet, alone suffices as the cross-traffic basis in the context of the Pantheon data set, it is nevertheless useful to evaluate the ability of iBox's RCT model to accommodate multiple basis protocols. To this end, we generated 150 simulated traces in ns-3 corresponding to each of Cubic and LEDBAT as the foreground protocols and ground truth cross-traffic comprising either only Cubic, only LEDBAT, or a combination of the two, 50 traces each. We then evaluate how well iBox with RCT using a basis set of {Cubic}, {LEDBAT}, or {Cubic, LEDBAT} helps recreate the results in terms of throughput, delay, and packet loss metrics.

Figure 11 shows the results, where, as before, each circle and star corresponds to an individual connection for ground truth and iBox with RCT, respectively (the mean is shown with a dark border). We see that with {LEDBAT} as the basis set, iBox successfully matches those connections where LEDBAT was the only cross-traffic (purple cluster) but is unable to match those connections where

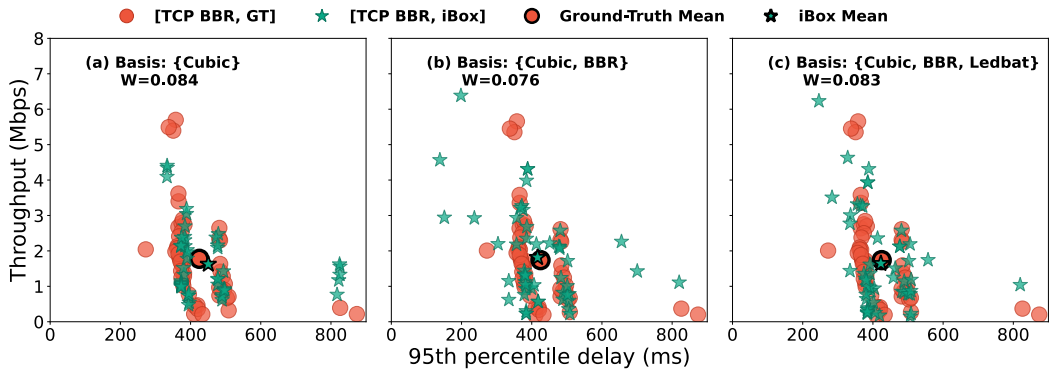

Fig. 10. Exploration of the impact of multiple basis protocols (Cubic, BBR, LEDBAT) on the distribution of p95 delay and throughput for iBox models corresponding to China Cellular data set with $k = 3$.

