TCP Congestion Signatures

Srikanth Sundaresan
Princeton University
srikanths@princeton.edu

Amogh Dhamdhere
CAIDA/UCSD
amogh@caida.org

Mark Allman
ICSI
allman@icir.org

kc claffy
CAIDA/UCSD
kc@caida.org

ABSTRACT

We develop and validate Internet path measurement techniques to distinguish congestion experienced when a flow self-induces congestion in the path from when a flow is affected by an already congested path. One application of this technique is for speed tests, when the user is affected by congestion either in the last mile or in an interconnect link. This difference is important because in the latter case, the user is constrained by their service plan (*i.e.*, what they are paying for), and in the former case, they are constrained by forces outside of their control. We exploit TCP congestion control dynamics to distinguish these cases for Internet paths that are predominantly TCP traffic. In TCP terms, we re-articulate the question: was a TCP flow bottlenecked by an already congested (possibly interconnect) link, or did it induce congestion in an otherwise idle (possibly a last-mile) link?

TCP congestion control affects the round-trip time (RTT) of packets within the flow (*i.e.*, the *flow RTT*): an endpoint sends packets at higher throughput, increasing the occupancy of the bottleneck buffer, thereby increasing the RTT of packets in the flow. We show that two simple, statistical metrics derived from the flow RTT during the slow start period—its coefficient of variation, and the normalized difference between the maximum and minimum RTT—can robustly identify which type of congestion the flow encounters. We use extensive controlled experiments to demonstrate that our technique works with up to 90% accuracy. We also evaluate our techniques using two unique real-world datasets of TCP throughput measurements using Measurement Lab data and the Ark platform. We find up to 99% accuracy in detecting self-induced congestion, and up to 85% accuracy in detecting external congestion. Our results can benefit regulators of interconnection markets, content providers trying to improve customer service, and users trying to understand whether poor performance is something they can fix by upgrading their service tier.

CCS CONCEPTS

• **Networks** → **Network measurement**;

IMC '17, Internet Measurement Conference, November 1–3, 2017, London, United Kingdom

ACM ISBN 978-1-4503-5118-8/17/11…$15.00
https://doi.org/https://doi.org/10.1145/3131365.3131381

KEYWORDS

Internet congestion, Throughput, TCP

ACM Reference Format:
Srikanth Sundaresan, Amogh Dhamdhere, Mark Allman, and kc claffy. 2017. TCP Congestion Signatures. In *Proceedings of IMC '17, November 1–3, 2017, London, United Kingdom, Internet Measurement Conference,* 14 pages.
https://doi.org/https://doi.org/10.1145/3131365.3131381

1 INTRODUCTION

Exploding demand for high-bandwidth content such as video streaming, combined with growing concentration of content among a few content distribution networks [8, 13–15, 26, 27, 50] —some large and sophisticated enough to adjust loading, and therefore congestion levels on interconnection links [18, 28]—has resulted in lengthy peering disputes among access ISPs, content providers, transit providers that center on who should pay for the installation of new capacity to handle demand. As a result, there is growing interest in better understanding the extent and scope of congestion induced by persistently unresolved peering disputes, and its impact on consumers. But this understanding requires a capability that the Internet measurement community has not yet provided in a usable form: the ability to discern interconnection congestion from the congestion that naturally occurs when a last mile link is filled to capacity. Implementing such a capability would help a variety of stakeholders. Users would understand more about what limits the performance they experience, content providers could design better solutions to alleviate the effects of congestion, and regulators of the peering marketplace could rule out consideration of issues where customers are limited by their own contracted service plan.

Although a large body of work has focused on locating the bottleneck link and characterizing the impact of loss and latency on TCP performance [23, 30, 32–34, 43, 48, 51, 52], they doesn't inform us about the type of congestion. Recent attempts to use coarse network tomography to identify interconnect congestion [36], also have shortcomings [49]. To the best of our knowledge, there is no technique that can reliably identify whether a flow is bottlenecked by an initially unconstrained path (that it fills up) or whether it was bottlenecked by an already congested path, without having *a priori* knowledge about the path, *i.e.*, the capacity of its bottleneck link and the traffic profile of the link. Such a technique would differentiate between, for example, a flow that is bottlenecked by the last-mile access link versus one that is bottlenecked by a congested interconnect link. In this paper, we identify distinctive signatures in flow RTT during the TCP slow start period that can reliably distinguish these two scenarios.

Our technique exploits the effect of the bottleneck link buffer on flow RTT. *Flow RTT* is the RTT of packets within a TCP flow, which can be computed using sequence and acknowledgment numbers within the packets. When a TCP flow starts on an otherwise uncongested path, it drives buffering behavior in the bottleneck link by increasing its occupancy. On the other hand, when the flow starts in a path that is already congested, flow RTT is dominated by buffering in the congested link. We identify two parameters based on flow RTT during TCP slow start that we use to distinguish these two cases—the coefficient of variation of flow RTT, and the normalized difference between the maximum and minimum RTT. We use these two parameters, which can be easily estimated for TCP flows, to build a simple decision tree-based classifier.

We validate the classifier using both a controlled testbed as well as real-world data. We build a testbed to conduct extensive controlled experiments emulating various conditions of access and peering link bottlenecks, and show that classifier achieves a high level of accuracy, with up to 90% precision and recall. We then apply our techniques on two real-world datasets. Our first dataset consists of throughput test data collected by the M-Lab infrastructure [38], specifically Network Diagnostic Test (NDT) measurements from January through April of 2014 [3, 4]. This timeframe spanned the discovery and resolution of an interconnect congestion event between Cogent (a major transit provider) and large access ISPs in the US. Data in January and February showed a marked drop in throughput during peak hours compared to off-peak data in March and April, after Cogent resolved the issue. We use this episode to label the dataset—peak hour traffic in January and February as interconnect congested and off-peak traffic in March and April as access-link congested—and find that the decision tree classifier allows us to classify these flows accurately. Because the flows are coarsely labeled (we do not have ground truth data about the clients that ran the throughput test, and therefore resort to blanket labeling based on month and time-of-day), we conduct a more focused experiment, running throughput periodic tests between a single host and a single M-Lab server between February and April 2017. We choose these hosts based on evidence we found of occasional interconnection congestion in the path between them using the Time Series Latency Probing (TSLP) [35] method; we also know the service plan rate of the client. In this experiment, our decision tree classifier detected interconnect congestion with an accuracy of 75-85% and access-link congestion with an accuracy of 99%. Our false negatives in this experiment mostly occur with higher throughput (but not enough to saturate the access link), and lower interconnect buffer latency, which suggest a legitimate gray zone when it is not clear what type of congestion occurred.

Our proposed technique has two important advantages: it provides per-flow diagnosis, and relies only on the flow itself without needing out-of-band probing. Out-of-band probing can introduce confounding factors such as load balancing, and differential servicing of probe packets. Our technique can supplement existing efforts to understand broadband performance, such as the FCC Measuring Broadband America, to not just understand what throughput users get, but also the role of the ISP infrastructure and the interconnect infrastructure in the throughput they achieve.

The rest of the paper is structured as follows. We develop the intuition behind our technique in § 2, and validate the intuition by building a model and testing it using controlled experiments in our testbed in § 3. We describe how we use and label M-Lab data during a 2014 peering congestion event, and how we conduct a more focused experiment using M-Lab in 2017 in § 4, and how we validate our model using these datasets in § 5. We then discuss the limitations of our model in § 6. We describe related literature in § 7, and conclude in § 8.

2 TCP CONGESTION SIGNATURES

We are interested in the scenario where we have a view of the flow from the server, but no knowledge about the path or link capacities. This scenario is common for speed test providers such as M-Lab NDT, or Ookla's Speedtest [40]. These tests inform the user about the instantaneous capacity of the path between the user and the speed test server is, but not whether the capacity is limited by the access link (*i.e.*, the user's ISP service plan). In this section we develop the intuition behind our technique to detect the nature of congestion in TCP flows. We first define the two types of congestion events we are interested in, and then describe how we build a model based on TCP flow RTT that can distinguish them.

2.1 Self-induced *vs* External Congestion

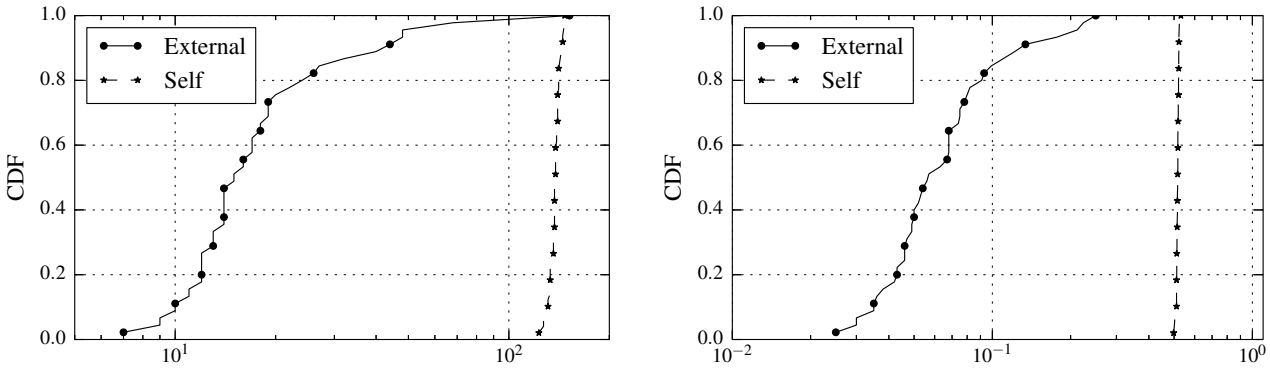
We refer to the link with the smallest available capacity on the path between a server and client as the *capacity bottleneck link*. Further, we say a link is "congested" when traffic is being buffered at the head end of the link (*i.e.*, the traffic load is greater than the available link capacity).

