On rate limitation mechanisms for TCP throughput: a longitudinal analysis

João Taveira Araujo[a], Raul Landa[a], Richard G. Clegg[b], George Pavlou[a], Kensuke Fukuda[c]

[a]*Dept of Electronic and Electrical Engineering, University College London*
[b]*Dept of Electrpnic Engineering and Computer Science, Queen Mary University of London*
[c]*Dept of Informatics, National Institute of Informatics, Tokyo*

Abstract

TCP remains the dominant transport protocol for Internet traffic. It is usually considered to have its sending rate covered by a sliding window congestion control mechanism. However, in addition to this normal congestion control, a number of other mechanisms limit TCP throughput. This paper analyzes the extent to which network, host and application settings define flow throughput over time and across autonomous systems. Our study draws on data from a longitudinal study spanning five years of passive traces collected from a single transit link. Mechanisms for this include limiting by application, interference with the TCP window control mechanism and artificial limitations on maximum window sizes by the operating system. This paper uses a large data set to assess the impact of each mechanism. We conclude that more than half of all heavy-hitter inbound traffic remains throttled by constraints beyond network capacity. For this data set, TCP congestion control is no longer the dominant mechanism that moderates throughput.

Keywords:

1. Introduction

This paper considers the changes to traffic on the Internet viewed from a particular sampling point over a five year time horizon. The traffic is sampled from WIDE, a Japanese academic provider. The paper focuses on TCP throughput behaviour. However, traffic over this period has been subject to many changes, including changes in network topology, capabilities and policy; changes in user behaviour; changes in traffic make-up and changes in end-user operating systems.

A TCP flow usually increases its sending rate until it detects congestion (signalled either by packet loss or by an increase in delay, according to TCP flavour) and then reduces its rate accordingly. When this happens, TCP is governed by network properties such as delay and loss. However, other mechanisms may be the dominant influence on throughput. Providers of streaming content may choose to throttle traffic to prevent wastage due to users who do not download an entire stream [1] or clients with buffering limitations such as those in mobile networks [2]. In other situations, such as one click hosting (e.g. *rapidshare* or *mediafire* [3, 4]) high throughput rates are a premium service and the file server will ensure that typical users will get a lower throughput. In these cases, which we call *application pacing*, the content provider deliberately slows traffic to the users. In the case of file sharing applications like bittorrent, users often deliberately limit their own bandwidth to allow overhead for other applications [5].

Other factors limit TCP throughput, for example socket buffer sizes set by operating system (OS) vendors or tuned by network administrators. In addition, by the nature of the TCP protocol, there is an upper bound on the window size a receiver can advertise. Without window scaling enabled [6] no TCP connection window can exceed 64KB. Another source of throughput intervention includes traffic shaping, often performed by local network operators and system administrators; evidence for this can be detected in passive traces [7]. This paper attempts to answer (from the perspective of the data set studied) the question *what are the primary mechanisms that currently govern TCP performance in the Internet*?

There is no such thing as a typical data set for studying the Internet. This data set covers a wide period of time (2007–2011) and the traffic is to and from a variety of hosts (commercial, academic, individual users). While the unanonymised data necessary for this type of study was only available up to 2011, many of the trends observed within this paper could be expected to continue. The high level of use of application paced data seems unlikely to end soon as video traffic becomes prevalent and the increasing adoption of window scaling seems unlikely to reverse. Although it would be unwise to draw too general conclusions without using the techniques in this paper on further data sets, nonetheless this study provides insight into the nature of

TCP control mechanisms likely to be in use in the wider Internet today[1].

2. Related work

Despite their inherent value, longitudinal studies of Internet phenomena are rare. The Internet has been shaped by both technological change and by political and commercial factors. These changes cause problems for observational studies where data must be collected and curated over long periods of time, and this has resulted in a scarcity of relevant datasets. Collaborative research efforts such as CAIDA [9] or Oregon Routeviews [10] have provided some datasets. The usefulness of these, however, can be severely affected by the need for data privacy. The dissemination of interdomain routing information, where no such requirement exists, has assisted in a wealth of research on wide ranging topics, from quantifying path diversity [11] to locating Internet bottlenecks [12]. In contrast, longitudinal datasets relating to passive measurements have nurtured a much smaller community of researchers often focusing on characterising traffic [13]. Stripped of the locality contained within IP addresses, researchers are left unable to relate these findings to a wider context. Instead, cross-sectional studies characterising traffic aggregated by location are frequently conducted under different contexts [14], but lack the temporal perspective only longitudinal studies can afford. Efforts to characterise the spatial properties of traffic over time [15, 16, 17] have defined the changing of Internet topology and traffic alike but fall short of relating such shifts with their impact on relevant metrics such as loss or delay.

This paper builds on a wealth of prior work on understanding Internet traffic. Much of the underlying motivation is shared with the landmark study by Zhang et al. [18] on the characteristics of Internet flow rates. Using traces spanning both access, peering and regional links, Zhang et al. analyse traffic according to potential rate limiting factors. Amongst other findings, host window limitations were found to affect over 30% of traffic for the access networks studied. Importantly, the authors found a strong correlation between flow throughput and flow size, postulating that this could derive from user behaviour, with large transfers more likely to be performed over higher bandwidth connections.

Flow characteristics and TCP behaviour at large have since been subject to frequent reassessment. Of particular relevance to the current work

[1]An earlier, reduced version of this work appeared initially in [8].

are passive studies which delve into the inner mechanisms of TCP. In [19], Jaiswal et al. infer the sender's congestion window by identifying the congestion control variant from the behaviour observed during loss recovery. The use of separate state machines for each variant however proves unscalable given the many flavours of TCP congestion control which have since been deployed. In [20], Lan et al. analyse flows according to size, duration, rate and burstiness and characterise the observed correlations for heavy-hitters specifically, uncovering evidence of increased application influence on flow rates and burstiness and consequently suggest treating flow size and duration as independent dimensions.

One central aspect to the analysis of TCP behaviour is the estimation of round trip time (RTT) from packet capture data. In addition to SYN-based methods, Shakkotai et al. [21] evaluate further techniques to estimate the RTT of a unidirectional flow. The *rate change* method establishes a relation between the RTT and the increase in sending rate, assuming linear window increases during congestion avoidance. Unfortunately, this assumption no longer holds, both due to the proliferation of less conservative congestion control algorithms such as CUBIC [22], and due to application-driven flow control. An alternative is the use of frequency-domain techniques [23, 24, 25], which are a natural fit given the self-clocking nature of TCP. However, a common difficulty with the application of spectral analysis is extracting the fundamental frequency which corresponds to the RTT in the presence of noise. In applying the Fourier transform to inter-packet arrival times, for example, Qian et al. [25] note that less than half of all flows have distinguishable *flow clocks*; our own experience with Fast Fourier Transform (FFT)-based RTT recovery was also found to be unreliable even after pre-processing available data to enhance inherent periodicities (see section 4).

Finally, it is important to elucidate which changes in traffic properties are intrinsic to TCP and data transfer, and which ones arise from large-scale changes in the Autonomous System (AS)-level topology of the Internet. In the decade and a half since publication of [18], the Internet has undergone significant changes, shifting from a broadly hierarchical form to a flatter, more interconnected structure [16, 26]. Given the longitudinal nature of this paper and its focus on interdomain traffic in particular, the insights provided by these studies on the macroscopic effects of content consolidation are discernible within the studied dataset, and as such are a source of validation for many of the observations herein.

In addition to such general investigations, this paper is equally indebted to comprehensive work of a narrower scope. Significant portions of the observed traffic pertain to well-known applications which have been previously studied. Rao et al. [2] survey strategies used for video streaming at both Youtube and Netflix and characterise the properties of interleaved *block sending* patterns used to pace streams. These patterns are also the subject of [27], in which the burstiness of Youtube traffic in particular is found to result in considerable losses over residential connections. A large portion of the traffic observed in the MAWI dataset originates from Hyper Text Transfer Protocol (HTTP) file sharing services, commonly referred to as One-click Hosting (OCH) websites [3]. In [4], the authors study the characteristics of such traffic over a three month period, detailing the different throttling strategies used by different providers.

3. Dataset

This section provides an overview of the datasets used in this work and some of the data processing required before approaching the longitudinal study of Internet traffic rate limiting. The dataset used is composed from the original, un-anonymised, header-only traffic traces from the Measurement and Analysis of the WIDE Internet (MAWI) dataset [28], a set of daily packet captures from the WIDE backbone network which provides connectivity to universities and research institutes in Japan. Traffic is collected daily from 14:00–14:15 JST. Although this dataset extends back largely uninterrupted from late 2001, the present work focuses on just over five years of data following a network upgrade to the monitored link on October 2006. The monitored link carries mostly trans-Pacific commodity traffic between WIDE customers and non-Japanese commercial networks. Traffic towards WIDE is referred to as *inbound* traffic, whereas traffic originating from within WIDE is referred to as *outbound* traffic.

A preliminary overview of the dataset used is provided in Table 1. In total, 5.7 billion flows containing data are traced over five largely uninterrupted years; this represents approximately 30 terabytes of TCP traffic. For the purposes of this work, most analysis will focus on inbound traffic, 60% to 80% of which originates from port 80, referring only to analysis of outbound traffic when contextualising findings. Given the sender side plays a critical role in shaping traffic, analysing traffic for which the source is restricted to a small set of networks within Japan is of limited use in accurately depicting traffic trends at large. Hosts within Japan are instead fixed as traffic sinks,

Year	**Days**	**TCP data flows** ($\times 10^6$)	**Traffic (TB)**		**Unique** ($\times 10^3$)	
			In	Out	AS	Prefixes
2006	91	20.52	0.43	0.45	10.90	56.86
2007	350	102.56	2.11	2.49	17.21	113.79
2008	358	112.26	2.43	2.10	24.74	156.54
2009	364	113.97	2.48	2.53	19.71	143.87
2010	365	113.70	2.58	3.43	20.38	148.03
2011	358	114.74	3.44	5.14	19.99	140.56
Total	**1886**	**5777.55**	**13.50**	**16.14**	**34.12**	**341.22**

Table 1: Overview of traced MAWI dataset.

thus sharing a similar perspective on inbound traffic as many other similarly sized networks.