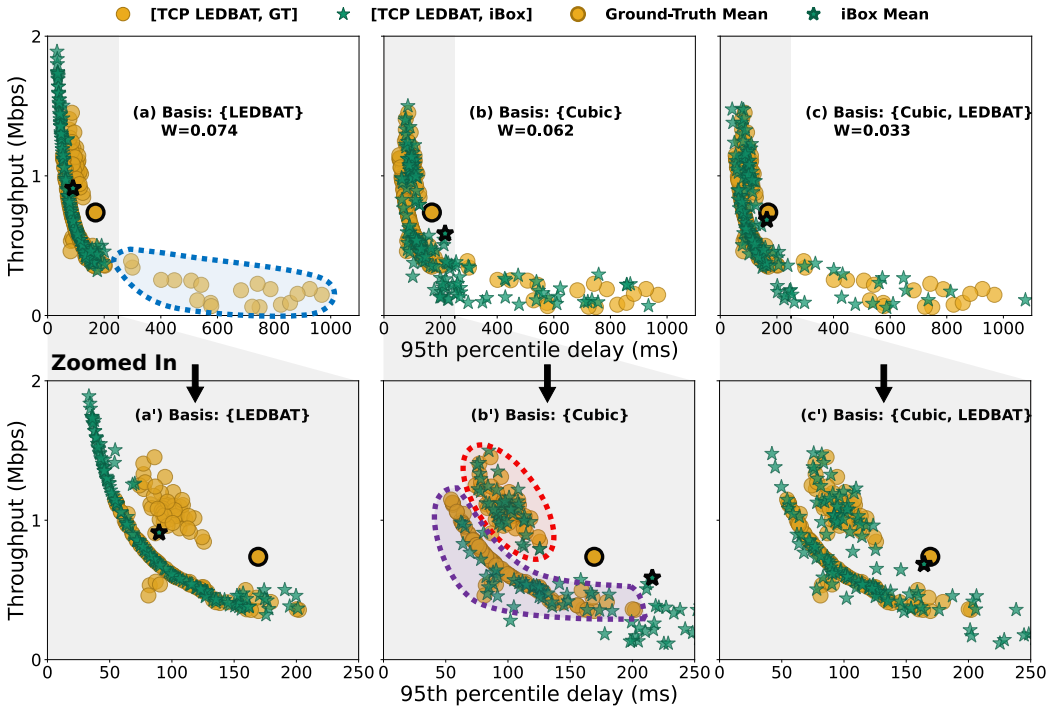

Fig. 11. Top row depicts experiments on synthetic data where the foreground protocol was LEDBAT, the underlying cross-traffic comprised a mix of Cubic and LEDBAT flows and the basis sets were constructed with different combinations (columns 1-3) of basis protocols (Cubic, LEDBAT). The purple cluster corresponds to ground truth flows where cross-traffic comprised only LEDBAT, whereas the red and blue clusters correspond to ground truth flows which also had Cubic as part of the cross-traffic set (for clarity, we only mark these clusters in one plot each though these are present in all six plots). The bottom row is a zoomed-in version in the 0-250 ms delay range.

		1D Wasserstein Distance		
Protocol	Basis set	Throughput	p95 delay	Packet Loss
LEDBAT	LEDBAT	0.095	0.085	0.015
	Cubic	0.109	0.056	0.015
	Cubic, LEDBAT	**0.051**	**0.027**	**0.003**
Cubic	LEDBAT	0.055	**0.001**	0.036
	Cubic	0.080	**0.001**	0.030
	Cubic, LEDBAT	**0.034**	**0.001**	**0.019**

Table 4. Evaluation of iBox, using reactive cross-traffic estimates, on synthetic data with multiple basis protocols. The numbers are the 1D Wasserstein distance (normalized between 0 and 1) between the ground-truth and the simulated distributions of the metric values. The best match for each metric is in boldface.

Cubic was also part of the cross-traffic (red and blue clusters) and especially those connections where Cubic cross-traffic produced large delays (blue cluster). On the other hand, while {Cubic} does better, it is not able to match the cross-traffic behaviour well in the cases where the actual

Network type	Location	Protocol	1D Wasserstein Distance		
			Throughput	p95 Delay	Packet Loss
Ethernet	China	Cubic	0.164	0.065	0.072
		Vegas	0.162	0.064	0.104
		BBR	0.237	0.089	0.149
Ethernet	Colombia	Cubic	0.349	0.115	0.209
		Vegas	0.192	0.105	0.105
		BBR	0.233	0.109	0.122
Cellular	China	Cubic	0.084	0.116	0.153
		Vegas	0.049	0.173	0.165
		BBR	0.053	0.050	0.217
Cellular	India	Cubic	0.024	0.073	0.037
		Vegas	0.157	0.114	0.022
		BBR	0.066	0.058	0.072
Cellular	US	Cubic	0.031	0.083	0.083
		Vegas	0.065	0.183	0.061
		BBR	0.108	0.224	0.121
WiFi	Nepal	Cubic	0.041	0.107	0.079
		Vegas	0.106	0.036	0.183

Table 5. Evaluation of iBox (using reactive cross-traffic estimates) on Pantheon data.

cross-traffic comprised LEDBAT alone. The match is best when the basis set is {Cubic, LEDBAT}, providing the Bayesian optimization procedure the opportunity to pick an appropriate mix of the basis protocols. The zoomed-in view on the bottom row in Figure 11 highlights how when a combination of both protocols is used in the basis set {Cubic, LEDBAT}, both red and purple clusters are matched quite well.

Table 4 quantifies the comparison in terms of the 1D Wasserstein distance for the various metrics, again underscoring the ability of our Bayesian optimization procedure to pick out the best match with a mix of basis protocols.

6.6 Overall Results

We evaluate iBox with reactive cross-traffic estimates on a suite of datasets comprising various protocols, countries, and network types (listed in Table 1) obtained from the Pantheon testbed. For each network type and location, we train the iBox model using (*only*) TCP Cubic traces, but evaluate on TCP Vegas and BBR, in addition to Cubic itself. We consider the distributions (over traces in a given dataset) of throughput, P95 delay, and packet loss rates; and compute the 1D Wasserstein distance — in this case, between the ground-truth and the simulation separately for the throughput, P95 delay, and loss rate metrics.