Self-induced congestion occurs when a TCP flow starts in an otherwise uncongested path, and is able to saturate the capacity bottleneck link. In other words, self-induced congestion occurs when *a flow's throughput is limited by the capacity bottleneck link and the flow itself drives buffer occupancy at the head of the bottleneck link*. An example of such congestion is when a speed test saturates the client's access link.

External congestion occurs when a TCP flow starts in a path with an already congested link. In this case, the *available capacity* on the bottleneck link is essentially zero because the link is congested.[1] In terms of buffer behavior, the new flow has little impact on buffer occupancy because external traffic was congesting the link before the new flow started. For example, a speed test that is bottlenecked by an already congested non-edge link, say an interconnect link, and which therefore is unable to saturate the client access link, experiences external congestion.

Flows could fail to saturate the path bottleneck link for other reasons, *e.g.*, high latency, random loss, low application demand, or small window sizes. Such flows do not experience congestion, and existing techniques [10, 52] can detect such factors limiting TCP throughput; we do not consider them in this paper.

[1] Note: Because Internet traffic is elastic the new flow will ultimately utilize some of the capacity of the bottleneck link because other flows will backoff.



(a) Max - min RTT (ms) of packets during slow start

(b) Coefficient of variation of RTT of packets during slow start

Figure 1: RTT signatures for self-induced and external congestion events. Self-induced congestion causes a larger difference between the minimum and maximum RTT, and also larger variation in RTT during the slow start period. For illustrative purposes, we show data from from one set of experiments using a 20 Mbps emulated access link with a 100 ms buffer, served by a 1 Gbps link with a 50 ms buffer. The access link has zero loss and 20 ms added latency.

2.2 Challenges in Identifying the Type of Congestion

The server point of view has several advantages, the most important being that it has direct information about outgoing packets and TCP state. However, even with a detailed view of the flow, distinguishing between the two types of congestion that we list above is challenging. Some techniques include analyzing the flow throughput, TCP states [52], and/or flow packet arrivals [34, 48] or RTT. Each has its advantages and drawbacks.

Information about flow throughput is insufficient to determine the type of congestion unless we also know the actual service plan of the client. For example, if we see a 9 Mbps flow, the type of congestion event it encountered depends on whether the service plan was, say, 10 Mbps (likely self-induced), or 20 Mbps (likely external). However, typically, only the user and their ISP know the service plan rates, and external throughput test services such as M-Lab or Ookla do not have access to it. Additionally, available service plans in the U.S. vary across a wide range of throughputs, from less than 10 Mbps to exceeding 100 Mbps. Therefore achieved throughput, even with associated parameters such as congestion window size, is not a useful indicator of type of congestion.

TCP state analysis [52] can help us analyze TCP state transitions and flow behavior; however it does not help us differentiate between different kinds of congestion. Transitions to/from the fast retransmit or the retransmission timeout state can potentially tell us about congestion events. However, in practice, we found it difficult to parameterize and model these state changes. Simple techniques such as modeling the total number of fast retransmit and timeout states per time interval or the time to the first retransmit state have the same difficulty that it varies according to the path latency, service plan of the client, loss-rate, and cross-traffic, which are difficult to systematically account for in controlled settings. Ideally, we would prefer parameters that are robust across a range of settings.

Previous work has also used packet arrival patterns to uncover a congested path [34, 48]. Such techniques typically have the requirement that they be downstream of the point of congestion to be able to measure packet arrival rate. That is not possible with the server point of view, nor from clients unless they have access to network packets. Though packet spacing can be approximated by analyzing ACK arrival patterns, ACKs can be noisy, and cannot tell us any more than that the flow encountered congestion.

Flow RTT, particularly at a per-packet granularity, contains information about the condition of the underlying path. In particular, the RTTs of packets in a flow allow us to distinguish between an empty bottleneck buffer (increasing RTT as the flow fills up the buffer) and a busy buffer (RTT is relatively stable as it is dominated by the added latency due to an already full buffer). We use these properties of the RTT to build our model, and it relies only on one essential component of the path, the buffer, and therefore yields robust results in our controlled testbed that translate well to the real world. Flow RTTs are useful only during the slow start period, but fortunately this short interval is sufficient for us to be able to distinguish the two congestion states. We now describe the intuition behind this technique in more detail.

2.3 Using Flow RTT to Distinguish Congestion Type

- *Self-induced congestion:* The buffer at the head of the bottleneck link is empty when the flow starts. As the flow scales up, this buffer fills up, causing an increase in the flow RTT—the RTT measured towards the end of slow start will be significantly higher than the RTT measured at the beginning. The difference depends on the size of the buffer. Such a scenario typically happens in last-mile networks where the capacity of the link between the endpoint and the provider network is considerably smaller than backbone or interconnect links.

- *External congestion:* In this case the flow starts on a path containing a link that is already congested, meaning that the available capacity of that link is low and the buffer is already full or close to full, causing packets to queue. The state of this link reduces the ability of the flow to scale throughput before it encounters loss. The existing buffer occupancy increases the baseline latency of the path, and at the same time reduces variation, because the new flow's impact on the state of the buffer is much smaller than in the case of self-induced congestion.

We run two simple experiments over an emulated 20 Mbps "access link" served by a 1 Gbps "interconnect" link to illustrate the two cases. Figure 1a shows the CDF of the difference between the maximum and minimum RTT during the slow start phase. We define slow start as the period up to the first retransmission or fast retransmission. We see that the difference when the congestion is self-induced is roughly 100 ms, which is the size of the access link buffer that we emulate. This is what we expect, because this buffer fills up when the flow self-induces congestion. In the case of the external congestion, the difference is much smaller, because the flow encounters congestion at the 1 Gbps link. This congestion becomes part of the baseline RTT for the flow packets, leaving a smaller difference between the maximum and the minimum. The coefficient of variation of the RTT measurements (Figure 1b) also shows a similar pattern: the variation is smaller for external congestion than it is for self-induced congestion, because the impact of the buffer on the RTT is lower for the former case.

We use this phenomenon to distinguish the two cases. Our intuition hints at multiple potential metrics that one could use to measure the evolution of RTT: *e.g.*we could track the growth of the RTT to see if it increases monotonically. However, we decide on two simple metrics that are easy to compute from the flow RTT samples:

(1) *Normalized difference between the maximum and minimum RTT during slow start (NormDiff):* We measure the difference between the maximum and minimum RTT during slow start and normalize it by the maximum RTT. This metric measures the effect of the flow on the buffer—it gives us the size of the buffer that the flow fills—without being affected by the baseline RTT; a flow that fills up the buffer will have a higher value than one that encounters a full buffer.

(2) *The coefficient of variation of RTT samples during slow start (CoV):* This metric is the standard deviation of RTT samples during slow start normalized by the average. This metric measures the smoothing effect of the buffer on RTT while minimizing the effect of the baseline RTT. A flow that experiences self-induced congestion will see higher values of the CoV, because the RTT increases as the buffer fills up. The RTT for externally congested flows will be dominated by an already full buffer, and so the CoV will be lower.

Together, these metrics are robust to a wide range of buffer sizes. Although there are corner cases where the model could fail, particularly in case of highly occupied, but not fully congested buffers, we note that the notion of congestion becomes fuzzy in those cases anyway. We use these metrics to build a standard, simple decision tree that can accurately classify the two congestion events. We restrict our RTT samples to the first slow-start period when TCP's behavior is more predictable because it starts from zero—the RTT, and the path buffer state are at baseline. The path and its congestion characteristics are less likely to change during this period than over the (longer) course of the entire flow lifetime. Our experiments show that the throughput achieved during slow start is not always indicative of the throughput achieved during the lifetime of the flow, but it is indicative of the capacity of the bottleneck link during a self-induced congestion event. Therefore, our techniques are also useful as a starting point to estimate the link capacity, particularly in cases where the flow throughput changes during the course of the flow.

3 CONTROLLED EXPERIMENTS

We describe the custom testbed we use to run controlled experiment that emulate the type of flows we want to classify. We use these flows to build our decision tree classifier.

3.1 Experiment Setup

We run throughput tests between a client and a server in a local network connected via two links that we shape to effect the two kinds of congestion that we discuss in § 2; self-induced, and external. Our testbed (Figure 2) can emulate a wide range of last-mile and core network conditions.

Testbed hardware

The testbed consists of two Raspberry Pi 2 devices, two Linksys WRT1900AC routers, a combination of Gigabit and 100M Ethernet links, and various servers on the Internet. We use the Raspberry Pis and the Linksys devices as end- and control- points for test traffic. The Pis have a quad-core 900 MHz ARM7 processor, 1 GByte RAM, and a 100 Mbit NIC. The routers have a dual-core 1300 MHz ARM7 processor, 128 Mbytes RAM, and a 1 Gbps NIC. The testbed is physically located at the San Diego Supercomputing Center in San Diego, CA.