3.1. Tracing TCP Metrics

All TCP flows are reassembled and analysed for each daily trace. In addition to the five tuple used to define each connection, two additional restrictions are imposed: a contiguous sequence number space and a three minute timeout. These restrictions are helpful to deal with port reuse and unterminated flows respectively. Although the total number of TCP flows increased dramatically in 2011, the number of flows for which data payload was observed has remained stable, averaging over 100 million data flows traced per year.

There is much prior work with regards to reconstructing TCP flow from passive measurements and using this information to understand the end-to-end properties of traffic [29, 30, 31, 21]. However, the MAWI traces impose two constraints which require careful consideration, and ultimately imply the use of a custom TCP tracer. The first is the proportion of bidirectional flows, where both forward and reverse path are seen. In the dataset used this fluctuates between 40% and 60% over five years. Most available TCP tracers either ignore or are inadequate at processing unidirectional flows. The second is the short duration of each individual trace file. At only 15 minutes of line-rate data capture per day, it is wasteful to ignore flows which are not complete. Although the number of flows for which a SYN and FIN in either direction is observed has remained consistently high until late 2011, these flows are normally *mice*, i.e. flows that tend to be brief and which carry little traffic individually. In contrast, most *elephants* (flows that carry significant traffic individually) have durations that exceed that of each trace file.

Loss is inferred by accounting for *retransmissions* in the upstream data and *out-of-order packets* in downstream data; for the remainder of the paper the term *end-to-end loss* will refer to the sum of out-of-order and retransmitted data bytes over the total data bytes in a given direction. Anecdotally, this was found to be an adequate indicator of loss — with the exception of *hanging* TCP connections. In such cases where connectivity is lost, a host will proceed to retransmit packets while performing an exponential back-off. Although this results in negligible overall traffic, it can significantly skew the inferred loss ratio for uncommon destinations for which little traffic exists. To account for these cases, a 3-second timeout on retransmissions was imposed, after which the congestion feedback loop is considered to be broken.

Each daily trace in the dataset is processed from a packet level capture into a collection of flow level statistics, providing insight into the end-to-end characteristics of traffic. However, since a core objective of this work is to augment this time-based information with data describing the endpoints of each flow, aggregating by location is also required.

3.2. Aggregating by Location

Location information is added by mapping the original source and destination Internet Protocol (IP) addresses to its geographical and topological counterpoints. The routeviews archives [10] are used to reconstruct the mapping between each IP and both AS and network prefix; bi-hourly dumps of Border Gateway Protocol (BGP) Routing Information Bases (RIBs) are available in the WIDE archives since mid 2003. A daily RIB is reconstructed based on the views provided by contributing ASes, in particular IIJ and APNIC. Since there is no record of local policy, exact routes are not disclosed and as such there is no prior knowledge of the route taken by packets; this however does not hinder the ability to consistently map IPs to ASes. While discrepancies in AS destinations exist between different routeviews contributors, this happens almost exclusively on prefixes for which no actual traffic is seen.

Mapping IP to country is done through the use of GeoLite [32], a commercial geolocation database. While the accuracy of this solution is often disputed, locating traffic at a fine granularity is not a pressing concern. Most geographic emphasis will be placed on capturing macroscopic shifts in time at a national level, for which Geolite proves adequate. The archive for geolocation data only extends to 2009, before which the earliest match must be used. Additionally, the administrative mapping up until mid 2009 for a destination or source AS is verified to have remained the same in the relevant Routing Information Registrar (RIR) archives in order for a flow to be assigned a geographical location. After associating flows to country, region, AS and network prefix for both source and destination IPs, flow statistics are aggregated over each location identifier. This generates a daily collection of location identifiers and associated flow properties, from which the geographic and topological properties of the dataset can be sketched over time.

4. RTT estimation

Building on prior work presented in section 2, this section proposes an algorithm that scalably recovers the RTT from one-directional traffic traces. Although RTT estimation is a difficult problem, simplifying assumptions can be made. For the MAWI dataset most RTTs are relatively large, with the closest neighbouring country, South Korea, roughly 40ms away. By only processing bidirectional traffic from Japan, the expected RTT range can be reduced for all other traffic. The recovery mechanism then enhances the natural periodicity of traces and scalably constructs flights associated with specific application and protocol behaviour. In the following the mechanisms required by these two goals are described. In normal operation, many TCP operations involve cycles of data and ACK packets between two endpoints. A received data packet will immediately trigger an ACK. A received ACK will often trigger another data packet (since the amount of outstanding data has been reduced). Hence the process operates in a manner where the RTT T provides a natural *clock*. The most natural way to estimate RTT from TCP traces is to correlate data and ACKs exchanged in both directions. If only one direction of data is observed however, T cannot be directly observed. Instead, it must be estimated from the way in which TCP packets cluster in time in response to the data and ACK cycle.

The TCP congestion window (cwnd) determines the number of unacknowledged bytes that a TCP flow may maintain at any point in time. This can be referred to as *bytes in flight* because they are in transit between the sender S and the receiver R; an equivalent definition applies for the number of *packets in flight*. Once S has transmitted cwnd data bytes, it will refrain from transmitting more until either some bytes are acknowledged by R or cwnd is increased by the sender. In the absence of losses, neither of these events can happen until a TCP acknowledgement (ACK) is received; this immediately reduces the number of unacknowledged bytes, but may also lead to a significant cwnd increase (during e.g. *slow start*). In the presence of losses, however, bytes can be re-sent if a packet is timed out and considered lost; in this case, the number of unacknowledged bytes is reduced.

The main difficulty associated with one-sided TCP flow reconstruction is as follows. Let $t_1, t_2, \ldots$ be a set of times at which packets $p_1, p_2, \ldots$ were observed at S en route to R. Suppose that a packet p_j of size b is observed at time t_j. In addition, suppose that approximately one RTT T later, the sender S receives an ACK a_j from R for the b bytes of p_j. At this point, the TCP stack in S will decrease the number of unacknowledged bytes by b, thus opening the possibility for sending additional traffic to R. This can lead to another packet p_k to be transmitted; let this packet be observed at time t_k as it is sent towards R. Assuming that processing delay is insignificant, the RTT experienced by p_j can be approximated as $T \approx t_k - t_j$. Now consider what happens if packets are only observed in the $S \to R$ direction. Under such conditions, it is not possible to ascertain whether p_k was sent explicitly as a result of S receiving the unobserved ACK a_j, or whether it was sent as a result of an ACK a_i associated with a previous packet p_i rather than with p_j. If, however, a packet p_l is eventually observed that did result from the reception of a_j, the RTT can be estimated as $T \approx t_l - t_j$ with $t_l > t_k$. Following this same reasoning, approximately one RTT later a packet p_m will be observed for which $2T \approx t_m - t_j$; this can potentially continue for as long S has data to send and R continues sending ACKs. This is the underlying reason that RTT-related periodic regularities arise when considering the timestamps of observed packets [25].

The reasoning above is at the heart of the proposed algorithm to improve RTT recovery by enhancing packet stream periodicity. Assume that a packet p_j is observed at time t_j. Considering the set $\mathcal{T}_j$ of all values of $\Delta t = t_k - t_j$ for every $k > j$, it is apparent that it will include estimates not only for the RTT T, but also for all its multiples $2T, 3T, \ldots$ If $t_l - t_k \approx T$ and $t_k - t_j \approx T$ then it follows that $t_l - t_j = 2T$, and this value will also be included in $\mathcal{T}_j$.

By maintaining a set $\mathcal{T}_j$ for every packet p_j observed, at least some of its values will correspond to estimations of multiples of the RTT. It then follows that by creating a set $\mathcal{T}$ that includes values calculated starting from every packet p_j so that $\mathcal{T} = \cup_j \mathcal{T}_j$, numerous estimates for $2T, 3T, \ldots$ will also be included. Hence, the probability density function $H(t)$ of the values in $\mathcal{T}$ should show peaks around multiples of the RTT (see Figure 1).

The algorithmic recovery of T from $H(t)$ presents additional challenges. In particular, $H(t)$ may include a large number of RTT multiples, and a peak

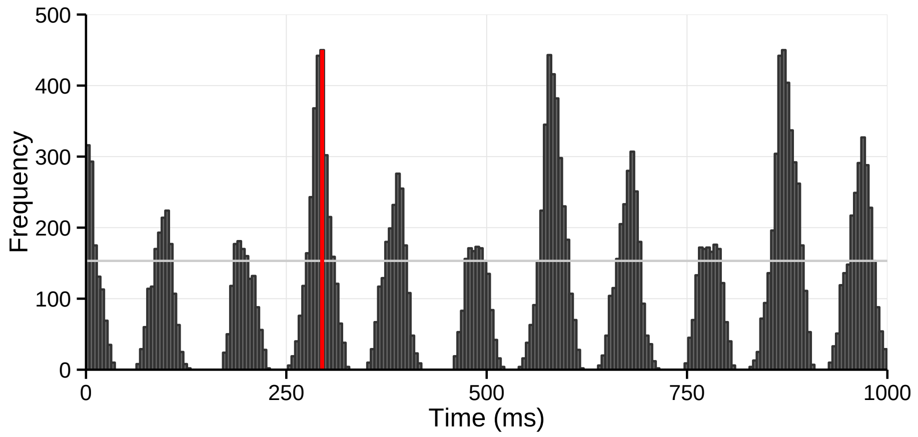


Figure 1: $H(t)$ for example flow. The horizontal line indicates $\overline{H}$ while the highlighted bin denotes highest peak which would be an initial (incorrect) RTT estimate.

will be found for all of them. Crucially, all these peaks may be of comparable magnitude, complicating the task of selecting a single peak. Moreover, these peaks need not be very pronounced, with histogram bins in close proximity of the peaks have very similar values as the peak itself. As such, taking RTT candidates directly from $H(t)$ may result in a large set of similarly-valued bins situated around a peaks at multiples of the RTT. For example, in Figure 1 the highest peak is at approximately 300ms (highlighted in red) but the actual RTT is around 100ms.