The results are presented in Table 5. First, we observe that most values are close to 0, indicating a good match between the distributions. Second, within each dataset, we find that the (normalized) distance obtained on the test protocols (Vegas, BBR) closely matches that obtained on the training (i.e., TCP Cubic) protocol, across metrics. These observations indicate that iBox faithfully replicates results across a range of countries, network types and test protocols. The encouraging results with iBox arise because iBox models infer the underlying network configurations rather than seeking to learn the input-to-output mapping. While the latter would tend to be protocol specific, the former is inherent to the target network conditions which allows for generalization to new test protocols.

Metric	GT	Pantheon	iBox
Throughput (Mbps)	3.93	3.15 (19.0%)	**3.88 (1.0%)**
Packet Loss (%)	3.48	6.34 (82.0%)	**3.61 (3.7%)**
P95 Delay (ms)	261.20	291.50 (11.6%)	**240.00 (8.1%)**

Table 6. Comparison with the Pantheon calibrated emulator on the Nepal Ethernet traces.

6.7 Comparison with Pantheon Emulator

The Pantheon calibrated emulator [40] was trained on some of the same data we use and more. Therefore, a head-to-head comparison would be interesting. One challenge, however, is that the code used to calibrate the Pantheon emulator is not available [20]. Consequently, our analysis focuses on a set of Ethernet traces from the Pantheon node in Nepal (Table 1) for which the authors of [40] have made available the calibrated emulator parameters derived by them.

The iBox model is trained using data from TCP Cubic. On the other hand, the Pantheon calibrated emulator is trained using data from a basket of protocols, as explained in Section 3. We compare the throughput, loss rate, and delay for a test protocol, TCP Vegas, using both models. From Table 6, we see that iBox yields a much closer match to the ground truth than the Pantheon calibrated emulator (percentage error shown in parentheses). This underscores the benefit of careful domain knowledge-based parameter estimation as compared to Pantheon, which uses a stochastic packet loss mechanism with a single parameter to indirectly capture some of the effects of cross traffic.

7 IMPLEMENTATION

We integrate iBox functionality into ns-2 [28], ns-3 [29], and NetEm [25], enabling iBox to automatically set the network topology via the parameters learnt from network traces. To enable cross-traffic recreation, in the NRCT case, a variable bit-rate traffic generator is used to send packets into the bottleneck as per the estimated trace (the pacing timer is triggered once every 1000 microseconds to ensure that the bitrate is maintained in a smooth manner) while in the RCT case, TCP senders are configured to start-up and stop according to the parameters learnt using Algorithm 1. It is straightforward to integrate iBox with other simulation/emulation frameworks.

For NRCT, the execution time depends upon the number of packets in the ground truth trace; but the computation is straightforward and is extremely fast (under 150 milliseconds for a trace with 200,000 packets) as it needs only a single pass through the time-series.

On the other hand, for RCT, the Bayesian optimization search is run for each trace to find the right cross traffic configuration. Each iteration involves picking a new set of parameters and running an ns simulation corresponding to the chosen setting. We ran our Bayesian optimization on a 16-core Intel(R) Xeon(R) CPU E5-2673 instance. For a trace comprising around 200,000 packets, an iteration (primarily bottlenecked on the ns simulation) usually takes under 10 seconds. Therefore, running the 100 iterations typically needed for convergence requires up to 1000 seconds per trace. Of course, for a dataset comprising multiple traces, the RCT model parameters corresponding to each trace can be learnt in parallel. For a typical 50-trace dataset in Table 1, we can learn the RCT model parameters for all the traces in under an hour.

In scenarios where compute is limited or it is known that the foreground protocol during the train and test setting is the same or of a similar kind, we can use the cheaper NRCT model to obtain cross-traffic estimates (as we can expect the background cross-traffic present during the train setting not to react any differently to the foreground traffic in the test setting). Conversely, if computation time is not a concern or the cross-traffic is required to react to differences in the behaviour of the foreground protocol in the test setting (e.g., a newer variant of the foreground

protocol with a slight modification or a completely different foreground protocol), then cross-traffic estimates from the more expensive RCT model should be used.

