End-to-end Detection of ISP Traffic Shaping using Active and Passive Methods

Partha Kanuparthy, Constantine Dovrolis
Georgia Institute of Technology *

ABSTRACT

We present end-to-end measurement methods for the detection of traffic shaping. Traffic shaping is typically implemented using token buckets, allowing a maximum burst of traffic to be serviced at the peak capacity of the link, while any remaining traffic is serviced at a lower shaping rate. The contribution of this paper is threefold. First, we develop an active end-to-end detection mechanism, referred to as ShaperProbe, that can infer whether a particular path is subject to traffic shaping, and in that case, estimate the shaper characteristics. Second, we analyze results from a large-scale deployment of ShaperProbe on M-Lab over the last few months, detecting traffic shaping in several major ISPs. Our deployment has received more than one million users so far from 5,700 ISPs. Third, we modify the ShaperProbe detection algorithm so that it can be applied passively on the traffic of any TCP-based application. We apply this passive method in NDT traces, also collected at M-Lab. Our work is the first to design detection methodologies and to measure traffic shaping deployments in the Internet.

1. Introduction

The increasing penetration of broadband access technologies, such as DSL, DOCSIS and WiMAX, provides users with a wide range of upstream and downstream service rates. Broadband users need to know whether they actually get the service rates they pay for. On the other hand, ISPs now have an extensive toolbox of traffic management mechanisms they can apply to their customers' traffic: application classifiers, schedulers, active queue managers etc. In this paper we focus on a class of such mechanisms referred to as *traffic shapers* or *traffic policers*.[1]

A traffic shaper is a single-input single-output packet forwarding module that behaves as follows: Consider a link of capacity C bps, associated with a "token bucket" of size σ tokens. Whenever the bucket is not full, tokens are generated at a rate ρ tokens per second, with $\rho < C$. The link can transmit an arriving packet of size L bits only if the token bucket has at least L tokens - upon the transmission of the packet, the shaper consumes L tokens from the bucket. So, if we start with a full token bucket of size σ tokens, and with a large burst of packets of size L bits each (suppose that σ is an integer multiple of L for simplicity), the link will be able to transmit k of those packets at the rate of the capacity C, with $k = \frac{\sigma/L}{1-\rho/C}$. After those k packets, the link will start transmitting packets at the token generation rate ρ. Usually ρ is referred to as the "shaping rate", the capacity C is also referred to as the "peak rate", while σ is referred to as the "maximum burst size". Another way to describe a traffic shaper is by specifying that the *maximum number of bytes* that can be transmitted in any interval of duration τ, starting with a full token bucket, is:

$$\hat{A}(\tau) = \min\{L + C\tau, \sigma + \rho\tau\}$$

The difference between a traffic shaper and a traffic policer is that the former has a buffer to hold packets that arrive when the token bucket is empty [6]. A policer simply drops such "non-conforming" packets. In other words, a shaper delays packets that exceed the traffic shaping profile (σ, ρ), while a policer drops them.[2] Policers can cause excessive packet losses and so shapers are more common in practice - we focus on the latter in the rest of the paper.

Why would a residential ISP deploy traffic shaping? First, to allow a user to exceed the service rate that he/she has paid for, for a limited burst size. In that case the user pays for ρ bps, with the additional service capacity $C - \rho$ is marketed as a free service enhancement. This is, for instance, how Comcast advertises their PowerBoost traffic shaping mechanism [5]. Second, an ISP may want to limit the service rate provided to the aggregate traffic produced or consumed by a customer, or to limit the service rate consumed by a certain application (e.g. BitTorrent). This form of shaping is relevant to the "heated" *network neutrality* debate. Third, certain ISPs prefer to describe their service rates as upper bounds for what the user will actually get, e.g., a downstream rate of *at most* 6Mbps. In that case, a shaper can be used to enforce the upper bound of the service rate.

The contribution of this paper is threefold. First, we develop an *active end-to-end detection mechanism*, referred to as *ShaperProbe*, that can infer whether a particular path is subject to traffic shaping, and in that case, estimate the shaper characteristics C, ρ and σ. Second, we analyze results from a large-scale deployment of ShaperProbe on M-Lab over the last few months, detecting traffic shaping in several major ISPs. Our deployment received about one million runs over the last two years from more than 5,700 ISPs;

*School of Computer Science, Contact author: partha@cc.gatech.edu

[1] When it is not important to distinguish between shaping and policing, we will simply refer to such mechanisms as "traffic shapers" or just "shapers".

[2] A shaper will also drop packets once its droptail buffer is full.

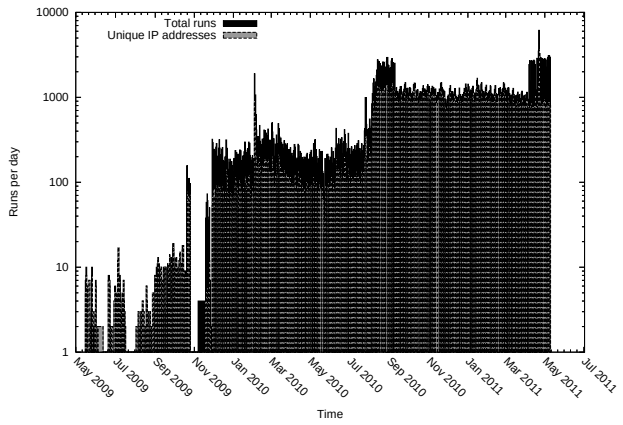
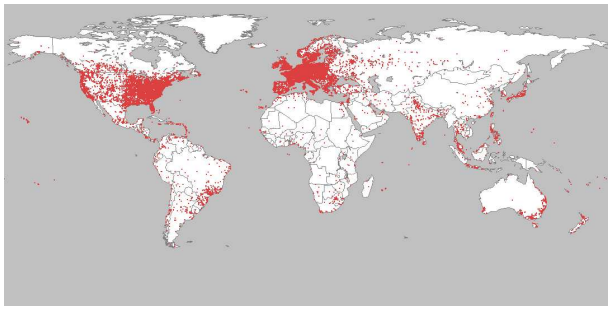


Figure 1: ShaperProbe: volume of runs. The gaps in time show downtime due to tool enhancements.



(a) Location of clients.

Figure 2: ShaperProbe: location of clients.

we currently receive 2,000-3,000 users a day (see Figures 2 and 1). Third, we modify the ShaperProbe detection algorithm so that it can be applied passively on the traffic of any TCP-based application. This approach can be used to detect if an ISP shapes the traffic of specific applications. We apply this passive method in the Network Diagnostic Tool (NDT) traces [9] collected at M-Lab. NDT is a TCP speed test and diagnosis tool, which collects packet traces for 10s TCP bulk-transfers in upload and download directions.

Traffic shaping detection and estimation methods can be used in different ways: *as a library (API)* that allows applications to adapt their performance in an *open-loop* fashion; and as a *service* that enables users and administrators to verify their SLAs/shaping configurations. In this paper, we design a service. There are several challenges that we need to tackle when designing a service that can scale up to thousands of users a day: from *accuracy* to *usability* to *non-intrusiveness* (on the network). In the following sections, we look at each of the factors.

The rest of this paper is structured as follows. Section 2 describes the active detection method. Section 3 discusses implementation and deployment of ShaperProbe, and some of the measurement systems issues we had to tackle over revisions of the tool. Section 4 looks at the data we collected from ShaperProbe using case studies of four ISPs. Section 5 describes the passive detection method, and Section 6 applies the method to NDT traces. Section 7 covers related work. We conclude in Section 8.

2. Active Probing Method

The active probing method is an end-to-end process in which a sender $\mathcal{SND}$ sends packets on the network path to the receiver $\mathcal{RCV}$. We diagnose traffic shaping for the path $\mathcal{SND} \rightarrow \mathcal{RCV}$ at $\mathcal{RCV}$. Suppose that the narrow link's capacity on the path is C, and that the sender probes at a *constant bit rate* $R_s = C$.

The receiver $\mathcal{RCV}$ records the received rate timeseries $R_r(t)$. We compute $R_r(t)$ by discretizing time into fixed size non-overlapping intervals of size Δ. For simplicity, assume that the probing starts at $t = 0$, and that intervals are numbered as integers $i \geq 1$. The i'th interval includes all packets received in the interval $[(i-1)\,\Delta, i\Delta)$, where packet timestamps are taken at $\mathcal{RCV}$ upon receipt of the last bit of each packet. The discretized received rate timeseries $R_r(i)$ is estimated as the total bytes received in interval i divided by Δ. Note that this estimator of $R_r(t)$ can result in an error of up to $\epsilon = \pm S/\Delta$ where S is the MTU packet size. By choosing a reasonably large Δ, we can reduce the magnitude of ϵ relative to the true received rate.

In the presence of a token bucket traffic shaper (or policer) on $\mathcal{SND} \rightarrow \mathcal{RCV}$, there exists a value of $i > 1$ at which the received rate timeseries $R_r(i)$ undergoes a *level shift to a lower value*. Our goal is to detect the presence of a level shift, and estimate the token bucket parameters using $R_r(i)$.