Three recovery algorithms for T are attempted. First, as a baseline, the highest peak in $H(t)$ is selected as a candidate for T. In addition, expanding upon the work of Qian et al. [25] a frequency-domain representation of $H(t)$ is used to identify T. This is done by selecting the highest peak of $|\hat{H}(\omega)|^2$, the *energy spectral density* of $H(t)$ (i.e. the norm squared of the Fourier transform of $H(t)$). Finally, a custom utility-based technique that operates directly on $H(t)$ is proposed which achieves superior performance to both of the aforementioned methods.

4.1. Utility-Based RTT Recovery

This method relies not on the identification of periodicities, but on explicitly matching experimentally found signatures. To this end, we consider the peaks of $H(t)$, which are then considered RTT candidates. However, trivial discriminators (such as simply selecting the highest peak) are not reliable. In this case, it was found experimentally that repeatable peaks and troughs also occur at multiples and sub-multiples of T, with the most important ones being $\frac{T}{3}$, $\frac{T}{2}$, T and $2T$. We design this detection algorithm around the idea that a given pattern of peaks and troughs can identify T.

If we define $\overline{H}$ as the mean height of $H(t)$, we can define a per-peak utility function $p(t)$ so that

$$p(t) = 1.0 - \exp\left(-2.0\left(\frac{H(t)}{\overline{H}}\right)\right).$$

This function has several advantageous properties: it is 0 if $H(t)$ is zero, 1 if $H(t)$ is infinite, and 0.5 if $H(t) = \overline{H}$. In other words it is a measure of the *peakiness* of the data, with $p = 1$ identifying an infinitely high peak, $p = 0$ identifying an empty histogram bin (trough), and $p = \frac{1}{2}$ implying that $H(t)$ is of exactly average height at that point. We can then score each candidate using the following utility function:

$$P(t) = 1.5p(t) + p(2t) - p\left(\frac{t}{2}\right) - p\left(\frac{t}{3}\right). \tag{1}$$

That is, the candidate RTT t scores highly if it is itself a peak, if it has a peak at a multiple $2t$, and if it also manifests troughs at sub multiples $\frac{T}{2}$ and $\frac{T}{3}$. The utility function here was a heuristic created by examining a large number of traces where the correct RTT was known and looking at candidate peaks. The following features were commonly observed:

- Peak height was the best indicator of being the "correct" RTT ($1.5p(t)$ term).

- A correct RTT candidate was always associated with peaks at multiples of the RTT ($p(2t)$ term.

- An incorrect RTT candidate peak was almost always a multiple of the correct RTT (for example the red peak in Figure 1). This possibility can be ruled out by decreasing the score for peaks at time t which are associated with peaks at $t/2$ or $t/3$ ($-p(t/2)$ and $-p(t/3)$ terms).

The system was not sensitive to the exact values of the coefficients (for example changing $1.5p(t)$ to $1.7p(t)$ made little difference to predictive power). Similarly, additional multiples and sub multiples were excluded as they showed very limited discriminating power experimentally.

The complete procedure for generating an estimated RTT for a given flow with packet arrival times $t_1, t_2, \ldots$ procedure is as follows.

- Collect the set $\mathcal{T}_j$ of all values of $\Delta t = t_k - t_j$ for every $k > j$.
- Place the values in this sets into bins to create a histogram $H(t)$ where $H(t)$ is the height of the histogram for an RTT candidate time t.
- From the values for $H(t)$ select a set of "peaks". This is a set of values of $H(t)$ that are local maxima. Call the values $\hat{t_1}, \hat{t_2}, \ldots$ candidate RTT values.
- For each candidate RTT value, $\hat{t_i}$ calculate the utility function $P(\hat{t_i})$ from (1).
- The value of $\hat{t_i}$ with the highest value of $P(\hat{t_i})$ is chosen as the RTT estimate.

Naturally it is important to test the robustness of these methods as, for example, we might suspect other clocks influenced the flow or a harmonic not the real RTT was identified. Figure 2 compares the three methods. For our comparison we use situations where the real RTT is known because bidirectional data is available and hence the true RTT can be directly measured. We use the three methods to calculate RTT and calculate the ratio of the estimated RTT to the true RTT (a ratio greater than 1 indicating an overestimate and less than one indicating an underestimate). It can be seen that the FFT based method often provides large underestimates and the maximum peak method often provides large overestimates. The utility based method provides fewer incorrect estimates – more than eighty percent of the time its RTT estimate is within ten percent of the correct value.

4.2. Comparing recovery algorithms

As described in section 4, $H(t)$ is calculated in such a way that RTT periodicity is amplified. This means that FFT-based techniques could potentially perform better on $H(t)$ than on the packet stream with no pre-processing. However, this is complicated not only because $H(t)$ contains periodicities at multiples of T, but also discontinuities that generate harmonics at frequency multiples of the RTT fundamental. Hence, although the FFT $|\hat{H}(\omega)|$ of $H(t)$ is much cleaner than that of the packet inter arrival time series on its own, its maximum peak rarely coincides exactly with the RTT clock (this corroborates reports by Qian et al. [25]). Thus, applying the FFT leads to another *peak detection problem* in which the RTT fundamental needs to be extricated

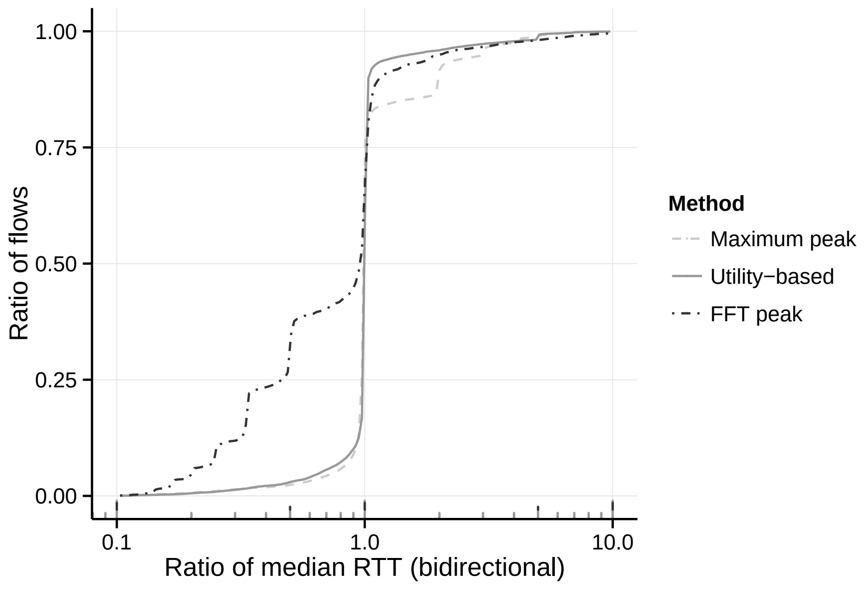


Figure 2: Accuracy of RTT estimator when compared to the median value of bidirectional estimate.

from its harmonics and sub-harmonics. The trivial solution to this problem, the application of a bandpass filter around the RTT frequency, is of course infeasible because the bandwidth and centring of such a filter depend on the RTT which is itself unknown. The utility-based algorithm described in Section 4.1 can hence be applied in either the time domain or the frequency domain; we chose to do it on the former on the interest of expediency and lower computational cost.

The performance of the analysed RTT recovery mechanisms is presented in Table 2, that shows the percentage of total flows below and above the RTT range given by the bidirectional estimates. We separate things for *inbound* traffic (where we are positioned at the receiver side) and *outbound* traffic (where we are positioned at the sender side). The utility-based algorithm is particularly useful to address RTT underestimation for flows over 10MB in size, which is our main objective since precisely that kind of estimation error would interfere with our ability to correctly decouple application behaviour from RTT-scale dynamics.

Sample data	Peak (%)		Utility (%)	
	Below	Above	Below	Above
Receiver side				
$flow < 10$MB	4.31	9.13	4.58	6.35
$flow > 10$MB	6.72	6.43	4.97	5.33
Sender side				
$flow < 10$MB	2.94	8.37	3.29	4.80
$flow > 10$MB	6.41	9.06	5.40	11.06

Table 2: Performance of peak-based and utility-based RTT recovery algorithms, showing proportion of flows for each sample dataset with estimates below and above bidirectional estimate.

5. Macroscopic traffic trends

Over a five year period, changes in routing and application popularity have continually redefined the nature of traffic under observation. This section provides a macroscopic view of these shifting trends. While these changes are specific to the data set observed they have importance to this study and explain some of the nature of the observations.

In 2008 a routing change diverted some inbound traffic from *national* sources away from the monitored transit link, resulting in a net reduction of traffic. In early 2009 routing changes saw an increase in *regional* traffic from Asian neighbours. These changes were reverted approximately six months later. During this period aggregate end-to-end loss rates increased as a result and most traffic was adversely affected by the increased utilisation. This suggests that the transit link itself may have been a bottleneck during this period. Finally, the impact of the Tohuku earthquake and tsunami in 2011 resulted in a noticeable break in demand coinciding with the start of the Japanese fiscal year in April, in which traffic traditionally ramps up.