Emulating core and last-mile network links

In Figure 2, AccessLink emulates representative access-link conditions: we use `tc` with a token-bucket with a 5 KByte burst filter to set its bandwidth to 10 Mbps, 20 Mbps, and 50 Mbps, loss to 0.02% and 0.05%, and latencies to 20 ms and 40 ms, with jitter set to 2 ms. We also utilize three buffer sizes for AccessLink: approximately 20 ms, 50 ms, and 100 ms; the first setting is on the lower end of buffer size for last-mile networks, while the last setting is lower than the maximum buffer we have seen. For example, the buffer sizes in three homes that we tested on were approximately 25 ms, 45 ms, and 180 ms. We use low buffer values to test the limits of our hypothesis: the larger the buffer, the more likely it is that our hypothesis will work.

InterConnectLink, connecting Router 1 and Router 2 in the figure, emulates an interdomain link at 950 Mbps with a 50 ms buffer (we shape it to 950 Mbps, slightly less than its 1 Gbps capacity in order to ensure that our experiments utilize the buffer). We do not add latency or loss to this link, though the buffer could naturally induce latency and loss when it is occupied.

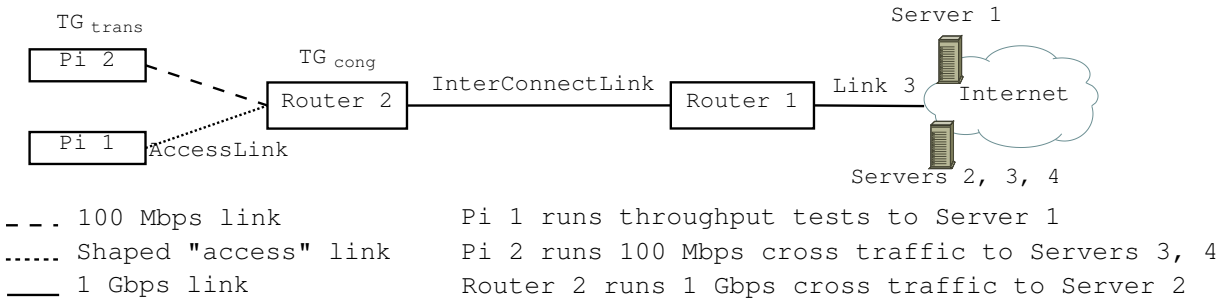


Figure 2: Experimental testbed. Router 2 connects to Router 1 using InterConnectLink, and Router 1 connects to a university network using Link 3. Both InterConnectLink and Link 3 have a capacity of 1 Gbps. Pi 1 and Pi 2 connect to Router 2 over a 100 Mbps link, which is limited by the Pi NIC. We emulate access links using AccessLink and a shaper on Router 2. We emulate an interdomain link using InterConnectLink and a shaper on Router 1.

We acknowledge the difficulty in getting precise numbers for the networks we are emulating, but we believe our settings capture a wide range of real-world access networks.

Emulating cross-traffic and congestion
We use two kinds of cross-traffic generators that we built ourselves to emulate real networks. The first traffic generator, TG_{trans}, written in Go [47] runs on Pi 2 and fetches files over HTTP from Servers 2 and 3 using a random process. These servers are located at the International Computer Science Institute in Berkeley, CA, and the Georgia Institute of Technology in Atlanta, GA, 20 ms and 60 ms away respectively. The generator fetches objects of size 10KB, 100KB, 1MB, 10 MB, and 100 MB, with the fetch frequency for an object inversely proportional to its size. Since TG_{trans} bypasses AccessLink, and can only generate a maximum demand of 100 Mbps (due to the Pi NIC limitation), it does not congest InterConnectLink. However, it provides transient cross-traffic on InterConnectLink which introduces natural variation; we run TG_{trans} during all our experiments.

The second traffic generator, TG_{cong}, runs on Router 2, and is a simple `bash` script that fetches a 100 MB file from Server 4 (which is less than 2 ms away) repeatedly using 100 concurrent `curl` processes. TG_{cong} emulates interdomain link bottlenecks by saturating InterConnectLink (capacity 950 Mbps); we run TG_{cong} for experiments that require external congestion.

Throughput experiments
We use `netperf` to run 10-second downstream throughput tests from Server 1 to Pi 1. We capture packet traces on Server 1 using `tcpdump` for each test, which we use for analysis. We run two types of experiments. First, we run `netperf` without congesting InterConnectLink, but with transient cross-traffic using TG_{trans}. This yields data for flows with self-induced congestion, because `netperf` saturates AccessLink, our emulated access link. We then run `netperf` along with both cross-traffic generators. The second cross-traffic generator, TG_{cong}, saturates InterConnectLink, which now becomes the the bottleneck link in the path. This scenario emulates a path with external congestion. For each throughput, latency, and loss combination, we run 50 download throughput tests.

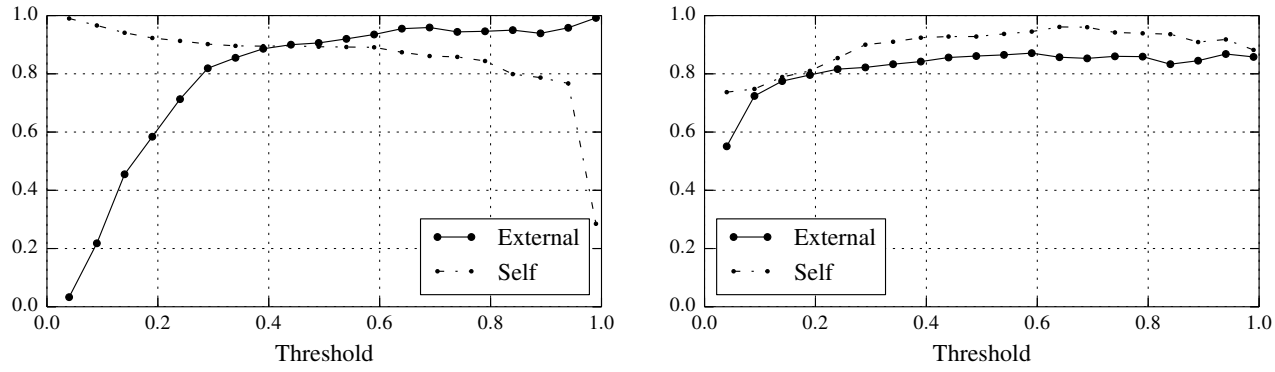
What constitutes access-link congestion?
There is no fixed notion of what constitutes acceptable throughput as a fraction of link capacity; however, we would expect it to be close to 1. This *congestion threshold* is important for us to label our test data as incurring self-induced congestion or external congestion. We therefore do not set the threshold arbitrarily: study the impact of a range of values of this threshold on our model and the classification of congestion, and show that our results are robust to a range of reasonable threshold values.

Labeling the test data
We use the congestion threshold for labeling the test data. We label the throughput tests that achieve throughput greater than this threshold during the slow start phase as self-induced congestion. For example, if we set AccessLink throughput in the testbed to 20 Mbps, and the threshold to 0.8, then we label flows that experience a slow-start throughput of greater than 16 Mbps as experiencing self-induced congestion. We do not use just the cross-traffic information to label the data, because inherent variability in the testbed result in some tests not achieving the access link throughput even if there is no external congestion, and vice-versa (some tests achieve access link throughput even when we are running both TG_{cong} and TG_{trans}; this could be because of transient issues such as some cross-traffic process threads restarting, and other TCP interactions, particularly when our emulated access link throughputs are low). However, these form a small fraction of the tests, and we filter these out, and we label the remaining data as externally congested.

3.2 Analysis and Model

We extract the RTT features from the Server 1 packet traces. We use `tshark` to obtain the first instance of a retransmission or a fast-retransmission, which signals the end of slow start. We then collect all downstream RTT samples up to this point; an RTT sample is computed using a downstream data packet and its corresponding ACK at the server. For statistical validity, we discard flows that have fewer than 10 RTT samples during slow-start. We compute NormDiff and CoV using these samples.

Building and Tuning the Decision-tree Classifier We use the `python sklearn` library implementation [42] to automatically build the decision tree classifier [46] using the NormDiff and the



(a) Prediction precision **(b) Prediction recall**

Figure 3: Model performance: we see that precision and recall are high for a wide range of threshold values.

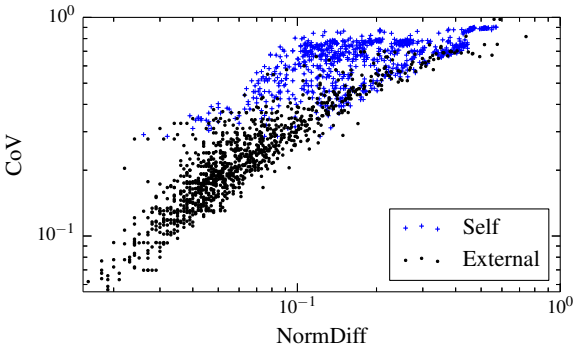


Figure 4: Raw NormDiff and CoV metrics for our controlled experiments. We see that both metrics are useful to separate the two types of congestion events.

CoV RTT parameters for classification. Our classifier has two tuneable parameters: the depth of the tree, and the threshold we use for estimating whether the flow experienced access-link congestion.