2.1 Detection

We want to detect a level shift in R_r in *real-time* (as we compute the received rate in each new interval). Note that the receiver $\mathcal{RCV}$ is also receiving (and timestamping) packets during this process. Hence, our detection method has to be simple. Our detection method is straightforward and relies on nonparametric rank statistics of R_r to be robust to outliers [14].

We compute ranks online. Suppose that we have estimated n values of R_r so far (in other words, the probing duration is $n\Delta$). At the start of each new interval $n+1$ (receipt of first packet in the interval), we compute $R_r(n)$ and update the ranks $r(i)$ of $R_r(i)$ for $i = 1 \ldots n$. We call τ as the *start of level shift* if all of the following three conditions hold true (for the smallest such index):

First, all ranks to the left of τ are *equal to or higher* than all ranks to the right of τ:

$$\min_{i=1\ldots\tau-1} r(i) \geq \max_{j=\tau+1\ldots n} r(j) \tag{1}$$

Second, we have observed a minimum time duration before and after the current rate measurement:

$$n_L < \tau < n - n_R \tag{2}$$

The value of n_L is chosen based on empirical observations of burst durations in ISPs, and n_R is a sanity check to ensure that the drop in rate is not a temporal variation (e.g., due to

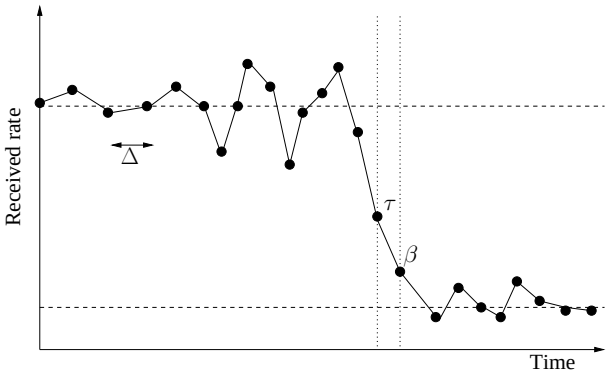


Figure 3: Active probing: Level shift detection.

cross traffic). Third, we require that there is a *drop* in the median rate at point τ:

$$\underset{i=1\ldots\tau}{\tilde{R}_r(i)} > \gamma \underset{j=\tau\ldots n}{\tilde{R}_r(j)} \quad (3)$$

where $\tilde{R}_r$ denotes the median, and γ is a suitable threshold (we include τ in both median estimates). We use the non-parametric median rate instead of the mean so that we are not heavily influenced by the magnitude of temporal drops in rate. We choose γ based on empirical observations of capacities and shaping rates in ISPs (Section 2.3).

Similarly, we detect the *end of level shift* index β such that $\beta \geq \tau$ and β is the *last* point which satisfies the rate condition in Equation 1. Figure 3 illustrates the two level shift indices.

2.2 Estimation

After we detect a level shift, we estimate the token bucket parameters from the rate timeseries R_r and the start and end level shift indices τ and β. The token generation rate (shaping rate) ρ is estimated as the *median* of received rate measurements *after* β (median to be robust to outliers):

$$\hat{\rho} = \underset{i=\beta+1\ldots n}{\tilde{R}_r(i)} \quad (4)$$

We estimate the token bucket depth (burst size) σ from the number of bytes sent till the τ'th time interval. We estimate a range for σ, since we discretize time in to intervals of size Δ, based on the estimate of ρ ($\hat{\rho}$) and the received rates:

$$\hat{\sigma} = \sum_{i=1}^{\tau} [R(i) - \hat{\rho}]\Delta \pm \frac{[R(i) - \hat{\rho}]\,\Delta}{2} \quad (5)$$

We assume in this discussion that the sent rate was always higher than ρ - we ensure this by designing our probing method to be able to send at a constant rate close to the narrow link capacity (Section 3).

2.3 Parameter Selection

We choose the parameters Δ and γ empirically, based on our experience with shaping detections in ISPs. Some of the parameters have been revised over new releases of ShaperProbe, and the current values are as follows.

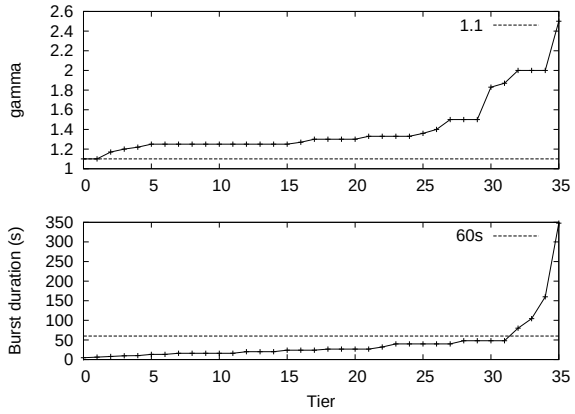


Figure 4: Advertised tiers for Comcast and Cox: required γ and Λ.

Figure 4 shows the ratio γ and burst durations[3] (minimum time required to consume tokens of a full bucket) using advertised tier information that we collected for Cox, and for Comcast in metro Atlanta. We note that all tiers have a capacity-to-shaping rate ratio of 1.1 and above, with two out of 36 tiers having a ratio of 1.1. We choose $\gamma = 1.1$ in our implementation. A conservative (low) γ works in practice, given that we require a number of points before and after the level shift, and that we send a constant rate probing stream.

We fix the probing duration Λ so that we detect as many ISP shaping configurations as possible, while at the same time keeping the total experiment duration reasonable in case of no-shaping. We see from Figure 4 that the burst duration is at most 48s, except for four tiers out of 36. We set $\Lambda = 60$s in our implementation.

Next, the averaging window size Δ is chosen so that it is large enough to offset estimation noise in R_r and small enough to include sufficient rate samples for detecting a level shift in a probing time of Λ. We fix Δ as follows. We perform 100 trials in the upstream direction on a Comcast residential connection, whose SLA we know (4.5Mbps shaped to 2Mbps). Figure 5 shows the shaping detection rate for different values of Δ. We see that as Δ approaches the inter-packet gap, the detection rate drops. We fix a large value, $\Delta = 300$ms, so that we can detect shaping in low capacity links.

3. Implementing ShaperProbe

While implementing ShaperProbe, we faced several challenges in designing a tool that works well on a variety of network conditions, OS platforms, and form factors (desktops and portable devices). First, we require a fast and accurate estimate of the narrow link capacity between the sender and receiver; we use this estimate to probe the end-to-end path. We use packet trains for capacity estimation; this estimate, however, has to be robust to non-FCFS links such

[3]Note that ISPs do not advertise bucket depths for some tiers; we choose the *most likely* σ for a given ρ based on the ShaperProbe data.

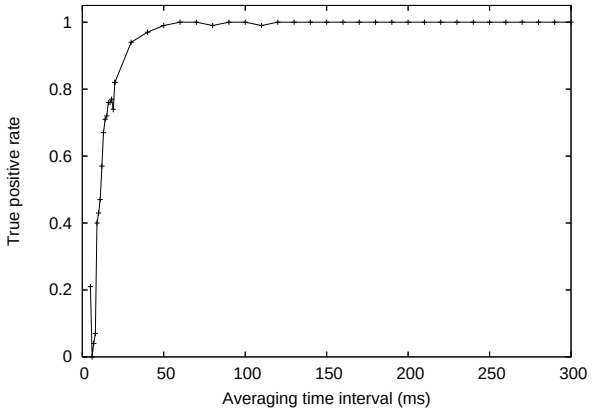


Figure 5: Effect of Δ: Comcast upstream.

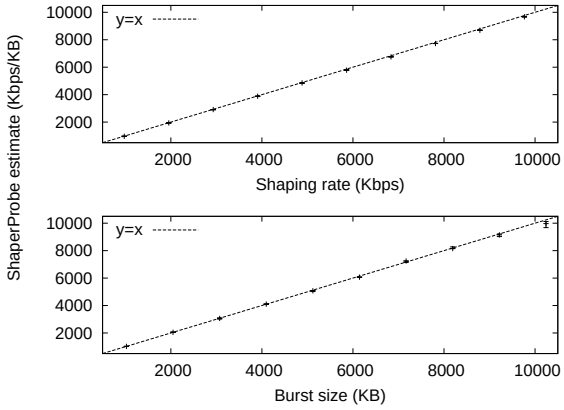


Figure 6: Estimation accuracy: shaping emulation.

as "noisy" 802.11 deployments in homes. Second, we design the probing method to be able to send at a *constant rate*, even at coarse-grained userspace OS timer granularities. While doing so, we avoid consuming CPU resources on the sender. Third, the ShaperProbe client has to be non-intrusive - it ends probing if it sees losses on the path (and reports to the user a diagnostic). Finally, cross traffic on the path may lead to temporal drops in the received rate R_r, in spite of the above implementation details - we need to design a filtering mechanism on R_r that can remove *outliers*. We discuss all of the above details in **Appendix A**.