5.1. Geographic distribution

Table 3 highlights the geographic distribution of both inbound and outbound traffic for the observed time period. The majority of traffic flows to and from the United States, which has increased its share of bandwidth in both directions over the past five years. The US accounts for almost 70% of inbound traffic in 2011. This should primarily be viewed as a reflection of routing policy, with regional traffic being diverted to alternate routes as Japan became increasingly interconnected to its neighbours.

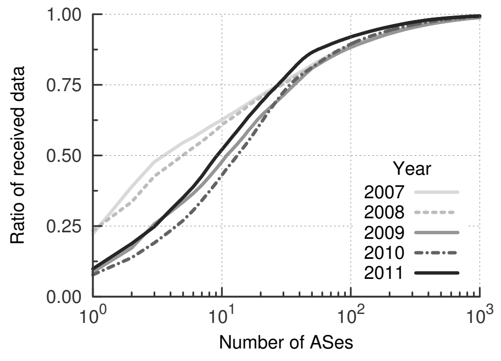
Breaking down US traffic by state reveals some changes in usage (see Table 3). The proportion of traffic originating from California has decreased over time, dropping from 55% of total US traffic in 2007 to only 35% in 2011. In its place, a larger set of states have emerged as content providers, with New Jersey, Florida and Virginia contributing over a quarter of all traffic originating within the US by 2011.

	Outbound traffic (%)					**Inbound traffic (%)**				
Country	2007	2008	2009	2010	2011	2007	2008	2009	2010	2011
United States	27.3	31.3	29.3	36.4	35.7	45.7	41.5	53.3	65.1	67.1
California	39.0	61.8	63.5	53.8	50.6	55.7	47.9	46.7	24.9	34.9
Texas	5.8	4.3	4.1	2.4	13.9	7.0	12.0	5.8	7.1	5.6
Colorado	1.9	1.2	0.6	8.5	2.8	4.9	6.0	5.9	9.7	5.8
Virginia	1.9	1.0	0.8	0.4	0.6	1.2	3.0	14.1	13.1	8.3
Washington	4.0	2.9	3.5	6.1	6.6	0.9	5.7	3.5	3.0	2.0
New Jersey	2.8	1.5	0.7	1.1	1.9	1.0	1.8	1.6	4.9	13.6
Massachusetts	1.6	1.1	0.9	6.1	4.9	5.4	2.1	1.8	1.6	2.0
Florida	3.1	2.3	1.3	1.1	0.9	1.0	0.4	0.4	8.5	7.9
Japan	11.6	15.4	17.7	16.7	16.1	33.8	32.2	7.3	8.1	11.5
China	7.9	20.5	17.8	10.3	5.9	2.5	5.3	6.3	4.6	3.1
Korea, Republic of	5.3	1.3	2.1	7.8	23.8	4.7	5.1	3.2	1.1	0.5
Germany	2.2	1.7	1.6	1.0	0.6	3.0	6.1	5.3	5.5	1.4
Taiwan	2.7	1.3	4.0	3.6	2.7	0.8	0.9	10.9	0.9	0.4
Netherlands	0.4	0.4	0.5	0.3	0.4	0.9	1.0	4.1	6.2	6.9
India	2.8	3.3	4.8	3.3	2.0	0.3	0.1	0.0	0.2	0.0
France	1.2	1.1	0.9	0.9	0.9	1.6	1.2	2.6	3.4	1.7
United Kingdom	1.1	1.0	1.0	0.9	0.7	2.5	2.2	1.6	1.3	1.3

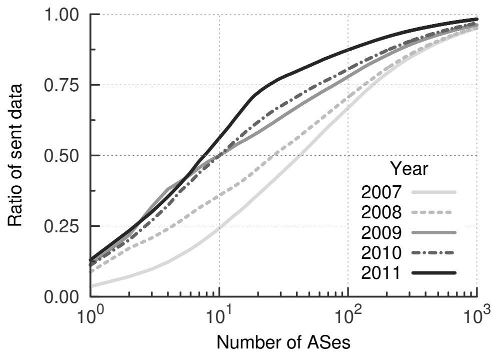
Table 3: Percentage of inbound and outbound traffic by country. U.S. state values are relative to total national traffic.

In the outbound direction a greater proportion of traffic flows toward Japan and China. Traffic to the Republic of Korea progressively increases from 2010 onwards but this is associated with successive routing changes. Combined with the drop in traffic towards China, this accounts for much of the drop in aggregate loss rates since 2010. European destinations overall have a small proportion of outgoing traffic and this proportion shrinks over the studied period. The discrepancy between outbound and inbound European traffic could be accounted for by the timezone when the measurements are taken 05:00GMT. This however does not account for why outbound traffic overall has been falling. Since most outbound traffic towards Europe at the time of measurement is likely to be scheduled transfers with no human intervention, a plausible explanation for this trend is the gradual shift away from file-sharing using peer-to-peer applications. This is further corroborated by the rise of hosting solutions which facilitate file-sharing, as shall become evident when analysing the breakdown of traffic by AS.

5.2. AS-level distribution



(a)



(b)

Figure 3: CDF of (a) inbound and (b) outbound traffic by AS.

It has been widely noted that inter-domain traffic has significantly changed over the past decade. An increasing proportion of traffic flows to and from a smaller set of content providers and consumer networks. Such traffic consolidation is often most apparent at the AS level where traffic becomes concentrated in a small number of AS (see, for example, [16]).

In the inbound direction traffic has remained consistently concentrated in the top 100 ASes accounting for approximately 90% of all data received, as shown by the cumulative distribution of inbound traffic by AS in Figure 3a. There is a visible drop in concentration amongst the top ASes between 2008 and 2009 as a wide range of Japanese prefixes were rerouted through a different ingress. This shift is clarified in Table 4, which lists the top ten ASes by received data for 2007, 2009 and 2011. While in 2007 traffic was already consolidated across a small set of ASes, a significant portion of transit traffic was still Asian: most traffic from NTT and Limelight originated from within Japan. Such traffic has gradually been pushed away from transit by 2011. Large carriers such as Cogent, Level3, Hanaro, China Telecom have also seen

ASN	AS name	%
2914	Ntt	29.92
36561	Youtube	15.89
15169	Google	3.80
22822	Limelight	3.70
174	Cogent	3.03
9318	Hanaro	2.46
3356	Level 3	1.97
20940	Akamai	1.53
19166	Acronoc	1.47
30212	DTI Services	1.05

(a) 2007.

ASN	AS name	%
3462	HiNet	9.78
15169	Google	8.64
43515	Youtube	7.92
2914	Ntt	5.89
46742	Carpathia LAX	4.17
4766	Korea Telecom	2.74
4134	China Telecom	2.62
3356	Level 3	2.14
4837	China Unicom	2.10
36561	Youtube	1.98

(b) 2009.

ASN	AS name	%
2914	Ntt	10.79
20473	Choopa	8.86
43515	Youtube	8.65
35415	Webazilla	6.01
40824	WZ Comm.	4.79
15169	Google	4.66
40263	FC2	3.61
30212	DTI Services	2.68
16265	LeaseWeb	2.56
29748	Carpathia (VA)	2.05

(c) 2011.

Table 4: Top 10 ASes for inbound traffic by year.

ASN	AS name	%
15169	Google	3.94
7132	Sbis AT&T	3.23
4134	China Telecom	3.14
10013	FreeBit	2.91
4788	TM Net	2.37
9318	Hanaro	2.21
9595	Xephion Ntt	2.09
9304	Hutchison AS	2.00
2914	Ntt	1.90
4837	China Unicom	1.72

(a) 2007.

ASN	AS name	%
15169	Google	11.55
2510	Infoweb	11.32
4134	China Telecom	9.10
2518	Biglobe NEC	6.00
3462	HiNet	3.78
4837	China Unicom	3.01
14778	Inktomi	2.04
7132	Sbis AT&T	1.51
36647	Yahoo	1.14
24560	Airtel	1.13

(b) 2009.

ASN	AS name	%
4766	Korea Telecom	11.39
15169	Google	8.66
2510	Infoweb	5.93
3549	Global Cross	5.38
9318	Hanaro	4.76
36647	Yahoo	4.63
2518	Biglobe NEC	4.23
46179	Mediafire	3.42
17858	Konyang Univ.	2.77
3462	HiNet	2.68

(c) 2011.

Table 5: Top 10 ASes for outbound traffic by year.

their importance diluted by ASes known to harbour OCH services such as Choopa, Webazilla, WZ Communications, Carpathia and LeaseWeb. Many of the hosted websites facilitate the distribution of copyrighted content, and as such are not capable of growing large enough to expand beyond hosted infrastructure without risking prosecution.

For outbound traffic, shown in Figure 3b, consolidation has been much more perceptible. For the top 10 ASes alone, the proportion of traffic has more than doubled between 2007 and 2011. By 2011, the distribution of traffic amongst ASes for inbound and outbound traffic bears a striking similarity but the nature of this concentration is markedly different, as made apparent by Table 5. Despite the increased importance of content providers and OCH services such as Google, Yahoo and Mediafire, significant portions of outgoing traffic continue to flow toward eyeball Internet Service Providers (ISPs) such as Korea Telecom, Infoweb, Global Crossing and Hanaro.

Overall, the observed traffic patterns match the insights provided by Labovitz et al. [16] on the changing nature of interdomain traffic but highlights that such trends have occurred at different paces depending on the direction of traffic. Inbound traffic showed strong signs of concentration as early as 2007, whereas outbound traffic has only become dominated by large consumer networks and regional providers more recently. The implications for transit traffic from an Asian perspective is less intuitive: with the increased adoption of Content Distribution Networks (CDNs) and Internet Exchange Points (IXPs), more transit traffic is being retrieved from further away as content in the United States shifts eastward (see Table 4).