- **Tree-depth** The tree depth for any decision tree classifier has to strike a balance between building a good model and overfitting for the test data. Since we only have two input parameters to the decision tree, and two output classes, we keep the tree simple. We evaluate tree depths between 3 and 5. We get high accuracy and low false-positives with all three depths. For the rest of the paper, we use a tree depth of 4.
- **Congestion Threshold** Since the congestion threshold determines how we label the test data, it has a direct impact on the classifier. A threshold that is too high, *e.g.*, close to 1, risks mislabeling flows that self-induce congestion as externally congested, because we will label even flows that achieve a large fraction of capacity as externally congested. Similarly, a threshold that is too low risks mislabeling flows that are externally congested. We do not want to build a model that is extremely sensitive to this parameter either. We therefore test a range of threshold values and show that our results are robust to these values.

3.3 Controlled Experiments Results

We obtained robust results from our decision-tree classifier on our test data without having to carefully tune it. Figure 3 shows how the congestion threshold affects the model, and its impact on prediction *precision*, and *recall*, for both classes of congestion. Lower thresholds, *e.g.* below 0.3, lead to poor results for predicting external congestion, while high thresholds, *e.g.* greater than 0.95, lead to poor results for predicting self-induced congestion. The precision and recall are consistently high for a wide range of values between 0.3 and 0.9, however, indicating that the model is therefore accurate and robust to a choice of threshold in that range. Good results for thresholds as low as 0.3 is partly because we only have a small number of data samples in the region between 0.3 and 0.6 (only about 12% of our sample), due to the difficulty in reliably configuring the testbed for middling throughput. We therefore only consider the region which have a high number of samples and good results—between 0.6 and 0.9.

Why do we need both metrics? Both NormDiff and CoV are a function of the same underlying phenomenon, that is, the behavior of the buffer at a congested (versus an uncongested) link. Intuitively, we expect that the NormDiff parameter performs strongly as an indicator of congestion type on paths with relatively large buffers and relatively low latency and loss. In such cases, the flow can ramp up quickly and fill up the buffer. The CoV parameter gives more accurate classification across paths with smaller buffers, and higher loss and latency, because even if NormDiff is lower, the signature of a buffer that is filling is captured by CoV.

Figure 4 plots the two metrics for our controlled experiments data: we see that while the two points are largely separated on either axis, there is also a significant overlap—therefore we use both metrics in order to cover a wide range of real-world scenarios.

The impact of multiplexing In our controlled experiments, we use a clear access link and, for congestion in interdomain links, we introduce cross-traffic with 100 concurrent bulk transfers. However, either of these settings can be violated in real-world situations; there can be cross-traffic in the access link, and, congestion in interdomain links could potentially be caused by a small number of flows (particularly as access link speeds get bigger).

We run experiments to see how our classifier works when we relax these assumptions. First, we fix the access link to 50 Mbps, and reduce the number of cross-traffic processes generated by TG_{cong} from 100 to 50, 20, and 10. As we reduce the number of processes, the throughput of our test flow increases, as there is less competition. This increases the buffer occupancy of our flow, and the technique can get confounded, concluding that congestion is self-induced. Indeed, in our experiments, the fraction of flows correctly classified as externally congested decreases from 93% when there are 100 concurrent flows, to 84% when there are 50, to 74% with 20, and 50% with 10. Similarly, we introduce cross traffic in the access link, with 1, 2 and 5 concurrent flows. Even with 1 cross-flow, our test flow does not obtain the full throughput; however it still gets significant buffer space; therefore the classifier still classifies 86% of flows as self-limited. With 5 flows, this number is 70%. In both cases, due to TCP's sharing mechanism, our test flow is able to obtain significant buffer occupancy, and therefore can be thought of as influencing buffer behavior; this is what our technique captures.

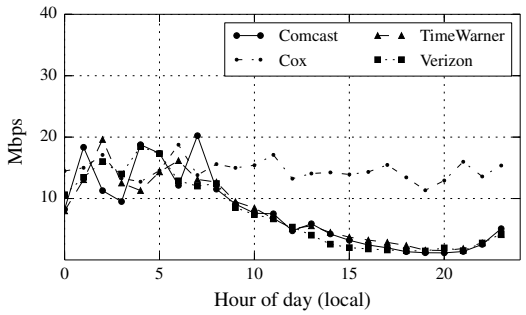
4 REAL WORLD VALIDATION USING M-LAB DATA

We validate our model on two real-world datasets from M-Lab. The first is a large dataset that spans the timeframe of a well publicized peering dispute involving Cogent, a major transit provider, and several large US ISPs in 2014. We term this dataset the *Dispute2014* dataset. The second is a more focused dataset consisting of data we collect from targeted experiments we conduct in 2017 between a client in the Comcast access network in Massachusetts and an M-Lab server hosted by TATA in New York. We choose this particular client and server combination because the interdomain link between the two ISPs experienced periods of congestion during our experiments, inferred using the Time Series Latency Probes (TSLP) [35] methodology. We term this dataset the *TSLP2017* dataset.

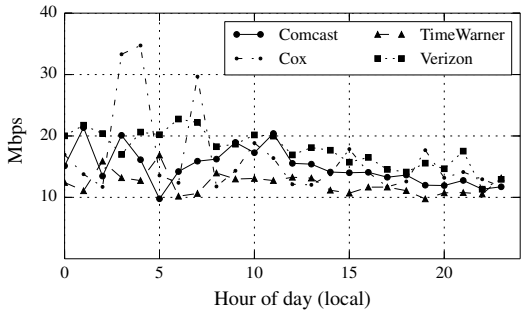
4.1 The Dispute2014 dataset

M-Lab and NDT

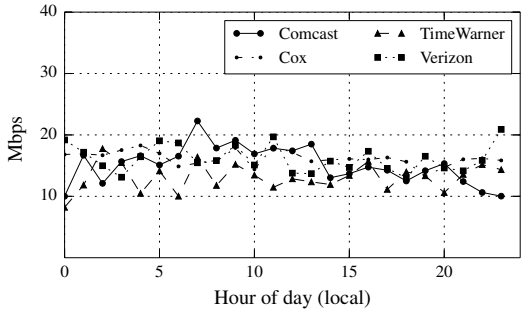
The M-Lab infrastructure currently consists of more than 200 servers located in more than 30 hosting networks globally, hosting a variety of network performance tests such as NDT [17], Glasnost [31] and Mobiperf [2]. Our study focuses on the Network Diagnostic Test (NDT) [17]. NDT is a user-initiated Web-based test which measures a variety of path metrics between the server and the client, including upstream and downstream throughput. For every NDT measurement, the server logs Web100 [7] statistics that



(a) Diurnal throughput graph for Cogent customers to M-Lab server in Los Angeles, January 2014. All ISPs except Cox see significant diurnal effects. Only Cox had a direct peering agreement with Netflix via the latter's OpenConnect program. For other ISPs, the NDT measurements were affected by congestion in Cogent caused by Netflix traffic. February has a similar pattern. [20]



(b) Diurnal throughput graph for Level3 customers to M-Lab server in Atlanta, January 2014. No ISPs see diurnal effects. Level3 did not have significant congestion issues in this time period, at least in the paths that NDT measures.



(c) Diurnal throughput graph for Cogent customers to M-Lab server in Los Angeles, April 2014. In contrast to January, ISPs do not see diurnal effect anymore. Cogent relieved congestion on their network, and Comcast signed an agreement with Netflix [20, 22]. March has a similar pattern.

Figure 5: Diurnal average throughput of NDT tests. We use the diurnal effect in Cogent to all ISPs (except Cox) in Jan-Feb as the basis for our labeling—tests during peak hours in January are externally congested. In contrast, we label off-peak Cox tests in Jan-Feb through Cogent, and all tests in Mar-Apr as affected by self-induced congestion.

provide TCP performance over 5 ms intervals. Web100 statistics include a number of useful parameters including the flow's TCP RTT, counts of bytes/packets sent/received, TCP congestion window, counts of congestion window reductions, and the time spent by the flow in "receiver limited", "sender limited" or "congestion limited" states. The server also stores raw packet traces for the tests in `pcap` format. We obtain both the Web100 logs and the trace files through Google's Big Query and Big Store [1, 6] where they are freely available.

Pre-processing the NDT data

We are interested in flows that experience congestion, so we filter the M-Lab data accordingly. We choose NDT measurements with downstream tests that lasted at least 9 seconds—the duration of the NDT measurement is 10 seconds, so these tests are likely to have completed—and which spend at least 90% of the test duration in the *congestion limited* state. We get this information from the Web100 output. We thereby exclude flows which were sender- or receiver-limited, or did not experience congestion for other reasons, such as high loss or latency. We do so because such flows would not have induced congestion in the path, and were not affected by external congestion either, and we therefore are not interested in them. These filters are the same as those used by M-Lab in their 2014 report that examines interdomain congestion [36]. These are also necessarily different from our earlier definition of congestion (using thresholds based on access link capacity) because in the bulk of our dataset, where we depend on crowdsourced measurements of the NDT data, we do not know the ground truth access link capacity of the users that ran these tests.

The peering disputes of 2013/2014.