In this section, we describe the tool design and our experience in improving measurement accuracy, reliability and mitigating noise, as well as our implementation of the M-Lab ShaperProbe service [11], which has been operational on M-Lab since May 2009.

The M-Lab Service: We have implemented the ShaperProbe client in user space on three platforms: Windows, Linux, and OS X for 32-bit and 64-bit architectures. The server runs on Linux. The non-GUI server-client functionality is about 6000 lines of open source native code. The client is a download-and-click binary and does not require superuser privileges or installation. A version of the ShaperProbe client has also been ported as a plugin to the *Vuze* BitTorrent client, and clocks several hundreds of users a day.

The server and client establish a TCP control channel for signaling. The client starts with capacity estimation in both upstream and downstream directions, and follows it with shaping detection in each direction. At the server, we log per-packet send and receive timestamps and sequence numbers for all probing phases of the tool, and the IP address and server timestamp for each run.[4] A typical run of ShaperProbe in a residential connection lasts for about 2.5-3 minutes.

ShaperProbe has been implemented as a service on M-Lab [11]. We currently run ShaperProbe server replicas on 48 M-Lab machines connected to tier-1 ASes. For measurement accuracy, we allow only one client at a time at one server replica. Incoming client measurement requests are first served by one of three load balancers, which redirects clients to a server instance uniform randomly; a fall-through mechanism further redirects clients from busy servers. The service has received about one million user runs over the last two years, and we currently receive 2000-3000 runs a day. Figures 2 and 1 show the volume and geography of runs.

[4] ShaperProbe data is provided unanonymized by M-Lab.

4. Shaping in ISPs

In this section, we take a first look at results from the ShaperProbe service deployed at M-Lab. We first look at accuracy using two ISPs for which we know the shaping ground truth, as well as using wide-area emulations.

Accuracy: We test the latest version of the ShaperProbe service on two residential ISPs, AT&T and Comcast, at two homes in metro Atlanta. We use the High-Speed Internet service of Comcast, and the DSL service of AT&T. At experiment time, Comcast was shaping {10Mbps up, 22Mbps down} to {2Mbps up, 12Mbps down} [5], while AT&T fixed link capacities to {512Kbps up, 6Mbps down} [2] without shaping. Out of 60 runs, we found zero false positives in either direction on AT&T, and two upstream false negatives due to capacity underestimation (under cross traffic) on Comcast.

We next emulate token bucket shaping on a wide-area path between a residential Comcast connection and a server deployed at the Georgia Tech campus. We use the LARTC `tc` tool on Linux with a 2.6.22 kernel on a dual-NIC 1GHz Celeron router with 256MB RAM. Figure 6 shows the Wilcoxon median estimate and confidence intervals for ShaperProbe's token parameter estimates on 20 trials for each token bucket configuration in the downstream direction. ShaperProbe detects all 200 trials as shaping, and accurately estimates the shaping rate and bucket depth for all configurations.

We now look at an analysis of M-Lab data. The ShaperProbe service [11] has been up since May 2009, and had seen several improvements over the first few months. We start with preprocessing on our traces.

Data preprocessing: We analyze data collected from the ShaperProbe service. First, we consider runs from the latest ShaperProbe release, collected between 20th October 2009

ISP	Upstream (%)	Dwnstrm. (%)
Comcast	**71.5** (34874)	**73.5** (28272)
Road Runner	6.5 (7923)	**63.9** (5870)
AT&T	10.1 (8808)	10.9 (7748)
Cox	**63** (5797)	**47.4** (4357)
MCI-Verizon	5.6 (8753)	8.4 (7733)

Table 1: Shaping detections: top-5 ISPs in terms of ShaperProbe runs. For each ISP we show percentage of runs with detected shaping and number of total runs.

and 9th May 2011 (845,223 runs). Each run's trace contains per-packet timestamps and sequence numbers for upstream and downstream probing ("half runs"). Second, we call a half run as "unfinished" if no shaping was detected and the run lasted less than a threshold duration, and discard such runs - we choose a conservative threshold of 50s. All finished half runs which are not diagnosed as shaping are considered as cases of *no-shaping*. Note that ShaperProbe probes each direction for 60s, and ends a half run if it either found shaping or if it found packet losses during probing; a half run can also be unfinished if the user aborted the client before it could run to completion. After preprocessing, we have a total of 281,394 upstream and 236,423 downstream half runs from a total of 5,693 ISPs.

Next, we cluster AS numbers into ISPs using their `whois` AS names. The AS information was obtained from Cymru's whois database in May 2011. We found that runs which passed the pre-processing checks come from 5,167 ISPs. The top five ISPs in terms of the number of runs we received, and the fraction of shaping detections are shown in Table 1.

It should be noted that there are several factors that influence the fraction of shaping detections of an ISP. First, ISPs provide multiple tiers of service, and some tiers may not have shaping (and tiers evolve with time). Second, an ISP may not deploy shaping in all geographic regions. Third, the access link type can be a factor: a DSL provider can dynamically change link capacity instead of shaping, while a cable provider is more likely to shape since DOCSIS provides fixed access capacities. Fourth, for a given connection, shaping could be dynamic based on time or load conditions in the ISP. Fifth, an ISP A can advertise address prefixes on its customers' behalf, and some of these customers (say B) could be ISPs deploying shaping (while A does not) - we can not distinguish A from B based on BGP prefix-to-ASN mapping. We study some of these factors in ISP case studies next. A few ISPs, however, disclose their tier shaping configurations; in such cases, we validate our observations.

4.1 Case Study: Comcast

Comcast offers Internet connectivity to homes [5] and enterprises [3], and uses two types of access technologies: cable (DOCSIS 3.0) and Ethernet. In each access category, it offers multiple tiers of service. Comcast shapes traffic using the PowerBoost technology [4].

Shaping profiles: We observed many shaping configurations in our observations between October 2009 and May 2011. Figure 7 shows the shaping configuration (capacity, shaping rate, and burst size) of each user run. We see that there are strong *modes* in the data; Table 2 is a summary of these modes. For higher link capacities, we see more number of modes in the shaping rate; however, at the tail of the capacity distribution, there is only one shaping rate that corresponds to the higher service tier provided by Comcast. We verified our tier observations with the Comcast website listings [3, 5]. Note that we may not observe all service tiers in this figure, depending on the distribution of users we receive across tiers. We also observe two to three burst sizes that are used across all tiers; the PowerBoost FAQ mentions 10MB and 5MB burst sizes [4].

C (Mbps)	ρ (Mbps)	σ (MB)	Burst time (s)
3.5	1	5	16.7
4.8	2	5, 10	15.2, 30.5
8.8	5.5	10	25.8
14.5	10	10	18.8

(a) Upstream.

C (Mbps)	ρ (Mbps)	σ (MB)	Burst time (s)
19.4	6.4	10	6.4
21.1	12.8	10	10.1
28.2	17	20	14.9
34.4	23.4	20	15.3

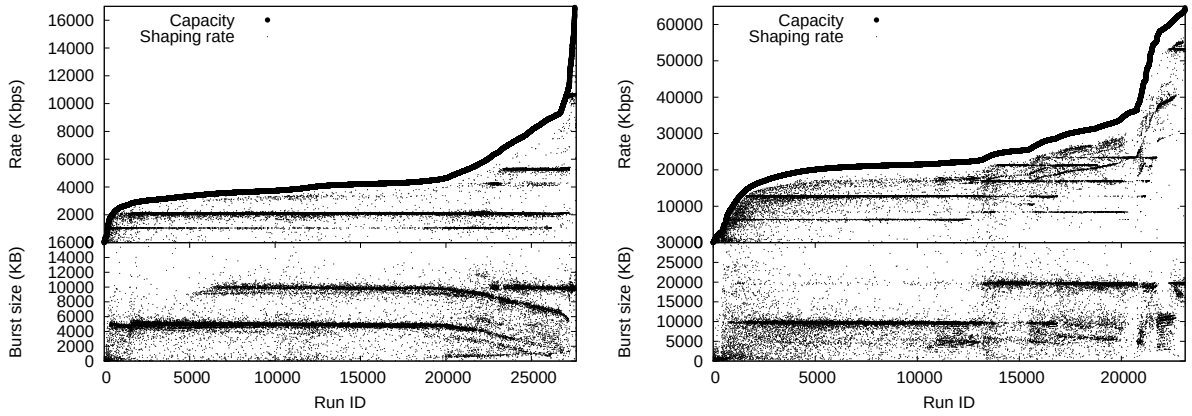
(b) Downstream.

Table 2: Comcast: detected shaping properties.

Note that the capacity curves do not show strong modes, unlike the shaping rates. This is due to the underlying DOCSIS access technology. The cable modem uplink is a non-FIFO scheduler; depending on activity of other nodes at the CMTS, the capacity estimates can vary due to the scheduling and DOCSIS concatenation. A DOCSIS downlink can also influence dispersion-based capacity estimates depending on activity in the neighborhood, since it is a point-to-multipoint broadcast link.