5.3. Delay

Understanding where traffic flows to and from is of great value at an operational, commercial and often political level. This portrays a small part of a wider picture. End users care about delay not the location of content although the two are connected. Intuitively, delay should decrease over time: content is brought closer to the user with technologies such as CDN; more efficient routing reduces path stretch; and access technology improves, cutting down queuing delay. This is partially confirmed by Figure 4, which displays the mean delay distribution for traffic in either direction. Given the long-tailed nature of traffic, many ASes have a limited number of RTT samples. As such, only the thousand most significant ASes in either direction are used to plot the respective CDF.

As with traffic distributions, the plots once again illustrate the same overall trend in subtly different patterns. While latency has dropped across the board, the rate of improvement is markedly different. Taking the median of both plots as a reference point, delay has dropped by $20ms$ between 2009 and 2011 for inbound traffic, nearly half the equivalent decrease for outbound traffic. The absolute values in both cases are still disparate: over 90% of ASes are reached within a round trip time of approximately $400ms$ when ranked by inbound traffic, whereas the equivalent value for outbound traffic is almost $200ms$ higher. For inbound traffic the average RTT is low enough that geographical properties are clearly visible. A first plateau close to $100ms$ is apparent for traffic to the American west coast, while traffic to European destinations is clustered close to $250ms$. Tellingly, this second plateau seems to be receding. When taken in conjunction with the geographic distribution of traffic in Table 3 this seems to confirm the reduction in the number of sources within Europe.

A pertinent question at this point is in trying to understand how delay relates to traffic volumes. Given the different nature of stakeholders monopolising traffic at either end of the spectrum, what can be said about the evolution of delay in either case? Figure 5 plots the cumulative distribution of the average RTT weighted by the respective volume of traffic. In interpreting such plots one should keep in mind that they provide a rough indicator of the average delay to be expected if one were to sample a packet belonging to the top N sources or destinations. As N increases, the resulting value approaches the average RTT for all traffic in a given direction.

Inbound traffic by AS highlights an expected rise in delay between 2008 and 2009, as both NTT and Limelight are replaced by more distant sources. However, from 2009 onwards the overall delay remains remarkably stable. While the cumulative distribution function of RTT shows improvement in delay at the tail, this results in very little improvement overall as traffic is dominated by a handful of entities.

Two explanations emerge for this behaviour. The first stems from the changing nature of the traffic being sampled. While functionally NTT represents the same entity over time, the traffic under observation is very different. Examination of the traffic to NTT over time shows that the data has been increasingly exchanged over peering links, particularly at a national level. This is particularly noticeable in Figure 5a, where the average delay towards NTT, the most significant AS for inbound traffic in both 2007 and 2011, increased by approximately $100ms$. This does not represent a degradation in quality of service, but rather a change in where traffic is flowing from within the AS.

A further reason relates to the placement of content. As previously established in Table 3, there appears to be a migration of content away from California. OCH providers such as Lemuria, based in Florida, Mediafire, based in Texas and District of Columbia and Carpathia, based in Virginia, are all contained within the top 20 ASes and have shifted traffic further from locations which had traditionally benefited from low latency as viewed from Japan.

Analysing the outbound traffic seemingly reveals the opposite effect, with the average weighted RTT dropping by over 100ms. Once more, there is no single reason which accounts for this effect. A number of possible explanations exist. In 2007 many of the top destination ASes were in developing Asian countries, where infrastructure has improved greatly since. Improvements in routing to countries such as China and the Republic of Korea have also had a positive net effect. This is visible in Figure 6, where the average RTT aggregated by country is plotted. Countries for which RTT estimates are available for less than 50 days in a year are filtered out. Between 2007 and 2009 most Asian and European countries experience significant improvements in RTT. The minor exceptions are typically countries for which delay was already comparatively low. By 2011, latency to most countries had been reduced below $500ms$.

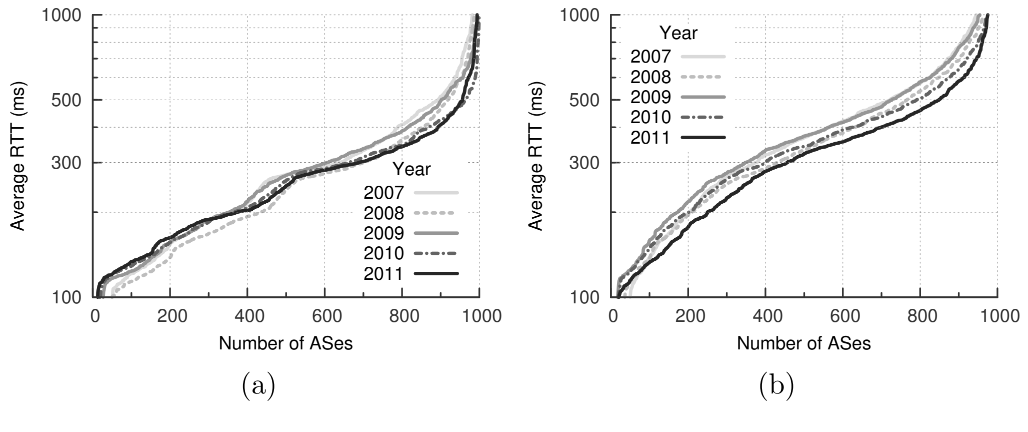
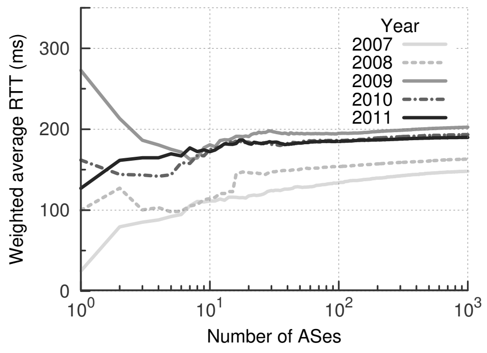
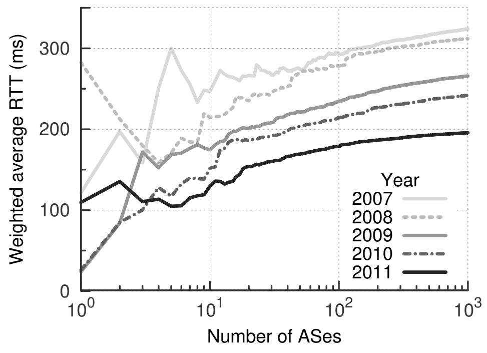


Figure 4: CDF of mean RTT by AS for (a) inbound and (b) outbound traffic.

Those companies that caused increasing RTT for inbound traffic, such as Mediafire or Ustream.tv, are also amongst the top destinations of traffic. It

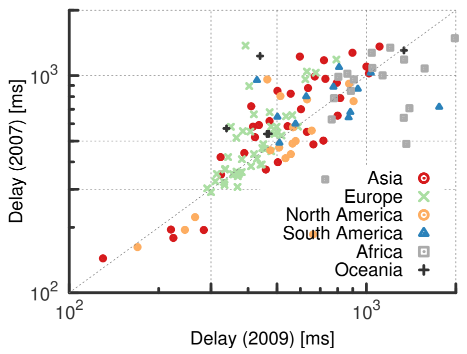


(a)

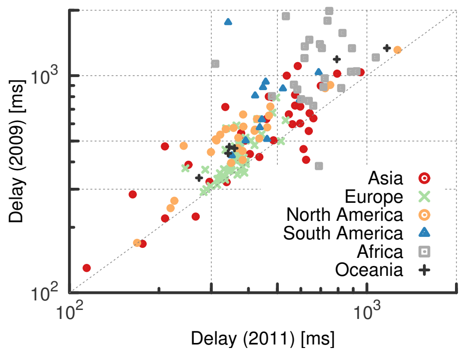


(b)

Figure 5: CDF of weighted RTT by AS for (a) inbound and (b) outbound traffic.



(a) 2007/2009.



(b) 2009/2011.

Figure 6: Scatter plot of mean RTT by country grouped by continent.

is interesting to note that as of 2011, data travelling from the top 1000 AS traffic sources was expected to experience the same latency as data travelling towards the 1000 most popular AS traffic destinations. In 2007, the value was two times higher for outgoing traffic. Traffic downstream is moving further away as content is not only placed closer to consumers and bypasses the transit link entirely, but also moves deeper within the United States, whereas traffic upstream has been drawn closer.

6. Flow classification

The aim of this section is to describe a process which distinguishes those flows which have their throughput limited by mechanisms other than the usual TCP response to loss and delay. Each flow can be characterised as being either application paced, in which the sending application is limiting the data provided, host limited, whereby local constraints at either end host cap throughput, or receiver shaped, in which an artificial constraint is imposed by a middlebox or receiver.

One fundamental precondition to decouple the influence that network loss, host configuration and TCP behaviour has on the throughput experienced by a flow is the reconstruction of the congestion window behaviour of TCP flows on the basis of observed data. Unfortunately, the congestion window value is internal to the sender's TCP state machine and may not manifest itself in the absence of sufficient data from the application layer. A more easily observed quantity which serves as a reasonable proxy for the congestion window is the number of unacknowledged bytes in flight, henceforth referred to as the *flight size*, which can be derived given an accurate estimate of the end-to-end delay. The evolution of both flight size and RTT can in turn be used to ascertain to what extent throughput is regulated by limitations imposed at different layers of the networking stack.