8 RELATED WORK

Traditional network simulation [28–30, 33] has focused on recreating the mechanisms of network elements such as links, queues, wireless contention, etc. While great attention is paid to the detailed workings of network elements, the task of configuring these and the workload is typically left to the user. Our work is complementary; it focuses on end-to-end behaviour rather than on the innards of the network and uses data to configure the simulator to recreate the appropriate end-to-end behaviour. Hence, we focus on prior work aimed at recreating end-to-end behaviour.

Trace-driven Network Simulation: This uses data to build a simple model of the network.

Early work in the mobile network setting [27] used ping traces to infer the network bandwidth, delay, and loss rate over a sliding window. At inference time, these are then replayed. While being simple, this approach does not model the impact of self-congestion (e.g., a buffer filling up and dropping packets because of an aggressive sender) or the impact of cross-traffic.

Mahimahi [24, 26] emulates the network by gleaning the link transmission opportunities from a packet trace, in addition to emulating a user-configurable propagation delay and a stochastic packet loss model. However, the transmission opportunities are inherently tied to the nature of flow that the trace was derived from; note that although certain settings such as cellular links provide isolation across nodes through proportional fair scheduling, interference by and on cross-traffic at the node of interest itself cannot be avoided. As such, the transmission opportunities seen in the training trace might not carry over to emulating very different flows (e.g., a sender that is much less or more aggressive).

Cellsim [39] operates similarly, using a stream of UDP packets to saturate the link. Doing so runs the risk of beating down or even entirely squelching cross-traffic to the same node in a cellular setting and even to other nodes in settings such as WiFi where there is no inter-node isolation.

iBox avoids this limitation by taking a fundamentally different approach: explicitly inferring a reactive cross-traffic model. That said, we view our approach as complementary to that of Mahimahi and Cellsim. In other words, it might be useful to combine iBox's cross-traffic model with Mahimahi's and Cellsim's model of variable bandwidth.

iBox's approach is similar to the Pantheon calibrated emulator [40], which also learns a simple network model, parameterized by the bottleneck bandwidth, buffer size, propagation delay, etc., that fits the observed traces. However, there are notable differences. First, [40] uses packet traces from a basket of TCP protocol variants for training to learn a best-fit network model, which is then tested with the same basket of protocol variants. In contrast, iBox uses TCP Cubic traces alone for training (which is appropriate given its dominance and hence the relative ease of obtaining Cubic traces) but then is able to effectively simulate the network path even for the testing of new, previously unseen protocols. Second, while [40] uses Bayesian optimization for learning all model parameters, iBox employs this relatively expensive procedure only for learning the reactive cross-traffic model. And even the learning of iBox's reactive cross-traffic model through Bayesian optimization is made significantly more efficient by working with a popular protocol, such as TCP Cubic, which allows leveraging the ns implementation of the protocol for rapid iterations, avoiding much slower emulation as might be needed with a basket of unsupported protocols as in [40]. Third, in contrast to iBox, [40] does not model cross-traffic, which is quite key, as shown by our results.

Time Series Modelling and Forecasting: Time series techniques have been employed in many domains for data-driven modelling and forecasting: finance [8, 13], weather [17, 21, 38], retail [6], and more. These techniques span a wide range: auto-regressive (AR), moving average (MA), combinations of AR and MA (Chapter 2, [32]), recurrent neural networks (RNN) [16], and

generative adversarial networks (GAN) [15]. Such models enable both prediction and long-range forecasting.

There has been work on applying such time series techniques to network data. Harpoon [36] focuses on macroscopic flow-level metrics while fs [37] considers packet-level metrics. [22] uses LSTMs for link-level traffic volume prediction. DoppelGANger [23] uses GANs to capture both correlations in various networked time series and correlations between the time series and metadata (e.g., the ISP where the trace was collected).

From the perspective of this paper, such application of time series techniques to networking suffers from two limitations. First, these do not consider the sending behaviour, which is typically driven by a control loop with exogenous input (e.g., receiver feedback). Such control loop dynamics have a direct bearing on the evolution of a flow in terms of packet delay, packet loss, etc. Second, the more sophisticated deep learning based techniques tend to be computationally expensive and so are not suitable for fast simulation or real-time emulation.

9 DISCUSSION

We discuss some opportunities for extending iBox.