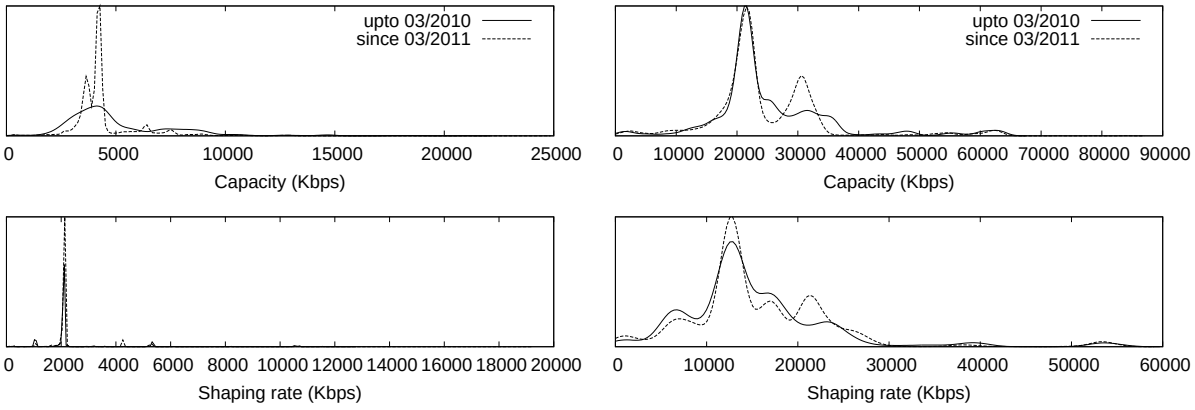
Did shaping configurations change with time? We compare data separated by about two years from Comcast - collected in October 2009-March 2010 and in 2011 (March-May). Figure 8 shows pdf estimates of shaping rates using a Gaussian kernel density function. We see that the capacity and shaping rates in the upstream direction did not change significantly; while the downstream links show a new capacity mode of 30Mbps and a shaping rate mode of 22Mbps in 2011. We did not find significant changes in burst size with time. Note that the above analysis assumes that the distribution of users across tiers remained *identical* between the two times.

Did shaping parameters change with time of day? We compare runs at 1200 and 0000 hours UTC (timestamps are taken at the server) in Figure 9; the data is taken from March-May 2011. We see that the upstream shaping rates have a similar distribution between the two times; the downstream



(a) Upstream. (b) Downstream.

Figure 7: Comcast: Shaping characteristics.



(a) Upstream. (b) Downstream.

Figure 8: Comcast: histogram of bandwidth with time.

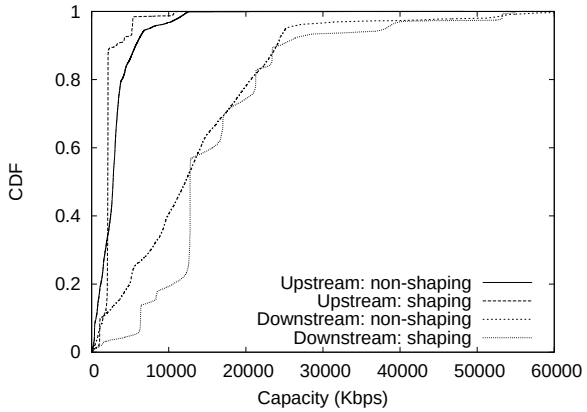


Figure 10: Comcast: CDF of capacities in shaping and shaping rate in non-shaping runs.

rates show a slight difference in densities - the lower shaping rates show a higher density at evenings in US times (PST/EST). The burst sizes show a similar trend - we see a higher density of lower burst sizes in the evenings than in the mornings. Note that the above analysis assumes that *the user sample (in terms of tier) that we get at different hours of day are identically distributed*.

Non-shaped runs: We look at all runs which were not diagnosed as shaping. Figure 10 compares distributions of capacities among such runs with shaping rates from shaping-detected runs. The non-shaped capacity distributions are *similar* to the shaping rate distributions; non-shaping runs can occur due to one or more of the following two reasons. First, Comcast provides service tiers which do not include PowerBoost, but have capacities similar to tiers with PowerBoost (e.g., the Ethernet 1Mbps and 10Mbps service for businesses). Second, it is possible that cross traffic from the customer premises resulted in an empty token bucket at the start of the experiment, and hence the estimated capacity was equal to the shaping rate (and we do not detect shaping).

4.2 Case Studies: Road Runner and Cox

Road Runner (RR) is a cable ISP that provides different tiers of service. A unique aspect of RR is that we have found evidence of downstream shaping, but *no evidence* of upsteam shaping in any tier on its web pages. The ShaperProbe trials with Road Runner also support this observation - 94% of upstream runs *did not* detect shaping, while 64% of downstream runs found shaping. Another aspect of RR is that the website advertises shaping based on the geographic region; for example, in Texas, RR provides four service tiers, the lower two of which are not shaped, while the upper two tiers are shaped [10]. Figure 11 shows the downstream shaping properties in our RR runs. We see three dominant shaping rates and a single dominant burst size. Under the hypothesis that Road Runner is not shaping upstream traffic, we can say that our false positive detection rate for its upstream is about 6.4%.

The distribution of capacity among non-shaping detected

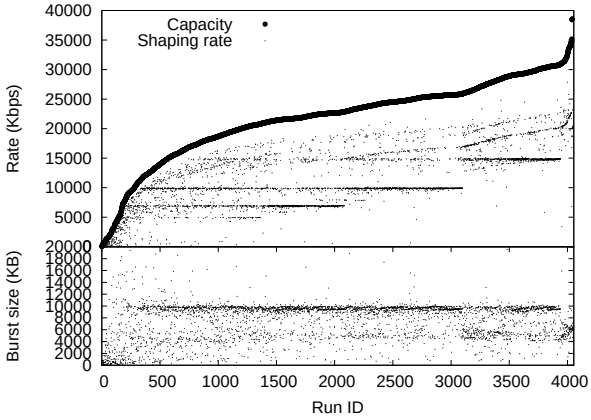


Figure 11: Road Runner: Downstream shaping.

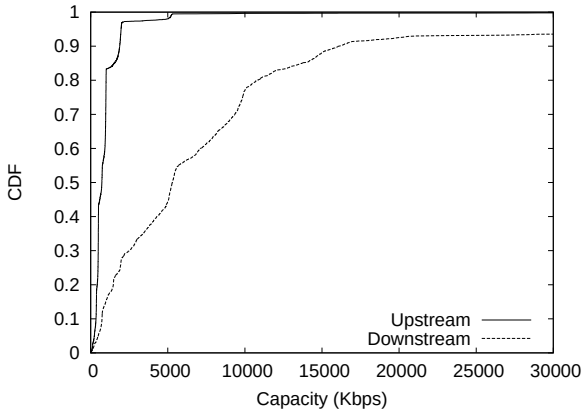
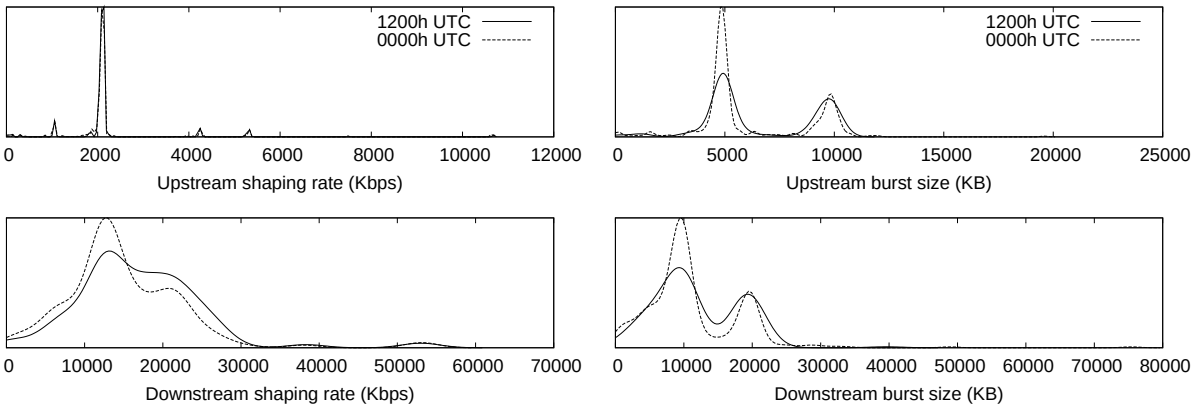


Figure 12: Road Runner: capacity of non-shaped runs.



(a) Shaping rate.

(b) Burst size.

Figure 9: Comcast: Shaping rate with time of day.

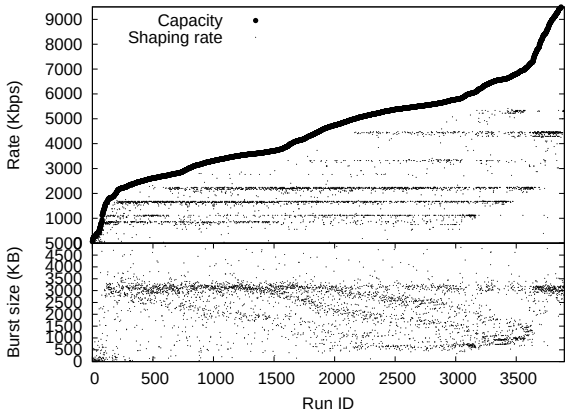


Figure 13: Cox: Upstream shaping.