Given a stream of packets, the methodology presented in section 4 can derive a candidate RTT for a TCP flow. Given a candidate RTT, a stream of packets with arrival times $t_1, t_2, \ldots$ can be aggregated into a stream of *flights*. Intuitively, a flight is a clustered subset of a TCP flow which exhibits its own temporal coherence; alternatively, it can be thought of as a series of consecutive packets that were (roughly) generated by the sender as a response to the same protocol operation. A flight f_i that begins with the jth packet and ends with the kth is defined to have a *total flight time* $\tau_i = t_{k+1} - t_j$. The algorithmic selection of initial and final packets in such a way that the resulting flights are indicative of TCP behaviour remains an open problem. The RTT is assumed to provide a natural time frame for the operation of TCP. As such, given an initial packet π_j and an RTT estimate T, the kth (and final) packet is selected to minimise *the flight time error* $e_i = |T - \tau_i|$. This mechanism resembles the methodology described in [18], but where flights are not defined as being both adjacent and disjoint; rather, flows are decomposed into a stream of potentially overlapping flights. This helps the algorithm mitigate the deleterious effects of small deviations in the estimated RTT, which alters the properties of each flight. Furthermore, since the flight size is continuous in time, it makes little sense to restrict window reconstruction to a single sample per round trip time.

Having obtained flight information from each flow, the predominant factor that affects its throughput can be determined. Within the context of TCP, flows are classified as being artificially constrained by three distinct processes: *application pacing*, *host limited* and *receiver shaping* described in the next sections. Flows not fitting into these categories are assumed to be limited by standard TCP mechanisms or to not be limited in throughput (as, for example, would be the case with the large number of short flows that never suffer loss or delay. Fitting a flow into one of these classes is, by necessity, hard to do precisely. Some flows may be limited by more than one mechanism in combination with limitations from available bandwidth (standard TCP mechanisms). Some flows may be sufficiently short that they begin to show signs of being limited by a particular mechanism but only for part of the flow duration. No ground truth is available for the data and it is hard to imagine how such a ground truth could be found in real (not artificial) data. The approach taken by the authors is to observe a large number of traces and to construct heuristics that positively identify flows that are limited by the three identified processes. The heuristics chosen are selected to be a strong enough selection procedure that there would be little doubt that the identified mechanism is the primary mechanism for that flow. This implies that the figures given for percentages of flows limited by each mechanism are conservative under estimates of the flows affected by these mechanisms. (Flows only affected for a small part of their duration are ruled out.)

6.1. Application paced

A flow whose throughput decreases because it has no outstanding data to send is temporarily limited by the application. Flights can be identified as

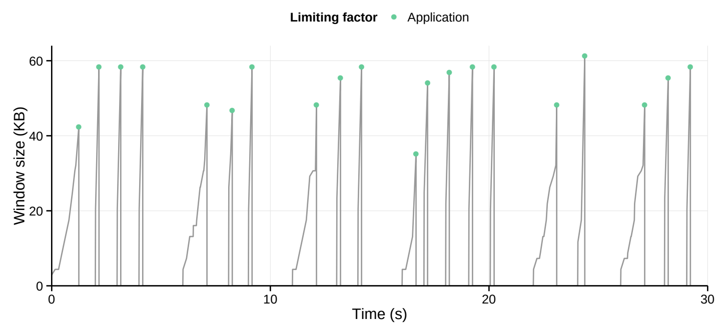


Figure 7: Congestion window over time for application paced flow.

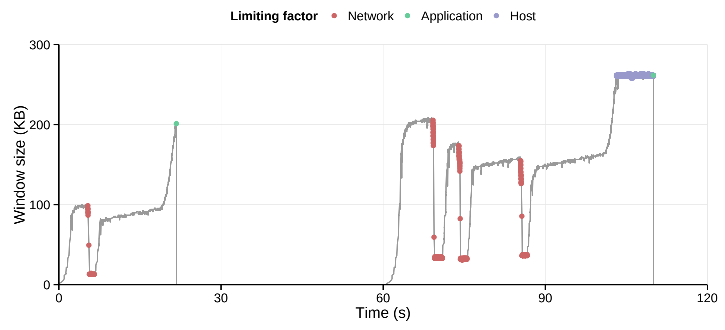


Figure 8: Congestion window over time for partially host limited flow.

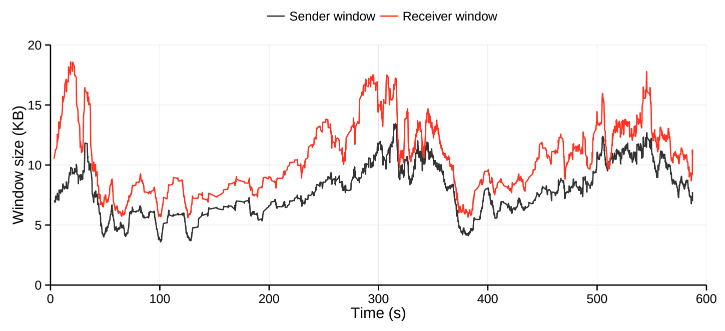


Figure 9: Congestion window over time for receiver limited flow.

being *application limited* if terminated with a packet smaller than the Maximum Segment Size (MSS) and followed by an inter-arrival time greater than the RTT, as consistent with [18]. The underlying reason for this definition is that most TCP implementations will wait some time for subsequent bytes to be written to the socket if the next packet to be sent is smaller than the MSS, unless the TCP_NODELAY option is set [33].

A flow with *application limited* flights however is not necessarily *application paced*. In practice, all flows for which the final packet is observed contain at least one such flight. For the purposes of this work however, the focus remains on identifying cases in which throughput is predominantly determined by application behaviour. One such example is illustrated in Figure 7, in which a stream is delivered by periodically writing blocks to the sending socket. The resulting network-level behaviour is distinct from traditional congestion control: short bursts are interspersed with protracted silence. Application limited flights, which terminate on non-MSS packets, are highlighted at the end of each burst.

This behaviour is in stark contrast to that exhibited in Figure 8, where distinct transfers are multiplexed on top of a single transport association over time. From the perspective of the network, there is little to distinguish the behaviour of such traffic from independent TCP flows. Application paced connections such as Youtube traffic however exhibit a degree of regularity which can potentially be exploited by the network in predicting demand or smoothing bursts.

In order to identify such recurring behaviour, flows are classified as being *application paced* if the period between bursts terminated by *application limited flights* is consistently under 10 seconds and the standard deviation of the intermediate pauses is under one second. This definition in particular purposely ignores flows which exhibit long silence periods due to user interaction, and follows closely the behaviour historically associated to Youtube streaming in particular.

6.2. Host limited

Given sufficient bandwidth and traffic to send, a flow may encounter local constraints at either end-host which cap its throughput. For instance, the buffer space allocated on both the sender and receiver side is often preconfigured, and it is common practice to tune these values down on popular servers and managed infrastructure in a bid to conserve memory or bandwidth. A receiver is also limited in the window size it can announce to the remote sender; if the windowscale option [34] is not negotiated during the TCP handshake, the advertised window cannot exceed 64KB.

In both cases, a local decision by either host can determine the upper bound of the flow rate. These *host limited* cases are characterised by a constant window size over time. The methodology described for flight aggregation at the beginning of this section typically generates a large number of flights, representing many likely combinations given a base RTT estimate. In order to identify the flat-lined behaviour of a host limited flow, the flight stream is first filtered to remove some of the uncertainty derived from small fluctuations in the RTT. The maximum flight size observed for each RTT interval is then selected, with a sequence of flights being classified as host limited if the same maximum was observed over six consecutive RTTs (this is twice the period suggested in [18]). In practice, increasing the period over which the maximum window size is tracked allows us to more accurately discern between host limited behaviour and more conservative bandwidth probing, such as that performed during the convex phase of TCP CUBIC [22].

A flow may be host limited for only brief periods of its lifetime, as illustrated in Figure 8. To filter out such cases where host limitations are not the predominant factor in defining flow throughput, a further requirement is imposed for a flow to be classified as being host limited: the average window size over a flow lifetime should be within 10% of the inferred host limit, which is not the case in Figure 8. In practice, flows can exhibit both application pacing and host limitations, with bursts being sent at a capped window size followed by application pauses. In such cases, a flow will still be classified as being *application paced* if it meets the requirements set out in the previous section, as doing so provides evidence that it controls throughput in spite of the degraded performance provided further down the stack. This line of reasoning applies equally to the occurrence of sporadic loss; so long as block delivery is ensured within the time frame dictated by the application, it remains in control.

6.3. Receiver shaped

A flow which is neither *application paced* or *host limited* can still be artificially constrained by flow control (rather than by congestion control). Traditionally, in TCP the sender is responsible for regulating throughput. However, the receiver can also shape throughput by manipulating the *advertised window* announced on every acknowledgement. Such receiver window auto-tuning has been available on Windows operating systems since Vista [35], and can also be leveraged by middleboxes to throttle inbound traffic [36]. To evaluate the potential impact of such behaviour, a further heuristic is proposed to identify receiver-shaped traffic. For flows in which both directions of traffic are observed it is possible to correlate the evolution of the advertised window with the size of reconstructed flights. Figure 9 displays an example of a receiver-shaped connection, in this case throttled by an intermediate middlebox. Since the advertised window may be fluctuating, it is not always obvious which of the many updates were effectively applied by the sender as successive values supersede each other. An example of a reconstructed flow which is subjected to receiver shaping is displayed in Figure 9.

For flows in which both directions are observed, it is possible to classify flights as being receiver-shaped if there is a statistically significant correlation between the advertised window size and the maximum flight size observed. Harnessing the stream filtering used in detecting host limited behaviour, such analysis is performed over a sliding window of 10 RTT intervals. A flight is flagged as being receiver shaped if the correlation between receiver window and flight size is statistically significant; a flow is considered to be predominantly receiver shaped if over half of its flights are flagged as such. This covariance analysis is not performed on flights which contain out-of-order or retransmitted packets. In such cases both the receiver and sender window sizes are correlated *by definition*: the receiver buffer will temporarily fill expecting the next packet in sequence while the sender will reduce its congestion window due to temporary setback in ACK clocking.