First, despite the promising results even in such networks as cellular, the iBox model comprising a single-bottleneck, FIFO queue, and fixed bandwidth would have to be augmented to accommodate more complex and dynamic environments (e.g., high mobility scenarios). For instance, the bottleneck bandwidth could be estimated as a time series instead of as a fixed value. Furthermore, we could employ ML-based black-box optimization techniques (such as Bayesian optimization) to jointly learn even the basic parameters such as bottleneck bandwidth and buffer size, which would be more robust, and likely more accurate, than estimating these individually based on domain knowledge.

Second, we only rely on passive estimation of the iBox model parameters based on existing traces, which help avoid any imposition or overhead on the network and existing applications. However, in general, we could augment such data with selective active probing (e.g., packet-pair probing, to address the limitations pointed out in Section A.1). Such active probing could also be targeted at "novel" paths, i.e., ones with characteristics quite different from the profiles already included in iBox.

10 CONCLUSION

We have presented iBox, a data-driven network path simulator that seeks to recreate the end-to-end behaviour of network paths. iBox reduces the network path into a simple single-bottleneck model and estimates its parameters based on end-to-end packet traces. A key part of iBox is the cross-traffic model, for which we present both a non-reactive model and a reactive model. Our evaluation using data from the Pantheon testbed and also controlled experiments is promising.

ACKNOWLEDGEMENTS

We thank our shepherd, Derek Eager, and the anonymous reviewers for their insightful feedback. We also thank Vasiliy Novikov for helping us integrate iBox functionality into NetEm. Finally, we are grateful to P. Brighten Godfrey, Saikat Guha, Radhika Mittal, Akshay Nambi, and Ramachandran Ramjee for their valuable feedback on earlier drafts of the paper.

REFERENCES

[1] LEDBAT. https://en.wikipedia.org/wiki/LEDBAT.
[2] iBox project website. https://aka.ms/ibox.
[3] TCP BBR congestion control comes to GCP – your Internet just got faster. https://cloud.google.com/blog/products/networking/tcp-bbr-congestion-control-comes-to-gcp-your-internet-just-got-faster.
[4] TCP CUBIC. https://en.wikipedia.org/wiki/CUBIC_TCP.

[5] uTorrent Transport Protocol. https://www.bittorrent.org/beps/bep_0029.html.

[6] I. Alon, M. Qi, and R. J. Sadowski. Forecasting aggregate retail sales:: a comparison of artificial neural networks and traditional methods. *Journal of retailing and consumer services*, 8(3):147–156, 2001.

[7] S. Ashok, S. S. Duvvuri, N. Natarajan, V. N. Padmanabhan, S. Sellamanickam, and J. Gehrke. iBox: Internet in a Box. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, HotNets '20, page 23–29, New York, NY, USA, 2020. Association for Computing Machinery.

[8] W. Bao, J. Yue, and Y. Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944, 2017.

[9] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for Hyper-Parameter Optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[10] J. Bergstra, D. Yamins, and D. D. Cox. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. ICML'13, page I–115–I–123. JMLR.org, 2013.

[11] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, SIGCOMM '94, page 24–35, New York, NY, USA, 1994. Association for Computing Machinery.

[12] Y. Cao, A. Jain, K. Sharma, A. Balasubramanian, and A. Gandhi. When to use and when not to use bbr: An empirical analysis and evaluation study. In *Proceedings of the Internet Measurement Conference*, IMC '19, page 130–136, New York, NY, USA, 2019. Association for Computing Machinery.

[13] M. P. Clements, P. H. Franses, and N. R. Swanson. Forecasting economic and financial time-series with non-linear models. *International Journal of Forecasting*, 20(2):169–183, 2004.

[14] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions On Networking*, 12(6):963–977, 2004.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[16] S. Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, 1(1):17–61, 1988.

[17] A. Grover, A. Kapoor, and E. Horvitz. A Deep Hybrid Model for Weather Forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 379–386, New York, NY, USA, 2015. Association for Computing Machinery.

[18] S. Ha, I. Rhee, and L. Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, jul 2008.

[19] Hyperopt. http://hyperopt.github.io/hyperopt.

[20] About the calibrated emulators. https://groups.google.com/g/pantheon-stanford/c/sbiP6OAN1NY/m/MmPL9l6mAQAJ.

[21] R. J. Kuligowski and A. P. Barros. Localized Precipitation Forecasts from a Numerical Weather Prediction Model Using Artificial Neural Networks. *Weather and forecasting*, 13(4):1194–1204, 1998.

[22] A. Lazaris and V. K. Prasanna. Deep Learning Models For Aggregated Network Traffic Prediction. In *15th International Conference on Network and Service Management (CNSM)*, 2019.