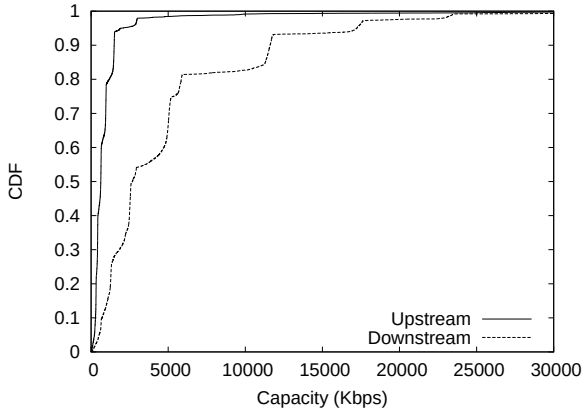


Figure 14: AT&T: Capacity of non-shaping runs.

RR runs is shown in Figure 12 (x-axis is truncated). An interesting observation with the RR non-shaping runs is that, unlike the case for Comcast, the downstream capacity mode of 750Kbps in non-shaping runs does not equal any of the shaping modes. This may indicate the possibility that some of the runs in Figure 12 are tiers that do not include shaping.

Cox provides residential [7] and business Internet access using cable and Ethernet access technologies. Similar to Comcast and RR, Cox shapes traffic at different tiers for both residential and business classes; the website shows that the residential shaping rates and capacities are dependent on geography. Moreover, we could gather residential tier shaping data from the Cox residential website [7]. For example, the upstream shaping properties of Cox runs in Figure 13 agree with some of the ground truth tier information that we found: (C, ρ)Mbps: (1, 0.77), (1.3, 1), (2, 1.5), (2.5, 1), (2.5, 2), (3, 2), (3.5, 3), (5, 4) and (5.5, 5). Note that the ground truth we collected represents a single time snapshot; while the data covers two years. We also found a single burst size mode among the runs.

4.3 Case Study: AT&T

Our final case study is that of an ISP for which we do not see a significant number of shaping detections (10% or less). AT&T provides Internet access to a wide range of customers, from homes and small businesses to enterprises (including other ISPs). They use DSL and Ethernet access technologies, and provide four tiers of DSL services [1, 2]. We did not find any mention of traffic shaping in the AT&T tier descriptions [1, 2].

Capacity: We first look at the 90% runs that were not detected as shaping. The distribution of capacities of non-shaped runs is shown in Figure 14. Given the point-to-point nature of DSL links, we can estimate the narrow link capacity more accurately than cable links; we see many modes in the capacity distribution: {330Kbps, 650Kbps, 1Mbps, 1.5Mbps} for upstream, and {1Mbps, 2.5Mbps, 5Mbps, 6Mbps, 11Mbps, 18Mbps} for downstream. Note that some of these rates might be offered by ISPs which are customers of AT&T, but whose prefixes are advertised through an AT&T's ASN. We did not observe changes in the capacity modes between

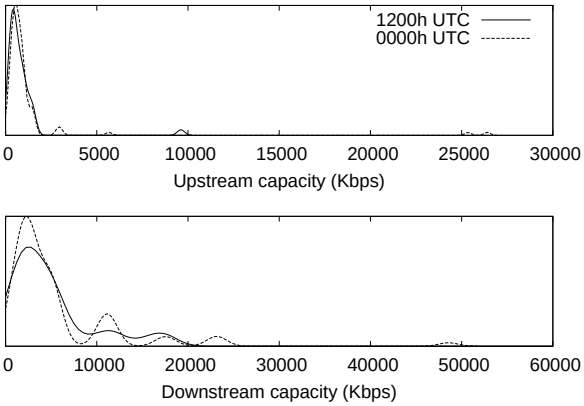


Figure 15: AT&T: pdf of capacity based on time of day.

2009, 2010 and 2011.

Did capacity change with time of day? We look at the capacity distribution at two one-hour periods of day separated by 12h; and consider non-shaping traces in the period March-May 2011. Figure 15 shows a pdf of the capacity at the two UTC times. We see that the *relative densities* of link capacities did not change significantly between different times of day.

Shaping runs: We look at properties of the 10% of AT&T runs which were diagnosed as shaping; see Figure 16. We see that about a third of these runs show a strong shaping rate mode and an associated burst size mode. Out of the 333 runs which had a shaping rate mode, 80% of the hostnames resolved to the domain `mchsi.com`, which is owned by the cable ISP Mediacom [8]. This case represents a limitation of our IP address to ISP translation method.

5. Passive Method

In this section, we design a technique for passive inference of traffic shaping. The passive inference technique takes as input an application packet trace (either real-time or offline) at a sender $\mathcal{SND}$ or the receiver $\mathcal{RCV}$[5]. The passive inference method is useful over active probing to detect cases where an ISP is shaping certain classes of traffic based on parameters such as destination/source which may not always be feasible to replicate with active probing.

We consider the specific case of a bulk-transfer application that uses TCP. Detecting shaping on TCP traffic is challenging for a number of reasons. First, TCP throughput can change with time depending on network conditions, and a level shift in TCP throughput occurs every time TCP decreases its window due to timeouts or packet losses. Second, TCP does not send a constant-rate stream, and so it harder to estimate the number of tokens in the token bucket. There can be time periods in which the TCP connection's throughput is below the shaping rate, causing accumulation of tokens. Third, after shaping kicks-in, TCP backoffs and timeouts can lead to accumulation of tokens (bringing the link service rate temporarily back to the capacity) - hence, the rate may not be bounded by ρ. We can not use the active detection method as-is on TCP traces for the above reasons. Specifically, our technique needs to distinguish throughput level shifts due to a shaper from those due to TCP backoffs.

Our passive detection method works on the received rate timeseries $R_r(t)$, and uses two properties of $R_r(t)$ when there is a shaper: (1) there will be a time t at which $R_r(t)$ undergoes a level shift, and (2) after the level shift, the received rate is *almost constant* (for a duration that depends on the connection's round-trip time and the link buffer size).

Rate estimation: We begin by constructing a received rate timeseries $R_r(i)$, $i \geq 1$, by dividing time into discrete non-overlapping intervals of size Δ. It is important that we choose a suitable value for Δ so that we have sufficient number of rate measurements to detect a level shift, and at the same time have low *noise*. We estimate Δ based on the TCP trace, since the inter-packet gaps can vary depending on TCP backoffs and timeouts. In the NDT traces, we have observed that some token bucket implementations generate tokens in periodic intervals (δ_{tb}). Depending on the length of this interval, packets buffered in a shaper that has no tokens will be served in short bursts (of size $\rho\delta_{tb}$ bytes) at the link capacity as long as there is a backlog. In order to reduce *variance* in the post-shaping rate timeseries, we estimate $R_r(i)$ as follows.

We record the start and end timestamps of received packets in the i'th interval: call them $s(i)$ and $e(i)$. If we received $B(i)$ bytes in the i'th interval, we estimate $R_r(i)$ using the inter-burst gap δ_b:

$$R_r(i) = \frac{B(i)}{\delta_b}$$

where, $\delta_b = \max\{s(i+1) - s(i), e(i) - e(i-1)\}$. The max. condition is to avoid overestimates of $R_r(i)$ by using a small δ_b, especially in the post-shaping estimates. If $B(i) = 0$ for some i, we set $R_r(i) = 0$. We ignore the TCP handshake and termination exchanges when computing R_r.

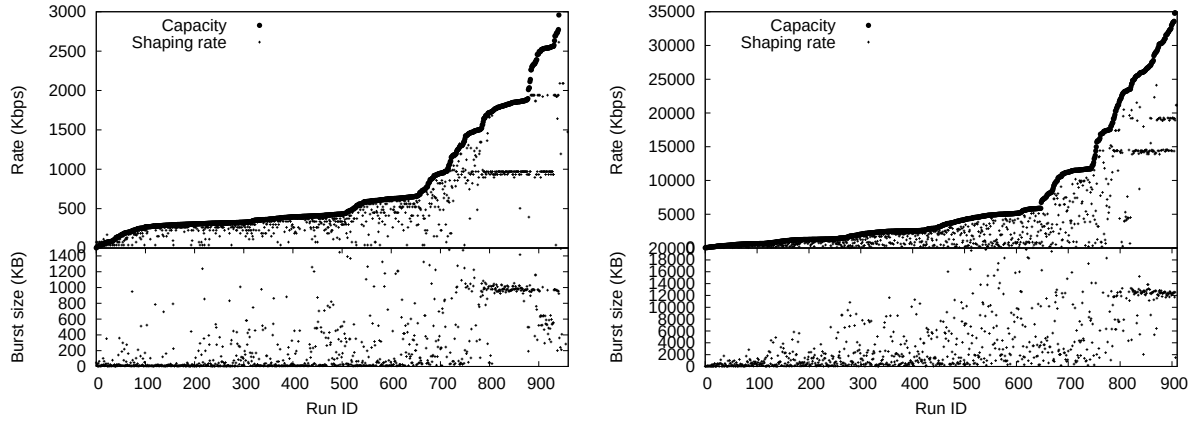
Note that in the above description, we assumed that $R_r(i)$ is estimated using the application packet trace at $\mathcal{RCV}$. We have implemented the previous method for a trace captured at $\mathcal{SND}$ by observing sequence numbers of TCP ACKs from $\mathcal{RCV}$. This works even in the case of delayed ACKs, since we treat each ACK as a *logical packet* at $\mathcal{RCV}$, whose size is determined by difference in ACK sequence numbers, and whose receive timestamp is estimated as the receive timestamp of the corresponding ACK. We ignore duplicate ACKs in this computation.