In late 2013 and early 2014, there were media reports of poor Netflix performance on several access ISPs that did not peer directly with Netflix [12]. Netflix traffic to these access ISPs was delivered by multiple transit ISPs such as Cogent, Level3, and TATA. Throughput traces from the M-Lab NDT platform during this time period showed *strong diurnal patterns*, with significantly lower throughput during peak traffic hours (evening) as compared to lightly loaded periods (middle of the night). Such diurnal patterns were evident in throughput tests from Cogent servers to several large ISPs such as Comcast, Time Warner, and Verizon. A notable exception was Cox, which had already entered into a direct peering agreement with Netflix. Consequently, Cox's interconnections to Cogent were not affected by Netflix traffic and throughput from Cogent to Cox did not show diurnal patterns. Figure 5a shows the diurnal throughput performance of AT&T, Comcast, Cox, TimeWarner, and Verizon customers to NDT servers in Cogent. We see a substantial drop in throughput during peak hours to all ISPs except Cox. Other transit ISPs such as Level3, however, did not show such diurnal throughput patterns in the NDT data (Figure 5b). M-Lab released an anonymous report that concluded that the cause of performance degradation was peak-time congestion in interdomain links connecting the transit ISPs hosting M-Lab servers to access ISPs [36]. The reasoning was based on coarse network tomography—since NDT tests between Cox customers and the M-Lab server in Cogent did not show a diurnal pattern, whereas tests between other ISPs and the same server did, the report stated that the source of congestion was most likely the peering between Cogent and Comcast, Time Warner, and Verizon. Another independent study also confirmed the existence of congestion in these paths and narrowed down the source of congestion to ISP borders [35].

During the last week of February 2014, Cogent began prioritizing certain traffic in order to ease congestion within their network [20, 21], and then Comcast signed a peering agreement with Netflix [22]. These events had the desired effect in terms of easing congestion; we see in Figure 5c that NDT measurements to the Cogent server in Los Angeles no longer exhibited diurnal effects. We observed similar patterns for Cogent servers in New York and Seattle (not shown).

Collecting and labeling Dispute2014

We do not have service plan information for the Dispute2014data, and so we cannot use the threshold technique to label it; we instead use Figure 5 to inform our labeling. We label peak hour (between 4 PM and 12 AM local time) tests in January and February from affected ISPs (*i.e.*, those that see a sharp drop in performance during peak hours; Comcast, Time Warner, and Verizon) as externally congested. We label off-peak tests in March and April (between 1 AM and 8 AM local time) as self-induced congestion limited. To minimize noise, we do not consider off-peak tests in January-February, or peak tests in March-April. Our labeling method assumes that off-peak throughput tests are limited by access-link capacities; this assumption is based on annual reports from the FCC Measuring Broadband America (MBA) program, that show that major ISPs come close to or exceed their service plans [29]. However, we do not have ground truth, and therefore our labeling is necessarily broad and imperfect. Nevertheless, the substantial difference in throughput between peak and off-peak hours, along with the general agreement in the community about the existence of peering issues lead us to expect that a large percentage of our labels are correct.

We consider two transit ISPs: Cogent, which was affected by the peering disputes and Level3, which was not. We study two geographical locations—Los Angeles (LAX), and New York (LGA)—for Cogent, and one location—Atlanta (ATL)—for Level3. We study four access ISPs: Comcast, TimeWarner, Verizon, and Cox. Of these, Cox is unique because its performance to Cogent was not affected by the peering disputes, and therefore is useful as a control to show that our techniques are generally applicable across different transit and access ISPs.

4.2 The TSLP2017 dataset

The Dispute2014 dataset is large, but it is broad, and our labeling is coarse. We also do not have ground truth available for that dataset to accurately label flows as access-link or externally congested. We therefore run targeted experiments to generate a dataset with more accurate labels. Our goal was to find an interdomain link that was periodically congested—and that we could reliably identify as so—and run throughput tests from a client whose access-link capacity we know to a server "behind" the congested link. A challenge with this basic idea is that we must find an M-Lab server such that the path from our client to the server crosses a congested link. To do so, we use the dataset resulting from our deployment of the TSLP [35] tool on the Archipelago (Ark) [19] infrastructure to identify interdomain links between several large access ISPs and transit/content

providers that showed evidence of congestion. The TSLP technique measures the latency from a vantage point located within a network to the near and far routers of interdomain links of the host network. An elevated latency to the far end of the link, with no elevation to the near end, suggests that the targeted link is congested. Luckie *et al.* [35] showed that periodic increase in latency occurring when the expected load on the link is the highest (during peak hours in the evening) is a good indicator of congestion on the interdomain link. The TSLP technique is quite different from what we propose in this paper; TSLP targets specific links that are identified in advance, and it requires external probes to measure those links. We ran traceroute measurements from our Ark node in Comcast's network located at Bedford, Massachusetts to each M-Lab server, to find instances of paths that traversed congested interdomain links identified with the TSLP technique.

Collecting TSLP2017

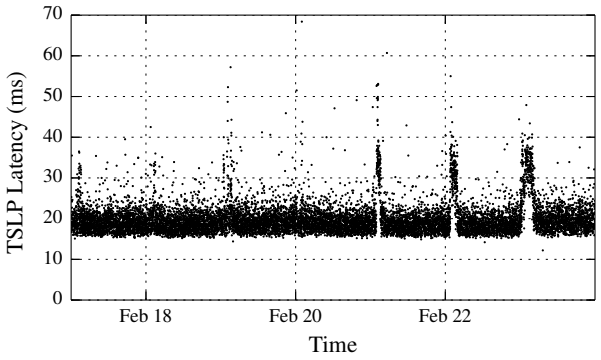
We find one case where the path between the Ark node and an M-Lab server hosted by TATA in New York traverses the interconnect between Comcast and TATA in New York City that is occasionally congested as indicated by an increase in the latency across that link during peak hours. The latency increase we measured went from a baseline of about 18ms to a peak of above 30ms. This increase of about 15ms likely reflects the size of the buffer on the link between the Comcast and TATA routers.

We run periodic, automated NDT measurements between this Ark node and the TATA server during both peak and off-peak hours. We establish the baseline service plan for this Comcast user—25 Mbps downlink—by talking to the Ark host, running independent measurements using `netperf`, and by examining the NDT throughput achieved during off-peak hours. We run throughput tests every 1 hour during off-peak hours and every 15 minutes during peak hours between February 15 and April 30, 2017. Our TSLP measurements are continuously ongoing.

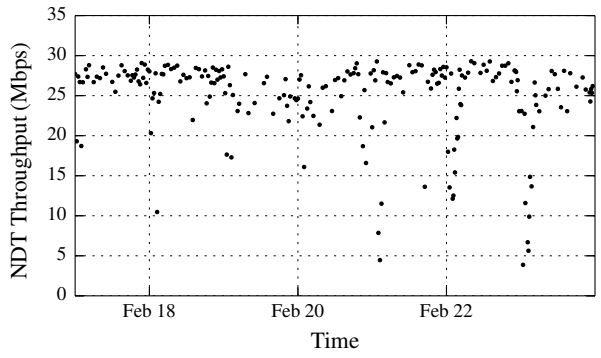
Indeed, we find a strong negative correlation between the NDT throughput and the TSLP latency to the far end of the link (Figure 6). During periods when the latency is low (at the baseline level), the NDT test almost always obtains a throughput close to the user's service plan rate of 25 Mbps. During the episodes of elevated latency, throughput is lower. We collect and process this data the same way we do the data from Dispute2014.

Labeling TSLP2017

We know both the expected downstream throughput of the access link (25 Mbps), and the baseline latency to the TATA server (18 ms), which increases to above 30 ms during congested periods. We label a test as externally limited if the throughput is less than 15 Mbps *and* the minimum latency is greater than 30ms; and as access-link congested otherwise. This allows us to be certain that the tests that we labeled as externally limited were conducted during the period of elevated latency detected by TSLP and were affected by congestion on the interdomain link. We collected 2593 NDT tests in the 10 week period, of which we were able to label 20 tests as externally congested and the rest as self-induced. The relatively low number of external congestion events speaks to the difficulty of getting real-world congestion data.



(a) Latency measurements using TSLP



(b) Throughput measurement using NDT

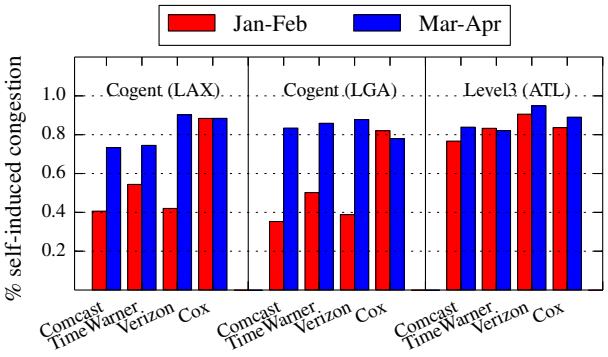
Figure 6: Sample of the data we use in TSLP2017 showing measurements of latency and throughput between an Ark node in a Comcast network in Massachusetts and an M-Lab node in New York City hosted by TATA. There are periodic spikes in latency that indicate congestion in the interdomain link between Comcast and TATA. These latency spikes correspond to drops in throughput for the Ark host; we therefore label these periods as externally congested and the other periods as self-induced. The service plan for the host is 25 Mbps downstream.

5 RESULTS

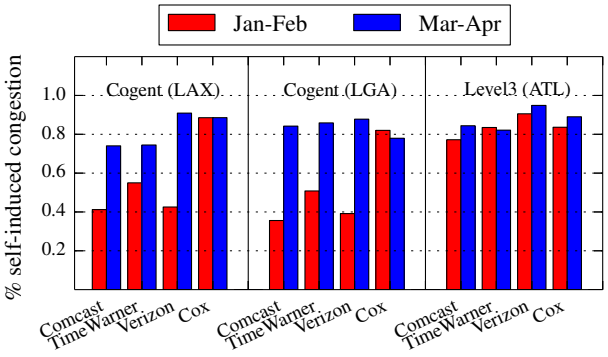
In this section, we analyze the M-Lab datasets using our model and show that it can detect the real-world peering incidents from the measurements.