[23] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 464–483, New York, NY, USA, 2020. Association for Computing Machinery.

[24] Mahimahi. http://mahimahi.mit.edu/.

[25] Network Emulator. https://man7.org/linux/man-pages/man8/tc-netem.8.html.

[26] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan. Mahimahi: Accurate Record-and-Replay for HTTP. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, pages 417–429, Santa Clara, CA, 2015.

[27] B. D. Noble, M. Satyanarayanan, G. T. Nguyen, and R. H. Katz. Trace-Based Mobile Network Emulation. In *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '97, page 51–61, New York, NY, USA, 1997. Association for Computing Machinery.

[28] ns-2 Network Simulator. https://www.isi.edu/nsnam/ns.

[29] ns-3 Network Simulator. https://www.nsnam.org/.

[30] OPNET Technologies. https://www.riverbed.com/in/products/steelcentral/opnet.html.

[31] Pantheon: The Training Ground for Internet Congestion Control Research. https://pantheon.stanford.edu/.

[32] R. Prado and M. West. *Time series: modeling, computation, and inference*. CRC Press, 2010.

[33] QualNet: Network Simulation. https://www.scalable-networks.com/qualnet-network-simulation.

[34] Y. Rubner, C. Tomasi, and L. J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, 40(2):99–121, Nov 2000.

[35] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind. Low Extra Delay Background Transport (LEDBAT). RFC 6817, IETF, Dec. 2012.
[36] J. Sommers and P. Barford. Self-Configuring Network Traffic Generation. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, page 68–81, New York, NY, USA, 2004. Association for Computing Machinery.
[37] J. Sommers, R. Bowden, B. Eriksson, P. Barford, M. Roughan, and N. Duffield. Efficient network-wide flow record generation. In *2011 Proceedings IEEE INFOCOM*, pages 2363–2371, 2011.
[38] C. Voyant, M. Muselli, C. Paoli, and M.-L. Nivet. Numerical weather prediction (NWP) and hybrid ARMA/ANN model to predict global radiation. *Energy*, 39(1):341–355, 2012.
[39] K. Winstein, A. Sivaraman, and H. Balakrishnan. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 459–471, Lombard, IL, Apr. 2013. USENIX Association.
[40] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein. Pantheon: the training ground for internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 731–743, 2018.

APPENDIX

A ESTIMATING BASIC NETWORK PARAMETERS

We begin by discussing the estimation of basic network parameters in the iBox model: the bottleneck bandwidth, the buffer size, and the propagation delay. We also discuss techniques to filter the raw measurements to home in on the subset that is more reliable. In the specific context of RCT, the estimates obtained through these heuristics, even if an approximation, can be used to effectively seed the Bayesian optimisation based search.

A.1 Bottleneck Bandwidth (b)

Packet pair based estimation [14] is the traditional method for estimating the bottleneck bandwidth. However, this is challenging to employ in various network settings such as non-FIFO queuing and variable bandwidth, as in a cellular setting. Furthermore, the absence of "naturally occuring" packet pairs in a trace (e.g., TCP transfer) would be a limiting factor. Indeed, Figure 12 shows that the packet-pair estimate of peak bandwidth is almost always under 1 Mbps whereas even the connection-level throughput stretches to nearly 5 Mbps.

Since the iBox model approximates such links with a simple FIFO model and a fixed bandwidth, it is important that the estimation of the bottleneck bandwidth is done in a manner that is robust to departures from this simplified model in the short run. We compute the receive rate over windows of 100 ms and pick the 95th percentile as our estimate of the bottleneck bandwidth, b. So long as the sender fills the pipe at least 5% of the time, we would be able to estimate the bottleneck bandwidth accurately. A protocol such as TCP Cubic, the most widely used transport protocol in the Internet, tends to fill the pipe and therefore facilitates estimation of the bottleneck bandwidth. Figure 12 shows that the iBox estimate of peak bandwidth (corresponding to the fastest 100 ms window) is, as expected, somewhat higher than the connection-level throughput (over the entire connection duration, 30s in our setting).

A.2 Propagation Delay (d)

To estimate the propagation delay, we would ideally like end-to-end delay measurements where the other components of delay — the transmission delay and the queuing delay — are minimized. The transmission delay tends to be small relative to the propagation delay (since it is the transmission time of a single packet) and moreover can be estimated based on b and factored out. So minimizing the queuing delay is our focus.