Note that it is unlikely that the TCP ACKs will be paced due to shaping in the other direction during a TCP transfer, since the MSS-to-ACK size ratio (about 28.8 without L2 headers) is much larger than the capacity ratio in asymmetric links.

5.1 Detection

We use a sliding window of size w points on the rate timeseries R_r to detect shaping. The value of w is chosen small

[5] Since a TCP end-point can be both sender and receiver (depending on the application), we distinguish sender from receiver based on user input.



(a) Upstream.

(b) Downstream.

Figure 16: AT&T: Shaping characteristics.

so that we can observe level shifts and constancy of a few points after the level shift (without noticing TCP-based rate drops). Suppose that w is odd and let $w = 2k+1$ points. We say that there is a level shift *due to shaping* at a given point τ if all of the following conditions hold true in a window w centered at τ (we require that $\tau > k$).

First, k rate values before τ are larger than the k rate values after τ:

$$\min_{i=(\tau-k)\ldots(\tau-1)} R(i) \geq \max_{j=(\tau+1)\ldots(\tau+k)} R(j) \qquad (6)$$

Second, the *aggregate* throughput of w points before τ is significantly larger than the *aggregate* throughput of w points after τ[6]:

$$\bar{R}_r(i)_{i=(\tau-w)\ldots(\tau-1)} > \gamma \; \bar{R}_r(j)_{j=(\tau+1)\ldots(\tau+w)} \qquad (7)$$

for a threshold $\gamma > 1$. We use aggregate throughput and not median, since w is small. Third, we have received at least one packet in each Δ interval after τ in a w-window:

$$R_r(i) \neq 0,\ i = (\tau+1)\ldots(\tau+w) \qquad (8)$$

This condition allows us to discard some cases of TCP timeouts. Finally, we require *constancy of rate* in the w points after τ. We elaborate on this condition next.

Rate constancy: This condition differentiates TCP recovery dynamics after a loss/timeout from the output of a shaper just after it runs out of tokens. The property we test is that just after the shaper runs out of tokens, incoming packets are buffered and serviced at the rate ρ (for a sufficiently small duration), since the incoming rate is larger than ρ. In other words, in this period, the inter-packet gap will be about S/ρ, where S is the current packet size. In practice, however, some token bucket implementations serve fixed-size bursts of packets during a shaping period. The size of this burst depends on the inter-token gap δ_{tb} and the token rate ρ, and each burst is served at the link capacity C, with a large inter-burst gap. We account for this temporal burstiness in our rate constancy heuristic.

The basic idea is to look at "variability" of a timeseries of cumulative bytes, $B_c(t)$, received at $\mathcal{RCV}$. The variability would be higher for TCP recovery than for output of a shaper. In order to construct $B_c(t)$, we identify bursts of packets served at the link capacity. We identify a burst as the largest set of consecutive packets in which each packet pair is received at a rate higher than $0.9C$. For each identified burst, we add a point (t, B) to $B_c(t)$, where t is the timestamp of the first packet in the burst, and B is the cumulative bytes received up to the end of that burst. At the end of the window $\tau + w$, if $B_c(t)$ contains less than 5 points, we say that the post-shaping rate is *not constant*.

Once we construct $B_c(t)$, we sample uniform randomly pairs of points $(t_1, B_c(t_1))$ and construct the slope m_1 of the line connecting them. We construct 500 such samples (with replacement), and estimate the *nonparametric variability* of the set of slopes $M = \{m_1 \ldots m_{500}\}$. The variability estimate $v(M)$ of M is defined as $(M_{0.9} - M_{0.1})/M_{0.5}$, where M_p is the p-percentile value in M. We avoid using the parametric equivalent, coefficient of variation, since it can be biased by few "outlier" slope samples. We say that the rate post-shaping is *constant* if, for all packets received in the time interval $[\tau+1, \tau+w]$, $v(M) < 0.15$ (we observed that this threshold is appropriate in our experiments with NDT traces from various ISPs).

5.2 Estimation

Once we detect the presence of shaping at point τ, we determine the shaping parameters as follows.

The shaping rate ρ is estimated as the aggregate received rate in a window of w points after point τ (or m points if there are $k \leq m < w$ points after τ):

$$\hat{\rho} = \bar{R}_r(j)_{j=(\tau+1)\ldots(\tau+w)} \qquad (9)$$

[6] If τ is such that we have less than w points before or after it, we consider only those points.

The link capacity C is estimated as the aggregate received rate in w points before τ:

$$\hat{C} = \max_{i=(\tau-w)\ldots(\tau-1)} \bar{R}_r(i) \quad (10)$$

It is relatively challenging to estimate the token bucket depth σ since the TCP send rate can be variable, and the arrival rate at the shaper can in fact be lower than ρ even before shaping starts. Given the received rate timeseries $R_r(i)$, the size of tokens in the token bucket at time i is given by the recursive relation:

$$\hat{\sigma}(i) = \max\left\{\min\left\{\hat{\sigma}(i-1) - [R_r(i) - \rho]\,\Delta, \sigma\right\}, 0\right\} \quad (11)$$

where, σ is the bucket depth, $\hat{\sigma}(0) = \sigma$, and $\hat{\sigma}(\tau+1) = 0$. We estimate the bucket depth as $\hat{\sigma} = \hat{\sigma}(0)$.

Equation 11 is nonlinear and depends on σ itself. We solve for σ as follows. We construct a function $\hat{\sigma}(\tau+1) = f_\sigma(\rho, \tau, R_r, \Delta, \hat{\sigma})$ that gives the value of token bytes after shaping, $\hat{\sigma}(\tau+1)$, given a value for $\sigma = \hat{\sigma}$. The function f_σ has the following properties: (1) it is non-decreasing as a function of $\hat{\sigma}$, (2) $f_\sigma \approx 0$ for $\hat{\sigma} \leq \sigma$ and (3) f_σ increases monotonically with σ for $\hat{\sigma} > \sigma$. We solve for σ using binary search on an interval $[\sigma_{\min}, \sigma_{\max}]$ such that σ is the maximum value of $\hat{\sigma}$ for which $f_\sigma \approx 0$.

In our implementation, we use $\sigma_{\min} = 10$KB, $\sigma_{\max} = 100$MB, and define the approx. relation above as $f_\sigma < 5$KB (close to the minimum bucket size on commercial implementations, as seen in NDT traces). Since binary search finishes in logarithmic time, it is possible to implement the above estimation in a live TCP capture.

5.3 Parameter Selection

We describe how we choose parameters Δ, w, and γ, based on our observations using NDT data from different ISPs.

Averaging interval: The value of Δ has to be large enough so that we construct a low variance rate timeseries. We determine Δ as the *largest* value for which we do not have any zero rate points in R_r. In addition, we bound Δ so that $\Delta_{\min} < \Delta < \Delta_{\max}$. In our implementation, we empirically choose $\Delta_{\max} = 250$ms and $\Delta_{\min} = 30$ms. [7]The above upper bound on Δ implies that for a given TCP trace, we may still have zero rate points in R_r.

We efficiently determine the value of Δ using binary search. For cases where the trace is collected at the sender $\mathcal{SND}$, we use $\Delta_{\max} = 500$ms to allow for TCP implementations that use delayed ACKs.

Window size: The value of w is chosen based on the estimate of Δ and the trace duration Λ, as the number of Δ time intervals that that fit in a fraction of Λ: $w = \kappa\Lambda/\Delta$ points for some $\kappa < 0.5$. We choose $\kappa = 0.2$ in our implementation, and round-off w to the nearest odd integer. Note that as Δ increases, w decreases to accomodate the same window duration.

[7]We could also choose $\Delta_{\min}$ by estimating the connection's round-trip time at the time of the TCP handshake.

Rate ratio: The value of γ is chosen so that we accommodate as many shaping cases as possible. We found that $\gamma = 1.6$ helps in eliminating false positives due to TCP-based rate drops. This value is larger than γ in our active detection method, and we are working on making it more robust to TCP artifacts.

6. Evaluating Passive Detection

We use traces collected from M-Lab's Network Diagnostic Tool (NDT) [9] from 25 April to 27 April, 2010. NDT performs a TCP bulk-transfer in upstream and downstream directions for 10s and collects per-packet traces at the server side kernel. Incoming NDT clients are redirected to the nearest M-Lab NDT server instance based on latency.