5.1 Performance of the classifier on the Dispute2014 dataset

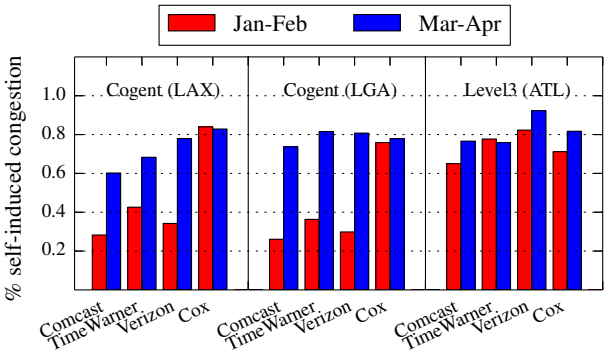
We test the classifier on the Dispute2014 dataset using labels that we describe in § 4.1. Given the nature of the peering dispute and its effect on flow congestion (Figure 5), we would expect that with perfect labeling *no* peak-hour flows between Cogent and three ISPs—Comcast, TimeWarner, and Verizon—would be classified as experiencing self-induced congestion in the January-February timeframe, while *all* flows would be classified as self-induced congestion in March-April. However, due to the number of confounding factors



(a) Threshold 0.7



(b) Threshold 0.8



(c) Threshold 0.9

Figure 7: Performance of our classifier on Dataset 1. The classifier detects a higher fraction of self-induced bottlenecks in March-April than in January-February for paths that had congestion issues during January-February (Comcast, TWC, and Verizon to Cogent servers), but resolved them by March-April. Self-induced classification fractions are similar in the two periods for paths that did not have congestion issues in this timeframe (Cox to Cogent servers, and all ISPs to Level3).

that make our labeling imperfect, we look for a large difference in the fraction of flows to those ISPs classified as self-induced congestion in the two timeframes, specifically a significantly larger fraction of flows classified as self-induced congestion in the March-April timeframe.

Figure 7 plots, for each combination of transit ISP and access ISP, the fraction of flows in each time frame that we classify as experiencing self-induced congestion (Jan-Feb in red, Mar-Apr in blue). We separately plot the results we obtain for classification models built using three thresholds for detecting access link congestion: 0.7, 0.8, and 0.9. The results align with our expectation: we see a significantly lower fraction of flows for Comcast, TimeWarner, and Verizon to Cogent classified as self-induced in Jan-Feb compared to Mar-Apr. For example, in Figure 7b, we see that our classifier classifies only 40% of flows from Comcast in LAX to Cogent as limited by self-induced congestion in Jan-Feb, while this number is about 75% in Mar-Apr. The numbers for Verizon are 40% and nearly 90% for the same combination of ISP and location.

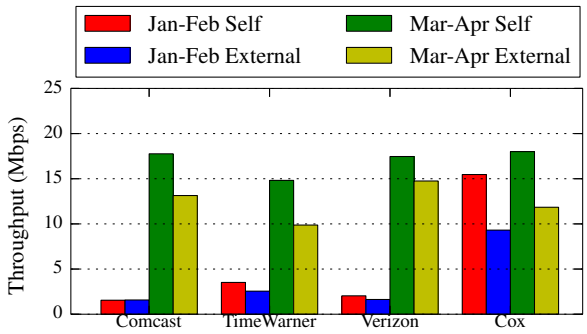
In contrast, we see little difference in the fraction of flows from Cox to Cogent and from all ISPs to Level3 that we classify as experiencing self-induced congestion in both time frames. For example, our classifier classifies about 80-90% of flows in both time frames as experiencing self-induced congestion and the rest as external(Figure 7b).

For Level3 to all access ISPs and Cogent to Cox, there is a small difference in the fraction of flows classified as experiencing self-induced congestion in the two timeframes: for example, in Figure 7c, we classify about 70% of Cox flows to Level3 as experiencing self-induced congestion in Jan-Feb, but closer to 80% in Mar-Apr. We expect that this difference is because we use peak-hour data in Jan-Feb, and off-peak hour data in Mar-Apr to minimize a potential source of error in our labeling. Even under normal circumstances, we expect more variability in throughput tests during peak hours, and therefore more tests to be affected by external congestion during those hours than during off-peak hours.

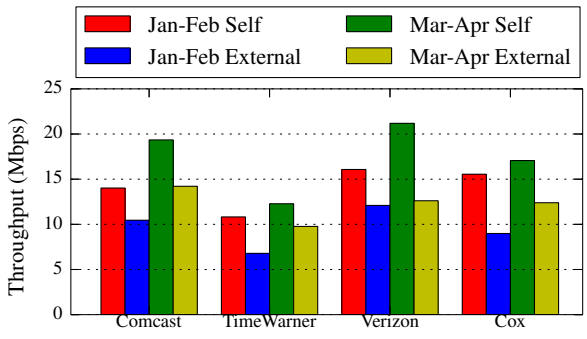
Figure 7 also shows the effect of the congestion threshold we use for the model. Higher values of the threshold mean that the criterion for estimating self-induced congestion is stricter, and we therefore expect to see fewer self-induced congestion events at higher thresholds. The fraction of flows classified as experiencing self-induced congestion drops as the threshold goes up; for example, for the Comcast/LAX/Cogent combination, the fraction during Jan-Feb goes down from 40% to slightly less than 30% as the threshold increases from 0.7 to 0.9. Changing the threshold, however, does not qualitatively alter the trend of the results.

5.2 Comparing throughput of flows in the two classes

We validate our classification using another insight. In the general case, when there is no sustained congestion affecting all the measured flows (*i.e.*, when episodes that cause externally limited flows do occur, they are isolated and do not affect all flows uniformly), we expect that externally limited flows would see lower throughput than self-limited flows; by definition, externally limited flows do not attain access link capacity. However, when there is sustained congestion that affects all measured flows (such as congestion at



(a) Cogent in LAX and LGA.



(b) Level3 in ATL.

Figure 8: Comparing the performance of classified flows before and after Dispute2014 resolution. We see that there is no difference between the throughput of the two classes during a congestion event (Cogent in LAX and LGA, for all ISPs except Cox), and a significant difference when there is no congestion event (Cogent/Cox, and all ISPs in Level3).

an interconnection point), we expect that the throughput of flows experiencing both self-induced and external congestion would be similar. This is because all flows traverse the congested interconnect. Flows through low capacity access links can still self-induce congestion, while flows through larger access links will not, however *the throughputs of flows in either class will be roughly similar.*

We illustrate this with a simple example. Consider that a flow traversing a congested interconnect can achieve X Mbps. For an access link with capacity Y > X, the flow will be externally limited with throughput X. For an access link with capacity Z < X or Z ≈ X, the flow can be access limited with a throughput of Z which is less than or equal to X. So if there is a congested interconnect that many flows traverse, then the throughput those flows achieve will be close irrespective of whether they experience self-induced or external congestion. If on the other hand there is no sustained congestion at interconnects, the distribution of self-limited throughputs should follow the distribution of access link capacities. Assuming that the self-limited and externally-limited flows sample the same population at random, the distribution of externally-limited throughputs should have a lower mean.

Figure 8 shows the median throughput of flows classified as self-induced and externally congested in both the January-February and the March-April time frames for Cogent and Level3. In January-February in Cogent, both sets of flows have very similar throughputs for Comcast, Time Warner and Verizon (Figure 8a). On the other hand, Comcast, Time Warner, and Verizon flows in March-April that were classified as self-limited exhibit higher throughput than flows constrained by external congestion. As expected, Cox does not show such a difference between Jan-Feb and March-April. Flows classified as experiencing self-induced congestion had higher throughput than externally limited flows in both timeframes. Figure 8b shows that in Level3 in Atlanta, which did not experience a congestion event in that time frame, there is a consistent difference between the two classes of flows.

5.3 How good is our testbed training data?

Since we built our classification model using testbed data, a natural question to ask is how sensitive is our classifier is to the testbed data? To answer this question, we rebuild the model using data from the Dispute2014 dataset, and test it on itself. More precisely, we split the Dispute2014 dataset into two, use one portion to rebuild the decision tree model, and test the model against the second portion. If our classifier is robust and not sensitive to the testbed data, we we would expect similar classification of the congestion events using either the model from the controlled experiments, or the new model built using the Dispute2014 dataset.

Our new model uses 20% of the samples of Dispute2014 data *except* the location and ISP that we are testing. For example, to test Comcast users to Cogent servers in LAX, we build the model using 20% of Dispute2014, except that particular combination. Figure 9 shows the result of the classifier using this model. That the classification of congestion—the percentage of flows classified as self-induced congestion—follows the same trend as the classification that uses the testbed data to build the model (Figure 7). For example, for the Comcast/LAX/Cogent combination, the fraction of self-induced congestion is about 15% and 55% in Jan-Feb and Mar-Apr in Figure 9 while it is about 30% and 60% in Figure 7c. In general, the new model is more conservative in classifying self-limited flows than the testbed model, but is qualitatively consistent with the testbed model. This consistency shows that our model is robust to the data used to build it, and also that the testbed data also provides data approximating to the real-world.