Given that receiver shaping classification requires correlating information in both directions of a TCP connection, it will come as no surprise that the absence of the reverse path can introduce false positives into our measurements. This happens because any given flow might be receiver shaped in such a manner that the heuristic erroneously attributes its behaviour to host limitations. In the absence of additional evidence, this misclassification is difficult to detect explicitly. Instead, the ratio of receiver shaped flows which would have been incorrectly identified is calculated for cases where the reverse path was not observed. This error rate can then be used to evaluate the accuracy of classifier results.

6.4. Classification heuristic

The complete heuristic to identify the throughput limitation comes from combining RTT estimation (see Section 4) with the heuristics from this section. The following procedure is then used to identify the limitation mechanism for that flow.

- Using the method from Section 4, calculate T the estimated RTT value.
- Find sequences of packets ending with a packet of size less than MSS – application limited flights. Calculate the lengths between the end of each application limited flight and the next packet (OFF periods for transmission).
- If the lengths of the OFF periods are less than ten seconds and the standard deviation is less than one second the flow is *application paced* and the classification terminates here.
- Split the flow into flights for each RTT interval.
- For each RTT interval for a flow calculate the maximum flight size.
- If the same maximum flight size is observed over six consecutive RTTs this size is inferred to be a potential host limit for the window.
- If the average window size observed for the flow is within 10% of this inferred host limit then the flow is *host limited* and the classification terminates here.
- Split the flow into sliding windows of width 10 RTT. For each of these windows calculate the maximum flight size and the advertised window size.
- If the correlation between maximum flight size and advertised window size over all such sliding windows is statistically significant then the flow is *receiver shaped* and the classification terminates here.
- Remaining flows are not classified as having limitation on throughput other than standard TCP mechanisms.

Year	Loss (%)	Limitation (%)			
		Application	Host	Receiver	**Total**
2007	1.29	49.47	18.58	0.55	**68.60**
2008	1.37	49.55	17.80	0.69	**68.04**
2009	1.44	47.10	14.50	2.57	**64.17**
2010	1.22	36.78	20.44	3.21	**60.43**
2011	0.82	46.10	13.49	0.60	**60.20**

Table 6: Percentage of traffic bytes affected by each constraint by year.

7. Revisiting assumptions

Having processed each daily trace individually, flow characteristics are aggregated longitudinally in order to trace the evolution of constraints affecting TCP across time and both spatial and topological dimensions, following the process described in section 3. The analysis in this section questions four commonly held assumptions regarding Internet throughput, for example those used for modelling in [18].

7.1. Throughput is primarily shaped by TCP

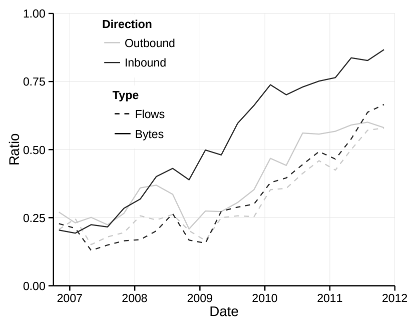
Internet flow rates are commonly viewed as the output of congestion control embedded at the transport layer. For example, models such as [37] and [18] work from the assumption that the throughput of a flow arises from properties of the network and the congestion control protocol interacting. There is an implicit assumption that the network is the bottleneck. Under such conditions, TCP acts as a distributed optimisation algorithm in allocating capacity to flows. Section 6, however, presented several cases where such an assumption does not hold. The prevalence of application pacing, host limitations or receiver shaping can all condition the accuracy of models which assume only elastic traffic adjusting to network conditions alone.

Table 6 displays the extent to which each of these limitations affects inbound traffic in the MAWI dataset over time. Traffic not found to be limited by one of the three mechanisms is assumed to either not have reached its throughput ceiling or to be limited by TCP effects. However, we cannot rule out other limiting effects on throughput that have not yet been identified. The bulk of the volume in bytes is either conditioned by host limits or application pacing. The use of receiver shaping on the other hand is both small in scale and temporally confined to 2009 and 2010. Over five years, the overall effect of the three selected constraints has dropped by close to 10%.

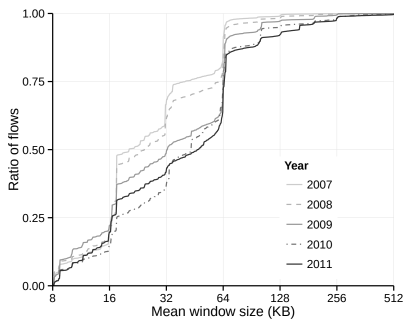
To understand where these dynamics stem from, Table 7 further breaks down these findings by autonomous system, listing the effect of each limitation for the five most significant traffic sources per year. Over the observed five years, traffic remains similarly consolidated: approximately 90% of all inbound traffic is sourced from the top 100 ASes. However, the weight of the most significant sources changes considerably. In 2007 and 2008, a considerable proportion of the traffic exchanged over the inter-domain link was content hosted within Japan (NTT, Limelight, see Table 4). From 2009 onwards, most of these local sources established peering connections, bypassing the observed link entirely. This accounts not only for the significant drop of traffic from NTT, but also its altered nature: after 2009 traffic from NTT travelled from further away and was less likely to be application paced.

As the weight of traditional carriers such as Cogent and NTT has waned, ASes known to harbour one-click hosting services such as Choopa, Webazilla, WZ Communications, Carpathia and LeaseWeb have gained significance (again see Table 4). Since many websites hosted in these ASes facilitate the distribution of copyrighted content, they have an incentive to continue using hosted infrastructure rather than deploying their own and risking prosecution. Furthermore, these domains are more likely to host applications which resort to capping the maximum window size as a means of throttling traffic. The increased weight of ASes which resort to these methods, such as Red Hat and Carpathia, significantly contributes to the unexpected increase of host limitations for 2010 displayed in Table 6.

Overall, these findings demonstrate that flow rates are not typically dictated by TCP congestion control alone. While the impact of application pacing and host limitations on rate control is decreasing, as of early 2012 60% of all inbound traffic was still subjected to either. The reduction of application pacing, however, may reflect the nature of the observed link, as many traditional streaming providers have migrated towards peering or CDNs, bypassing inter-domain links entirely. As such it is reasonable to expect the effect of application pacing to be more pronounced when considering traffic beyond transit. By 2011, successive capacity upgrades have led to a less congested network, but one where predicting how bandwidth is shared is harder due to the influence of stakeholders such as content providers and operating system vendors.

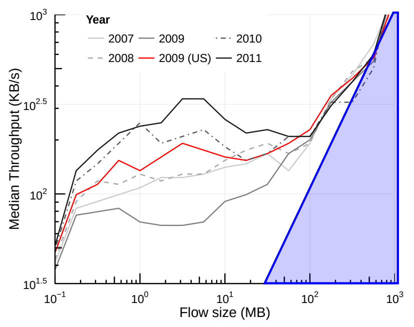


(a) TCP windowscale deployment over time.

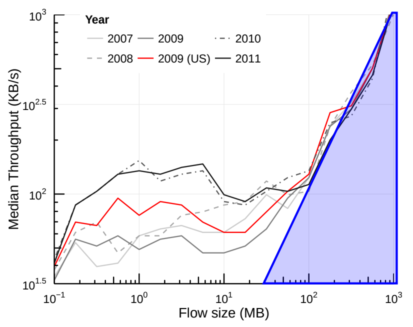


(b) CDF of window sizes for host limited flows.

Figure 10: Longitudinal evolution of TCP window parameters.

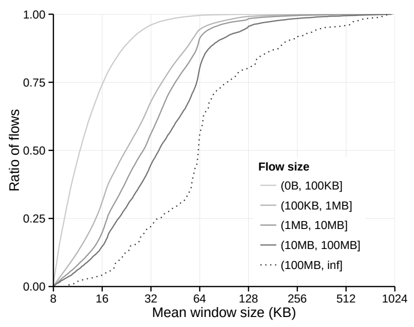


(a) Mean congestion window over mean RTT.

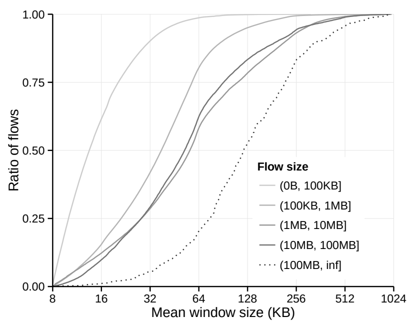


(b) Flow size over flow duration.

Figure 11: Median throughput for inbound traffic by flow size.



(a) 2007.



(b) 2011.

Figure 12: CDF of the average window size by flow size by year.

Year	ASN	AS name	Limitation (%)		
			Application	Host	Receiver
2007	2914	NTT	65.29	14.68	0.39
	36561	Youtube	77.41	11.19	0.11
	22822	Limelight	55.11	21.90	1.37
	15169	Google	24.11	10.29	0.08
	174	Cogent	47.65	32.22	0.76
2008	2914	NTT	62.40	13.71	1.03
	22822	Limelight	65.40	21.48	0.52
	36561	Youtube	72.48	12.16	0.09
	2518	BIGLOBE NEC	84.34	5.84	0.10
	15169	Google	52.98	17.13	0.18
2009	3462	HiNet	60.07	4.82	0.05
	15169	Google	74.79	12.16	0.02
	43515	Google (Youtube)	83.46	9.83	0.14
	2914	NTT	39.76	8.37	0.16
	46742	Carpathia (LAX)	41.04	48.01	2.03
2010	2914	NTT	21.80	4.91	0.00
	31976	Red Hat	9.62	41.63	0.00
	7366	Lemuria	51.95	15.72	5.85
	43515	Google (Youtube)	77.76	8.41	0.14
	46742	Carpathia (LAX)	33.06	42.71	4.21
2011	2914	NTT	50.33	8.19	0.18
	20473	Choopa	54.03	19.24	0.21
	43515	Google (Youtube)	69.71	7.56	0.16
	35415	Webazilla	40.02	11.23	0.95
	40824	WZ Comm.	42.08	17.43	0.05

Table 7: AS-level analysis of throughput limiting.