We consider only finished runs of NDT, which have a half run duration of exactly 10s. We analyze traces collected in the client upstream direction, since the shaping may be more likely to start in the limited duration of 10s in the upstream than in the downstream direction.

Accuracy: To benchmark the accuracy of our passive method, we collect 452 NDT upstream traces from March 11, 2010 from an M-Lab server in the US. We manually classify each trace into shaping and non-shaping categories based on visual inspection of the rate timeseries. The averaging interval to plot the rate timeseries is based on the passive algorithm for determining Δ. We found a total of 12 shaping cases out of 452 cases. Our method detects 11 of 12 cases as shaping, and all of the remaining 440 non-shaping cases as non-shaping. Note that the number of shaping cases is small (2.65%), since we are using a small experiment duration (10s), and in addition, the onset of shaping can be delayed due to TCP send rate variations.

We have also run the passive method on some of the ShaperProbe traces to verify that the two methods give consistent results.

Traces: We now look at a larger set of NDT upstream runs collected on 25-27 April 2010 for which we have ASN to ISP name mapping (we use sibling data from [25]). We have a total of 148,414 upstream runs from 900 ISPs. These traces are collected from *all* M-Lab servers, and correspond to clients from countries including US. The top ISP shaping detections by number of runs are Comcast, Road Runner, Telecom Italia, AT&T, Verizon and Cox.

We found that we see 0-2.1% upstream shaping detections for ISPs in the list, except for Cox, which had 20.5% (615 out of 2,998 runs) detections. For example, we saw 2.1% (284 out of 13,536) shaping detections in Comcast. Note that we observed less than 15% shaping detections for AT&T, Road Runner, and Verizon in the ShaperProbe data. This may be because of the experiment duration of 10s - for example, in the case of Comcast (Table 2(a)), we see that the minimum duration it takes to empty the token bucket among the upstream configurations is 15.2s (with a constant bit-rate stream sent at the link capacity); hence, we would not be able to see shaping in a 10s Comcast upstream trace. The burst duration may be longer with TCP rate variations, and we also

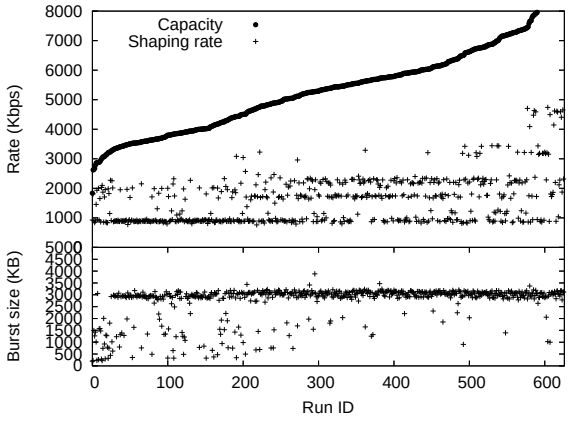


Figure 17: Passive detection: Cox: upstream.

require a minimum duration of $k\Delta$s after shaping. The fraction of shaping detections also depends on the distribution of runs among the tiers of the ISPs during the three-day data collection period. Under the hypothesis that all upstream tiers of Comcast exceed the 10s burst duration (or do not have shaping), we can say that the false positive detection rate for the passive method in cable upstream is about 2%.

6.1 Case Study: Cox

Among the ISPs in our traces, we found Cox to have a significant number of upstream shaping detections of 20%. Using the per-region advertised shaping configurations we collected from the Cox website [7], we computed the expected burst duration (for a constant send rate of C). Assuming a burst size of 3MB that we observed in the ShaperProbe data, we found that there is one tier which has a burst duration less than 10s - this tier has a capacity of 25Mbps and a shaping rate of 20Mbps (and burst duration of 5.03s).

Figure 17 shows shaping configurations of the Cox runs. The plot shows strong shaping rate and burst size modes similar to Figure 13, which is the counterpart using active probing (we show the same capacity ranges for the two plots). We see that some tiers are not sampled in this data as compared to those in ShaperProbe, since the latter is more historical. All tiers in this plot have a burst duration less than 10s - for example, for all runs with the 1Mbps shaping rate, the average capacity is 4.82Mbps; and with a 3MB token bucket size, the expected burst duration is 6.6s.

We found the following advertised tiers on the Cox website (in May 2010) for shaping rates of 1Mbps and 2Mbps: $\{C,\rho\}$ of $\{1.3, 1\}$Mbps, $\{2.5, 1\}$Mbps, $\{2.5, 2\}$Mbps, and $\{3, 2\}$Mbps. We see that the range of the estimated capacities for these two shaping rates are higher than the advertised rates. We plan to collect historical NDT traces to investigate this observation as a part of future work.

7. Related Work

Traffic shaping and policing are available in many network devices that work at L2 and above [21, 6]. Initial observations of downstream traffic shaping were recorded in a residential access network study done by Dischinger *et al.* in 2007 [12]. They used a 10Mbps train for 10s to measure received rate, and did not find evidence of upstream traffic shaping.

Recent research efforts on detecting traffic discrimination in ISPs resulted in the following tools. These tools compliment our work by detecting traffic *differentiation* policies, while we consider the problem of detecting and estimating traffic shaping with and without differentiation. Glasnost [13] uses aggregate throughput of an emulated application and compares it with a baseline to determine throughput differences, and has been deployed on M-Lab. NANO [20] focuses on detecting discrimination by comparing throughput observations from multiple end-hosts. NetPolice [24] uses ICMP probes to detect loss rate-based discrimination. NetDiff [18] uses ICMP performance and associates it with geography to diagnose differences in backbone ISPs. POPI [17] detects priority-based forwarding by observing loss rates between two flows. DiffProbe [15] detects delay and loss discrimination by comparing an emulated application flow with a normal flow. Weinsberg *et al.* further infer weights of a discriminatory scheduler [22]. In the classification area, traffic morphing [23] tackles avoiding statistical classification by altering flow characteristics.

Capacity estimation in non-FIFO scheduling environments such as wireless and cable upstream received attention in the literature, since existing bandwidth estimation techniques assume FIFO scheduling. Lakshminarayanan *et al.* [16] propose Probegap for available bandwidth estimation in cable upstream and in 802.11 links. Portoles-Comeras *et al.* study the impact of contention delays on packets in a packet train [19], which can, under certain conditions, result in the 802.11 capacity overestimates that we observed.

8. Discussion-Conclusion

Service providers increasingly deploy traffic management practices such as shaping and policing to manage resource hungry distributed applications. In this work, we presented an end-to-end active probing, detection, and estimation method of traffic shaping in ISPs. We have deployed our tool as a service on the M-Lab platform for about a year now, and discussed some of measurement issues that we have tackled over three releases of ShaperProbe. We showed using runs on two ISPs (with known shaping SLAs), and through emulations, that ShaperProbe has false positive and false negative detection rates of less than 5% in both directions.

We presented a first large-scale study of shaping, in four ISPs among our runs. We validated some of our observations using advertised tier data from ISPs. Characteristic of our plots is a strong modality of shaping rates and burst sizes across shaping detections, suggesting that ISPs typically deploy a small set of shaping configurations. We found some shaping detections for which the ISPs did not mention shaping in their service descriptions[8]. Lack of publicly-available

[8]ISPs, however, typically mention in their SLAs that "listed capacities may vary".

information, however, does not necessary imply that these are false detections.

We extended our active detection and estimation methods to work on passive TCP traffic traces. We evaluated the passive detection method on upstream traces collected from NDT on M-Lab. We also looked at detected shaping configurations of an ISP, and verified the configurations with the corresponding ShaperProbe data.

We found that upstream TCP traces with a duration of 10s are not sufficiently long to detect shaping. This can have implications on speed tests that rely on 10s transfers (possibly over TCP). The result of such a speed test would either be close to the peak rate (before shaping), or a value that falls between the peak rate and the shaping rate - because of which a user may interpret the outcome as incorrect.

Appendix A: ShaperProbe Implementation

We have assumed in the discussion in Section 2 that we have an estimate of the narrow link capacity. In practice, we can have cross traffic in the path, last mile wireless links, and end-host effects, which can add significantly to probing and measurement noise.

Capacity Estimation. We require an estimate of the narrow link capacity on the $\mathcal{SND} \rightarrow \mathcal{RCV}$ path before we probe for traffic shapers, since it determines our probing rate. We need an accurate estimate so that: (1) we minimize intrusiveness, i.e., not create persistent queue backlogs in the path due to our probing, and (2) we are able to consume tokens in the experiment duration. We implement capacity estimation as a *pre-probing* phase.