5.4 How well does our model perform on the TSLP2017 dataset?

The TSLP2017 dataset(§ 4.2) contains data from tests conducted between a Comcast access network in Massachusetts and a TATA server in New York. We recap our labeling criteria: since the user has a service plan rate of 25 Mbps and a base latency to the M-Lab server of about 18 ms we label NDT tests that have a throughput of less than 15 Mbps *and* a minimum latency of 30 ms or higher as limited by external congestion. We mark tests with throughput

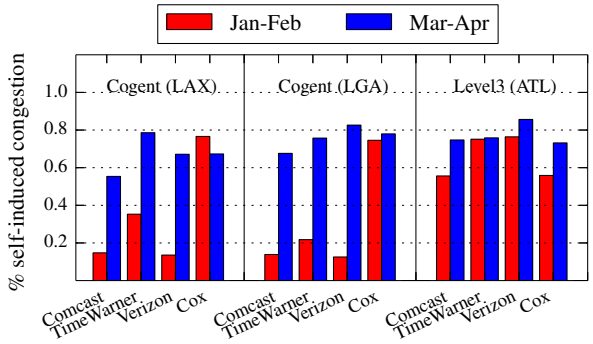


Figure 9: Detection on M-Lab data using a model built using the Dispute2014 dataset. The results are similar to the results in Figure 7c, which uses a model built using the testbed data.

exceeding 20 Mbps and minimum latency less than 20 ms as self-induced. Using our labeling criteria, we were able to obtain 2573 cases of access link congestion and 20 cases of external congestion over the course of our measurement period. Of these, our testbed model accurately classified more than 99% of self-induced congestion events and between 75% and 85% of external congestion events depending on the parameters used to build the classification model. The lower accuracy corresponds to using lower congestion thresholds to build the the model (*i.e.*, using a congestion threshold of 0.7 and 0.8 in the testbed data corresponded to an accuracy of 75%, while 0.9 corresponded to an accuracy of 85%). We also tested the TSLP2017 dataset using the model built using the M-Lab data described in Section 5.3. Our results were very similar for detecting self-induced congestion—more than 90%, while we were able to get 100% accuracy for external congestion.

The buffering that we observed in this experiment—both in the access link and the peering link—are small; about 15-20ms. Even with such a small buffer, which is essentially the worst case for our model due to its reliance on the shaping properties of buffers, our model performs accurately, furthering our confidence in the principles underlying the model.

6 LIMITATIONS

Our proposed method has limitations, both in the model, and the verification.

- **Reliance on TCP.** Our technique only works on protocols that have congestion control, and so will also potentially work with UDP-based protocols such as QUIC, though not for other UDP flows. However, the technique will work on paths that are congested by UDP flows, as long as buffering is the same for TCP and UDP.
- **Reliance on buffering for measuring the self-loading effect.** Our technique identifies flows that start on a path that has sufficient bandwidth to allow the flow to ramp up to a point that it significantly impacts the flow's RTT due to a *self-loading* effect. This has three consequences: (i) we rely on a sufficient sized buffer close to the user (at the DSLAM or CMTS) to create RTT variability. It is impractical to test all combinations of real-world buffers; however, we build and test our model using a wide variety of buffer sizes, both in our testbed and in real life, with excellent results. (ii) our classifier only classifies flows as externally limited or self-limited. We could potentially have cases where the buffer occupancy is high to the point that it affects throughput, but also is pushed to maximum occupancy by the flow we are interested in, or when multiple flows start up at the same time and congest a link; scenarios like this raise legitimate questions about whether or not the flow is self-limiting or not. We do not have a way of confirming how frequently (if at all) this occurs in the wild. Such scenarios are also difficult to recreate in the testbed. (iii) TCP flows can be limited due to a variety of other reasons such as latency, send/receive window, or loss, or even transient flash-crowding effects. We leave it for future work to develop a more comprehensive TCP diagnostic system that uses our techniques and others as building blocks. We test our techniques on a simple token bucket queue, but it will still work on other queuing mechanisms such as RED as long as there is an increase in RTT due to buffering. TCP variants such as BBR [39] that base their congestion control on latency might also confound our technique. BBR controls the amount of latency, and hence buffering that the flow induces. While testing our techniques across all TCP variants is beyond of the scope of our paper, we note that we have tested it with buffer sizes of 1–5 times the BDP: as long as the flow induces some consistent measurable buffering in the path, our technique will still identify the type of congestion accurately.
- **Reliance on the slow start time period.** We rely on TCP behavior during slow-start. The technique could therefore be confounded by a flow that performs poorly during slow-start, but then improves later on, and vice-versa. However, the classification that we obtain for the slow start period is still valid. If our model says that a flow was externally limited during slow start, but the overall throughput was higher than what the flow obtained during slow-start, we cannot tell whether the later segments were limited as well. The technique therefore gives us some understanding of the path capacity; we could use our understanding of performance during slow-start in order to extrapolate the expected behavior of the flow. We leave this for future work. However, in the reverse case, if the model says the flow was self-limited during slow-start, but overall throughput is significantly lower than what the flow obtained during slow-start, we can safely say that the throughput was affected by other factors.
- **The need for good training data for building a model.** The model fundamentally relies on a reliable corpus of training data in order to build the training model. We used training data from a diverse set of controlled experiments to build our model, and validate it against a diverse set of real-world data. Additionally, we also show that we get comparable results by building the model using real-world data. However, we do not claim that our model will work well in any setting. This problem requires a solid set of ground-truth data in the form of TCP connections correctly labeled with the type of bottleneck they experienced.
- **Reliance on coarse labeling for M-Lab data.** Due to the lack of ground truth regarding access link capacities, we label the M-Lab data coarsely (§ 4.1). However, all flows in January-February need not have been externally limited, and all flows in March-April need not have been self-limited. Variability in access link

capacities could result in low capacity links that self-induce congestion even when there is external congestion. Home network effects such as wireless and cross-traffic interference might also impede throughput, introducing noise into the labeling. However, given the severity of the congestion experienced in that time period as is evident from our analysis in § 4.1, published reports [36], and the general adherence of U.S. ISPs to offered service plans as evident from the FCC reports [5], we have reasonable confidence that our labeling is likely largely accurate.

- **Use of packet captures for computing metrics.** Our technique computes the two RTT-based metrics by analyzing packet captures. Packet captures are storage and computationally expensive. However, we note that the metrics are simple; indeed, Web100 makes current RTT values available light-weight manner. We leave it to future work to study how we can sample RTT values from Web100 to compute our metrics and how it compares to our current technique that uses packet captures.

7 RELATED WORK

There have been several diagnosis techniques proposed for TCP. T-RAT, proposed by Zhang *et al.* [52] estimates TCP parameters such as maximum segment size, round-trip time, and loss to analyze TCP performance and flow behavior. Dapper [30] is a technique for real-time diagnosis of TCP performance near the end-hosts to determine whether a connection is limited by the sender, the network, or the receiver. Pathdiag [37] uses TCP performance modeling to detect local host and network problems and estimate their impact on application performance. However, these techniques do not differentiate among types of congestion in the network. There have been several proposals for locating bottleneck links. Multiple techniques use packet inter-arrival times for localization: Katabi *et al.* [34], to locate shared bottlenecks across flows, Sundaresan *et al.*, to distinguish between a WAN bottleneck and a wireless bottleneck [48], and Biaz *et al.* [11] to understand loss behavior. A number of packet probe techniques in the literature use external probes to identify buffering in paths [35] or to measure available bandwidth or path capacity [23, 32, 33, 43, 51]. Sting [45] and Sprobe [44] are tools to measure packet loss and available bandwidth, respectively, using the TCP protocol. Antoniades et al. proposed *abget* [9], a tool for measuring available bandwidth using the statistics of TCP flows. While external probing techniques can be useful in locating the bottleneck link, such techniques are out-of-band and could be confounded by load balancing or AQM, and in the best case can only be indirectly used to deduce type of congestion (a congested link between two transit ISPs likely causes external congestion for all flows that traverse the link). Network tomography has also been proposed for localizing congestion [36, 41], or for discovering internal network characteristics such as latencies and loss rates using end-to-end probes [16, 24, 25]. Such techniques, however, are typically coarse in nature, can be confounded by factors such as load-balancing and multiple links comprising a single peering point, and require a large corpus of end-to-end measurement data to apply the tomography algorithm. Tomography cannot be applied on a single flow to infer the type of congestion that the flow experienced or the location of the bottleneck. Our goal in this work was to characterize the nature of congestion experienced by a given TCP flow based on flow statistics that are available at the server-side.

8 DISCUSSION/CONCLUSION

Till recently, last mile access links were most likely to be the bottleneck on an end-to-end path. The rise of high-bandwidth streaming video combined with perpetually fractious relationships between major players in the ecosystem has expanded the set of potential throughput bottlenecks to include core peering interconnections. Understanding whether TCP flows are bottlenecked by congested peering links or by access links is therefore of interest to all stakeholders – users, service providers, and regulators. We took some steps toward this goal by developing a technique to differentiate TCP flows that fill an initially unconstrained path from flows bottlenecked by an already congested link.

The intuition behind our technique is that TCP behavior (particularly in terms of flow RTTs during the slow-start phase) is qualitatively different when the flow starts on an already congested path as opposed to a path with sufficient available capacity. These path states correspond to peering-congested and access-link-limited flows, respectively. We show that the RTT variance metrics (both the normalized difference between the maximum and minimum RTTs, and the coefficient of variation in the RTT samples) are higher when a TCP flow is limited by a non-congested link, and therefore the TCP flow itself drives queuing (and hence RTT) behavior. We use this intuition to build a simple decision tree classifier that can distinguish between the two scenarios, and test the model both on data from the controlled experiments and real-world TCP traces from M-Lab. We tested our model against data from our controlled testbed as well as a labeled real-world dataset from M-Lab and show that our technique distinguishes the two congestion states accurately, and is robust to a variety of classifier and network settings.