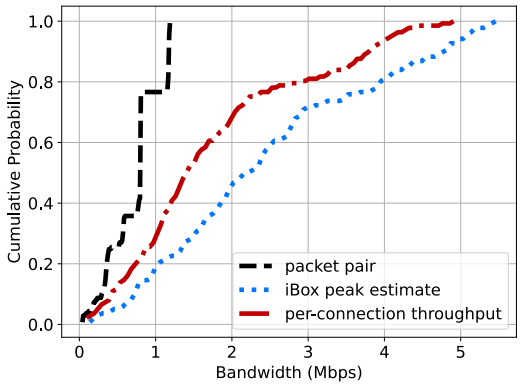

Fig. 12. Estimated peak bandwidth obtained using packet-pair and iBox techniques, compared with the connection-level throughput distribution

In general, considering the minimum over the delay samples from a stream of packets (e.g., those in a TCP connection) would help minimize the queuing delay. However, in a congested setting, it is possible that even picking the minimum delay might not help ensure that the measurement corresponds to an empty or near empty queue, resulting in an overestimation of the propagation delay. Furthermore, some packets taking an alternate, shorter path could result in a spurious estimation of the minimum delay.

Therefore, our approach is to focus on and test for the consequences of an empty queue: (1) the receiving rate would be lower than the bottleneck bandwidth estimated above. Indeed, if the queue remains empty for a long enough period, the receiving rate would likely drop to zero for a buffer-filling protocol such as Cubic; and (2) the receiving rate (measured at the receiver) would be roughly the same as the sending rate (observed at the sender).

Note that conditions (1) and (2) individually do not imply that the queue is empty. For instance, a queue could be non-empty and still condition (1) could be satisfied, say because (unobserved) cross-traffic consumes a fraction of the bottleneck bandwidth. Likewise, condition (2) could be satisfied when the queue is non-empty but in a balanced state, i.e., the inflow (corresponding to the sending rate) is equal to the outflow (corresponding to the receiving rate). However, conditions (1) and (2) being satisfied together very likely implies an empty queue.

Therefore, we filter out windows that do not satisfy (1) or (2). From the packets that remain, we pick out mode of the delay distribution as our estimate of the propagation delay, d. We find that this filtering technique produces a steady estimate of the propagation delay that is consistent across multiple connections between the same end-points (not shown).

A.3 Buffer Size (B)

The buffer size can be estimated as the bottleneck bandwidth (the b estimated above) times the difference between the end-to-end delay with a full buffer and that with an empty buffer. The latter delay corresponds to the propagation delay (d) estimated above. The former (full buffer delay) is what we need to estimate.

Just as picking the minimum delay does not necessarily correspond to an empty buffer, picking the maximum delay does not correspond to a full buffer. Therefore, we again consider the consequence of a full buffer, which is the likelihood of consequent packet drop. Accordingly, we consider the packets right before and right after the one that was dropped. We check to see if the sender-side

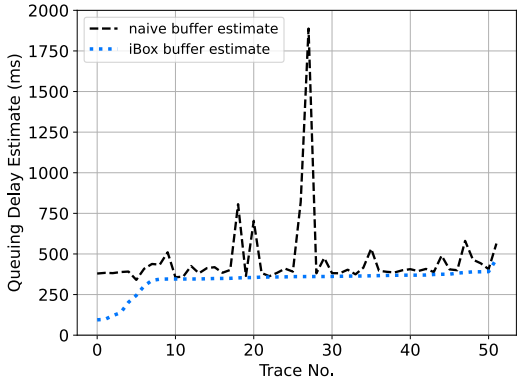

Fig. 13. Estimated full-queue delay (in ms) obtained using the naive and the iBox filtered techniques.

spacing between these packets and the dropped packet is under a threshold (5th percentile of all inter-packet spacings). Among all the packets in this filtered set, we pick the mode of the distribution as our estimate of the delay corresponding to a full buffer.

Figure 13 compares the estimate of the full-queue delay obtained with the iBox filtering outlined above and the raw maximum delay. We see that the former yields a consistent estimate (of about 375 ms) for the most part, whereas the latter spikes up to as high as 1800 ms, underscoring the benefit of iBox filtering.

Received October 2021; revised December 2021; accepted January 2022