We estimate capacity using UDP packet trains. Specifically, $\mathcal{SND}$ sends a packet train of N MTU-sized (size S) back-to-back packets $\{p_1, p_2 \ldots p_N\}$ to $\mathcal{RCV}$. After receiving the train, $\mathcal{RCV}$ estimates the narrow link capacity using the train dispersion δ (calculated as the difference between received timestamps of p_N and p_1):

$$\hat{C} = \frac{(N-1)S}{\delta}$$

We repeat this measurement for K trains and determine the path capacity as the median of the K train estimates to reduce underestimates due to cross-traffic. In our implementation, we use $N = 50$ packets and $K = 10$ trains. Note that if we do not receive a part of the train[9], we use N as the number of received packets, and δ is computed as the dispersion of the *received* train.

In practice, we found that the above methodology works well for wired end-hosts, but tends to incorrectly estimate the narrow link capacity in the downstream direction when the last mile is a wireless (802.11a/b/g) link - in cases where the narrow link is not the wireless link. This overestimation[10] occurs because of the non-work-conserving contention delay process in CSMA/CA access links, due to which a downstream train may have to wait until the channel is free [19]. We also observed differences in overestimates with some 802.11g NICs depending on the operating system/driver implementation. In order to avoid such overestimates, we send a *long* packet train after we determine the train-based capacity estimate. Specifically, if the train-based estimate is $\hat{C_T}$, we send a UDP stream of MTU-sized packets at a rate $\hat{C_T}$ on $\mathcal{SND} \rightarrow \mathcal{RCV}$ for a duration of 5s and revise our capacity estimate to be the aggregate received rate in that duration. Note that this UDP stream will only induce large queue backlog if we have a significant capacity overestimate.

Probing and Non-intrusiveness. It is important that we send the probing stream for shaping detection at a constant rate that is close to the capacity C for two reasons. First, we may detect false negatives if we cannot send at a rate equal to the path capacity. Second, end-host effects such as operating system noise and context switches due to the client environment can lead to drops in the sending rate at $\mathcal{SND}$, leading to potential false positives. We probe using MTU-sized (size S) packets, and send all probing traffic over UDP.

Probing: The goal of the sender is to send a sequence of packets at a rate close to C, for a maximum of $\Lambda = 60$s. For a given sending rate, implementations of packet trains typically use a CPU-intensive busy-wait loop to maintain inter-packet gaps at userspace. Busy-wait loops can lead to a drop in send rate, since a sender process running for *extended periods* would yield to other processes. To avoid such scenarios, we send periodic *short-duration trains* of back-to-back packets to minimize time spent in busy-waits.

We determine the length of short-duration trains at runtime as follows. The idea is to minimize the duration spent on busy-waits. We first measure the userspace sleep resolution period t_s on $\mathcal{SND}$, which is the minimum duration the send process can sleep without the need for a busy-wait loop. The train length N_p (packets) is then determined as the value which minimizes the residual delay $d(N_p) = N_p g - \lfloor N_p g/t_s \rfloor t_s$ where $g = S/C$ (the inter-packet gap). We upper bound N_p to 30 packets to accommodate OSes with a low sleep time resolution (e.g., some flavors of Windows). We maintain the probing rate by: (1) sending train of size N_p, (2) sleeping for a duration t_s for $\lfloor N_p g/t_s \rfloor$ (integer) times; followed by (3) a *short* busy-wait for $d(N_p)$.

Non-intrusiveness: We stop probing at $\mathcal{SND}$ if either of the following is true: (1) $\mathcal{RCV}$ detected a traffic shaper, (2) $\mathcal{SND}$ probed for the experiment duration Λ, or (3) $\mathcal{RCV}$ detected packet losses. We record the packet loss rate in time windows of Δ. We abort on packet losses if we either see a low loss rate (more than 1%) for a small time period, or if we see a high loss rate (more than 10%) for a smaller time period (a few consecutive time windows - it is necessary that this duration is longer than $n_R\Delta$).

[9] We use a timeout of 1s to receive each packet of the train.

[10] Typical downstream train overestimates on 802.11g are in the range 25-35Mbps, close to the L2 channel throughput.

Filtering Rate Noise. The received rate timeseries R_r may have temporal variations ("noise") at Δ-timescales in practice, even if $\mathcal{SND}$ sends at a constant rate, due to cross traffic variations in intermediate links or due to end-host noise at $\mathcal{RCV}$. Such noise manifests as *outliers*. We pre-process the R_r timeseries with a simple filtering heuristic before running the level shift algorithm. Each time we compute a new rate value $R_r(n)$, we condition the value $R_r(n-n_f)$ as follows (assuming $n > 2n_f$). We compute the median of n_f rate points to the left (say $\tilde{R}^l_{n_f}$) and n_f points to the right (say $\tilde{R}^r_{n_f}$) of the point $n - n_f$. We modify $R_r(n - n_f)$ if it is either greater than or less than *both* $\tilde{R}^l_{n_f}$ and $\tilde{R}^r_{n_f}$. Under this condition, we modify the rate value to the mean $R_r(n-n_f) = (\tilde{R}^l_{n_f} + \tilde{R}^r_{n_f})/2$. We condition each rate point at most once during the experiment.

When we modify a rate point, we need to recompute ranks of all rate values that fall between the old value and the new value of $R_r(n - n_f)$. Note that this filtering method would *not change* rate values that lie in the n_f-neighborhood of the level shift points τ and β, since the median condition on the $n - n_f$ point would not be satisfied for such points. We say that a rate R_a is larger than R_b if $R_a > \gamma R_b$ where γ is the rate threshold in Equation 3.

REFERENCES

[1] *AT&T FastAccess Business DSL Plans (May 12, 2010)*. `http://smallbusiness.bellsouth.com/internet_dsl_services.html`.

[2] *AT&T FastAccess DSL Plans (May 12, 2010)*. `http://www.bellsouth.com/consumer/inetsrvcs/inetsrvcs_compare.html?src=lftnav`.

[3] *Comcast Business Class Internet (May 12, 2010)*. `http://business.comcast.com/internet/details.aspx`.

[4] *Comcast High Speed Internet FAQ: PowerBoost*. `http://customer.comcast.com/Pages/FAQListViewer.aspx?topic=Internet&folder=8b2fc392-4cde-4750-ba34-051cd5feacf0`.

[5] *Comcast High-Speed Internet (residential; May 12 2010)*. `http://www.comcast.com/Corporate/Learn/HighSpeedInternet/speedcomparison.html`.

[6] Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting. *Cisco Systems: Document ID: 19645*.

[7] *Cox: Residential Internet (May 12, 2010)*. `http://intercept.cox.com/dispatch/341670774142925900 2/intercept.cox?lob=residential&s=pf`.

[8] *Mediacom: Hish-speed Internet (May 12, 2010)*. `http://www.mediacomcable.com/internet_online.html`.

[9] *Network Diagnostic Tool (M-Lab)*. `http://www.measurementlab.net/measurement-lab-tools#ndt`.

[10] *Road Runner cable: central Texas (May 12, 2010)*. `http://www.timewarnercable.com/centraltx/learn/hso/roadrunner/speedpricing.html`.

[11] *ShaperProbe (M-Lab)*. `http://www.measurementlab.net/measurement-lab-tools#diffprobe`.

[12] M. Dischinger, A. Haeberlen, K. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *ACM IMC*, 2007.

[13] M. Dischinger, M. Marcon, S. Guha, K. Gummadi, R. Mahajan, and S. Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *USENIX NSDI*, 2010.

[14] M. Hollander and D. Wolfe. Nonparametric statistical methods. 1973.

[15] P. Kanuparthy and C. Dovrolis. DiffProbe: Detecting ISP Service Discrimination. In *IEEE INFOCOM*, 2010.

[16] K. Lakshminarayanan, V. N. Padmanabhan, and J. Padhye. Bandwidth estimation in broadband access networks. In *ACM IMC*, 2007.

[17] G. Lu, Y. Chen, S. Birrer, F. Bustamante, C. Cheung, and X. Li. End-to-end inference of router packet forwarding priority. In *IEEE INFOCOM*, 2007.

[18] R. Mahajan, M. Zhang, L. Poole, and V. Pai. Uncovering performance differences among backbone ISPs with Netdiff. In *USENIX NSDI 2008*.

[19] M. Portoles-Comeras, A. Cabellos-Aparicio, J. Mangues-Bafalluy, A. Banchs, and J. Domingo-Pascual. Impact of transient CSMA/CA access delays on active bandwidth measurements. In *ACM IMC*, 2009.

[20] M. Tariq, M. Motiwala, N. Feamster, and M. Ammar. Detecting network neutrality violations with causal inference. In *ACM CoNEXT*, 2009.

[21] G. Varghese. *Network Algorithmics: an interdisciplinary approach to designing fast networked devices*. Morgan Kaufmann, 2005.

[22] U. Weinsberg, A. Soule, and L. Massoulie. Inferring traffic shaping and policy parameters using end host measurements. In *IEEE INFOCOM Mini-conference*, 2011.

[23] C. Wright, S. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, 2009.

[24] Y. Zhang, Z. Mao, and M. Zhang. Detecting traffic differentiation in backbone ISPs with NetPolice. In *ACM IMC*, 2009.

[25] Y. Zhang, R. Oliveira, H. Zhang, and L. Zhang. Quantifying the Pitfalls of Traceroute in AS Connectivity Inference. In *PAM*, 2010.