We emphasize two strengths of our technique. First, it operates on single flows, and uses statistics of on-going TCP flows rather than requiring out-of-band probing. Second, it requires TCP connection logs or packet captures at the server-side only and does not require control or instrumentation of the client-side. This approach differs from techniques for available bandwidth estimation or other bottleneck detection tools that generally require out-of-band probing and/or control over both endpoints of the connection. Our work also opens up avenues for future work, particularly in developing more accurate TCP signatures that can further help us understand network performance.

ACKNOWLEDGMENTS

We thank the anonymous reviewers, and our shepherd, Costin Raiciu for their comments and feedback which improved the paper. We also thank Measurement Lab for their timely help with the NDT data. This work was supported by NSF awards CNS-1539902, CNS-1535796, and CNS-1414177.

REFERENCES

[1] M-Lab Dataset. http://www.measurementlab.net/data.
[2] M-lab Mobiperf. https://www.measurementlab.net/tools/mobiperf/.
[3] M-Lab NDT Raw Data. https://console.cloud.google.com/storage/browser/m-lab/ndt.

[4] M-Lab Network Diagnostic Test. http://www.measurementlab.net/tools/ndt.

[5] Measuring Fixed Broadband Report - 2016. https://www.fcc.gov/general/measuring-broadband-america.

[6] NDT Data Format. https://code.google.com/p/ndt/wiki/NDTDataFormat.

[7] Web10g. https://web10g.org.

[8] R. Andrews and S. Higginbotham. YouTube sucks on French ISP Free, and French regulators want to know why. *GigaOm*, 2013.

[9] D. Antoniades, M. Athanatos, A. Papadogiannakis, E. P. Markatos, and C. Dovrolis. Available Bandwidth Measurement as Simple as Running Wget. In *Proceedings of the Passive and Active Measurment Conference (PAM)*, 2006.

[10] S. Bauer, D. Clark, and W. Lehr. Understanding Broadband Speed Measurements. In *38th Research Conference on Communication, Information and Internet Policy*, Arlington, VA, Oct. 2010.

[11] S. Biaz and N. H. Vaidya. Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver. In *IEEE Symposium on Application - Specific Systems and Software Engineering and Technology (ASSET)*, 1999.

[12] J. Brodkin. Netflix Performance on Verizon and Comcast has been Dropping for Months. http://arstechnica.com/information-technology/2014/02/10/netflix-performance-on-verizon-and-comcast-has-been-dropping-for-months.

[13] J. Brodkin. Time Warner, Net Neutrality Foes Cry Foul Over Netflix Super HD Demands, 2013.

[14] J. Brodkin. Why YouTube Buffers: The Secret Deals that Make-and-break Online Video. *Ars Technica*, July 2013.

[15] S. Buckley. France Telecom and Google Entangled in Peering Fight. *Fierce Telecom*, 2013.

[16] R. Caceres, N. G. Duffield, J. Horowitz, and D. F. Towsley. Multicast-based Inference of Network-internal Loss Characteristics. *IEEE Transactions on Information Theory*, 45(7), Nov. 1999.

[17] R. Carlson. Network Diagnostic Tool. http://e2epi.internet2.edu/ndt/.

[18] Y. Chen, S. Jain, V. K. Adhikari, and Z.-L. Zhang. Characterizing roles of front-end servers in end-to-end performance of dynamic content distribution. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, Nov. 2011.

[19] k. claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov. Internet Mapping: from Art to Science. In *IEEE DHS Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, pages 205–211, Watham, MA, Mar 2009.

[20] Cogent Now Admits They Slowed Down Netflix's Traffic, Creating A Fast Lane & Slow Lane. http://blog.streamingmedia.com/2014/11/cogent-now-admits-slowed-netflixs-traffic-creating-fast-lane-slow-lane.html, Nov. 2014.

[21] Research Updates: Beginning to Observe Network Management Practices as a Third Party. http://www.measurementlab.net/blog/research%5Fupdate1, Oct. 2014.

[22] Comcast and Netflix Reach Deal on Service. https://www.nytimes.com/2014/02/24/business/media/comcast-and-netflix-reach-a-streaming-agreement.html, Feb. 2014.

[23] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion Techniques and a Capacity-estimation Methodology. *IEEE/ACM Transactions on Networking*, 12(6), Dec. 2004.

[24] N. Duffield. Simple Network Performance Tomography. In *Proceedings of ACM SIGCOMM Internet Measurement Conference (IMC)*, 2003.

[25] N. Duffield, F. L. Pesti, V. Paxson, and D. Towsley. Inferring Link Loss Using Striped Unicast Probes. In *Proceedings of IEEE Infocom*, 2001.

[26] J. Engebretson. Level 3/Comcast Dispute Revives Eyeball vs. Content Debate, Nov. 2010.

[27] J. Engebretson. Behind the Level 3-Comcast peering settlement, July 2013. http://www.telecompetitor.com/behind-the-level-3-comcast-peering-settlement/.

[28] P. Faratin, D. Clark, S. Bauer, W. Lehr, P. Gilmore, and A. Berger. The growing complexity of Internet interconnection. *Communications and Strategies*, (72):51–71, 2008.

[29] Measuring Broadband America. https://www.fcc.gov/general/measuring-broadband-america.

[30] M. Ghasemi, T. Benson, and J. Rexford. Dapper: Data Plane Performance Diagnosis of TCP. In *Proceedings of the Symposium on SDN Research (SOSR)*, 2017.

[31] Glasnost: Bringing Transparency to the Internet. http://broadband.mpi-sws.mpg.de/transparency.

[32] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang. Locating Internet Bottlenecks: Algorithms, Measurements, and Implications. In *Proceedings of ACM SIGCOMM*, 2004.

[33] M. Jain and C. Dovrolis. End-to-end Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Transactions on Networking*, 11(4):537–549, Aug. 2003.

[34] D. Katabi and C. Blake. Inferring Congestion Sharing and Path Characteristics from Packet Interarrival Times. Technical Report MIT-LCS-TR-828, Massachusetts Institute of Technology, 2002.

[35] M. Luckie, A. Dhamdhere, D. Clark, B. Huffaker, and kc claffy. Challenges in Inferring Internet Interdomain Congestion. In *Proceedings of ACM SIGCOMM Internet Measurement Conference (IMC)*, Nov. 2014.

[36] M-Lab Research Team. ISP Interconnection and its Impact on Consumer Internet Performance - A Measurement Lab Consortium Technical Report. http://www.measurementlab.net/publications.

[37] M. Mathis, J. Heffner, P. O'Neil, and P. Siemsen. Pathdiag: Automated TCP Diagnosis. In *Proceedings of the Passive and Active Network Measurement Conference (PAM)*, Berlin, Heidelberg, 2008.

[38] Measurement Lab. http://measurementlab.net, Jan. 2009.

[39] Neal Cardwell and Yuchung Cheng and C. Stephen Gunn and Soheil Hassas Yeganeh and Van Jacobson. BBR: Congestion-Based Congestion Control. https://cacm.acm.org/magazines/2017/2/212428-bbr-congestion-based-congestion-control/fulltext.

[40] Ookla. How Does the Test Itself Work? https://support.speedtest.net/hc/en-us/articles/203845400-How-does-the-test-itself-work-How-is-the-result-calculated-.

[41] V. N. Padmanabhan, L. Qiu, and H. J. Wang. Server-based Inference of Internet Link Lossiness. In *Proceedings of IEEE INFOCOM*, 2003.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, R. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] V. J. Ribeiro, R. H. Riedi, and R. G. Baraniuk. Locating Available Bandwidth Bottlenecks. *IEEE Internet Computing*, 8(5):34–41, 2004.

[44] S. Saroiu, K. Gummadi, and S. Gribble. SProbe: Another Tool for Measuring Bottleneck Bandwidth. In *Work-in-Progress Report at the USENIX USITS*, 2001.

[45] S. Savage. Sting: A TCP-based Network Measurement Tool. In *Proceedings of USENIX USITS*, 1999.

[46] sklearn.tree.decisiontreeclassifier. http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[47] S. Sundaresan. Cross-traffic Generator. https://github.com/ssundaresan/congestion-exp.

[48] S. Sundaresan, N. Feamster, and R. Teixeira. Home Network or Access Link? Locating Last-mile Downstream Throughput Bottlenecks. In *Proceedings of the Passive and Active Measurement Conference (PAM)*, 2016.

[49] S. Sundaresan, D. Lee, F. Yun, X. Deng, and A. Dhamdhere. Challenges in Inferring Internet Congestion using Throughput Measurements. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, Nov. 2017.

[50] Verizon. Unbalanced Peering, and the Real Story Behind the Verizon/Cogent Dispute. http://publicpolicy.verizon.com/blog/entry/unbalanced-peering-and-the-real-story-behind-the-verizon-cogent-dispute, June 2013.

[51] H. Wang, K. S. Lee, E. Li, C. L. Lim, A. Tang, and H. Weatherspoon. Timing is Everything: Accurate, Minimum Overhead, Available Bandwidth Estimation in High-speed Wired Networks. In *Proceedings of ACM SIGCOMM Internet Measurement Conference (IMC)*, 2014.

[52] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.