7.2. Throughput is primarily sender driven

A more widely held and less frequently enunciated assumption is that flow throughput is primarily determined at the sender side. However, TCP confers the receiver the ability to throttle rates through flow control. From answering the previous assumption, it is clear that throughput *is* mostly determined by the actions of the sender. However receiver shaping and host limitations together affect a significant minority of flows, up to 24% of all traffic in some samples studied.

A critical component in determining the upper bound for the congestion window size is the negotiation of the TCP *windowscale* option during the initial handshake. In its absence, a sender cannot have more than 64KB in flight. Furthermore, the *default buffer size* on either end of the connection can also limit the size to which the congestion window can increase. Both settings are primarily subject to operating system configuration. Throughput conditions on the receiver side improve as OS upgrades are rolled out. The installed user base within WIDE is comparatively stable over time, and as such is expected to exhibit continual improvements unless a significant OS rollback were to occur or a large set of users with outdated OSes were to be added to the network. This however is unlikely, and a more plausible explanation for any significant degradation in host limitations lies in macroscopic shifts in routing or application popularity which lead to a change in *where* traffic originates from.

This hypothesis is tested by first verifying windowscale deployment over time. Figure 10a shows the ratio of traffic and flows for which windowscale was successfully negotiated. Results are calculated solely over traffic where the initial handshake was observed. For added context, data for the outbound direction is also displayed. The first result that stands out is the steady increase over time of windowscale usage, rising from 25% of all inbound bytes in early 2007 to almost 80% by late 2011. Furthermore, the effects of content consolidation manifest themselves in the disproportionate coverage of bytes when compared to flows. With the reduced stake of large ISPs in inbound traffic, transit traffic has become dominated by a small set of centrally managed stakeholders such as Google, lowering the effective barrier for deployment of protocol extensions. Conversely, the temporary drop in windowscale adoption for inbound flows in 2009 is partly due to the increase of traffic from Asian sources, in particular HiNet (see Table 7) as these sources were observed to have a high prevailance of host windows beneath 16KB.

Given the prevalence of windowscaling, the primary source of host limitation should therefore be due to the configuration of socket buffer sizes. Figure 10b shows the distribution of the average window size for flows which are flagged as being host limited. While the 64KB limit intrinsic to TCP is a common upper bound on window size, other defaults are apparent and have shifted over time. The use of 16KB and 32KB buffer sizes (default buffer sizes for Windows XP and Vista respectively) was progressively phased out over the five year period. In addition to traditional power-of-two increments of the window size, different limits are apparent amongst hosting providers: 50KB, 100KB (The Planet), 160KB (Limelight) and 200KB (SoftLayer), reflecting the overall weight such ASes can have in shaping transit traffic.

Year	Total (%)		Receiver (%)	
	Flows	Bytes	Flows	Bytes
2007	3.86	20.31	70.82	74.33
2008	3.91	19.26	68.96	71.16
2009	2.60	15.29	61.54	62.81
2010	3.07	21.73	53.16	64.63
2011	1.76	14.11	51.27	60.82

(a) Inbound traffic.

Year	Total (%)		Receiver (%)	
	Flows	Bytes	Flows	Bytes
2007	8.22	24.24	76.50	82.54
2008	11.80	32.40	68.81	84.38
2009	10.40	30.50	62.50	84.28
2010	4.00	27.00	76.14	88.07
2011	3.01	25.94	72.00	85.91

(b) Outbound traffic.

Table 8: Percentage of host limited traffic over time by total number of flows and bytes. The proportion for which the receiver side was the bottleneck is also shown.

While Figure 10b demonstrates that host limitations for inbound traffic have been lifted over time, it still does not adequately answer on what side of the connection they are imposed. Table 8 breaks down the proportion of host limited traffic over time for both inbound and outbound direction. In addition to presenting the ratio of flows and bytes affected by host limitations, the relative proportion of traffic identified as being conditioned by the receiver side is also displayed. In either direction a very small fraction of flows are affected. Small flows are both numerous and unlikely to last long enough for window limits to be reached or reliably detected. The affected flows therefore tend to be large, and as such can translate into a significant amount of traffic. The proportion of flows and bytes for which the receiver side imposed the maximum window size dropped by 20% and 15% respectively over five years for inbound traffic, reflecting the successive OS upgrades performed for hosts within WIDE. Interestingly, these trends do not surface for outbound traffic: hosts outside Japan were consistently more likely to dictate the maximum window size. In part, this reflects the different nature of the traffic under observation: outbound traffic for this dataset is more geographically and topologically diverse, with content in many cases being retrieved from Japan by residential hosts from within Asia.

The endpoint which ends up dictating the maximum achievable throughput through flow control is typically a function of the OS adoption cycle. With the windowscale option covering 80% of all inbound traffic, the main source of host level constraints are now conservative buffer sizes. For this dataset, the amount of window scaling within WIDE has increased at a faster rate than those outside (see Table 8). This could indicate that upgrades within WIDE have been performed more rapidly. As such, throughput has become increasingly sender driven over time for inbound traffic.

7.3. Throughput is correlated with flow size

Given host limitations are most likely to affect large flows, it's worth considering whether other constraints are applied disproportionately across flow sizes. A commonly held assumption is that throughput is correlated with flow size, which has been verified empirically in previous studies [38, 18]. Much of the data used in these studies however precedes the widespread adoption of high bandwidth connections and use of streaming media, both of which can impact the extent to which contention occurs in the network.

Figure 11 shows the median throughput as a function of flow size, by year. In Figure 11a, flow throughput is calculated as the ratio between the mean TCP window size (in bytes) and the mean flight length (in seconds). Compared to the more commonly used ratio of flow size by flow duration, displayed in Figure 11b, this method is less susceptible to application behaviour and as such provides a more accurate estimate of the achievable rate. In both cases, flows are binned by size on a logarithmic scale, with median throughput calculated across each bin.

Because the data sets are broken into fifteen minute samples this implies a relationship between the mean throughput and the maximum flowsize that can be observed in a sample. For example, a flow with a constant throughput of exactly 10KB/s can transfer 90MB if it starts at the beginning of the trace and lasts for all 900s of the trace. No flows longer than this can be observed. In Figure 11 and 11a the light blue shaded area shows the forbidden region defined by this relationship. In fact some flows intrude slightly into this region for three reasons: the flows are "binned" as described; in Figure 11a the mean is, in fact, the ratio described above which may not be the mean throughput for the session; and in Figure 11 the median not mean is used. However, the important thing to note is that the observed data necessarily avoids this shaded region and therefore the right hand side of the graph is not a true representation of the relationship between throughput and flow length.

As noted in section 5 the first half of 2009 was marked by a temporary routing change and a greater proportion of traffic from Asian neighbours, particularly over smaller flow sizes. For reference the throughput for traffic from the US alone is plotted for 2009,in which case a more natural yearly progression becomes apparent. For both plots, a clear disparity is visible across flows sizes: for flows in the 10MB to 20MB range, although throughput has consistently increased with time, it has done so at a lower pace than for flows under 10MB.

These results, therefore, confirm the idea that for small flows (<1MB), flow rate is correlated with flow size. This is as expected given the slow start mechanism, short flows should stay at a slow speed. For medium sized flows 1MB to 20MB, there is no clear correlation between flow size and flow rate. This is an unexpected finding and different to findings from previous research [38, 18]. Above 20MB the sampling problem because of the 15 minute samples becomes a problem and our data cannot tell us anything about the relationship between flow length and throughput.

The relationship is expounded by further analysing the average window sizes across flow sizes, displayed in Figure 12. In 2007, there is a visible correlation, with larger flows attaining higher window sizes. Furthermore, the distributions cluster prominently around 64KB due to a low rate of windowscale negotiation. By 2011, this clustering is less pronounced, with window sizes increasing across the board, but with larger flows often out-paced by shorter counterparts.

Clearly, the extent to which rates are constrained is closely tied to flow size. Table 9 breaks down the results from Table 6 by flow size. Many small flows on the other hand never exit slow start, in which case none of the studied constraints will be reached or readily identified. This dichotomy is reflected on loss rates, which will be higher for flows regulated by TCP congestion control. Additionally, the discrepancy in loss rates is further exacerbated by geographic properties: traffic exchanged over poor infrastructure tends to be smaller, with flows from China exhibiting particularly high end-to-end loss rates.

These results suggest that network upgrades are unlikely to improve performance for significant proportions of traffic. This is most visible in Figure 11, where improvements in capacity for coping with higher bursts of activity (Figure 11a) has out-paced the actual delivery rate set by

Year	Loss (%)	Limitation (%)			
		Application	Host	Receiver	**Total**
2007	1.90	14.65	5.45	0.10	**20.20**
2008	2.27	14.99	5.37	0.09	**20.44**
2009	2.39	15.66	3.83	0.55	**20.03**
2010	2.15	10.55	4.18	0.36	**15.09**
2011	1.19	11.13	2.53	0.05	**13.71**

(a) Flows under 10MB.

Year	Loss (%)	Limitation (%)			
		Application	Host	Receiver	**Total**
2007	0.96	61.62	23.07	0.71	**85.40**
2008	0.88	61.49	21.94	0.92	**84.35**
2009	0.98	57.86	17.70	3.28	**78.85**
2010	0.71	43.97	24.45	4.03	**72.45**
2011	0.62	52.95	15.55	0.71	**69.21**

(b) Flows over 10MB.

Table 9: Percentage of traffic in bytes affected by each constraint by year according to flow size, along with aggregate retransmission ratio.

applications (Figure 11b). Given the popularity of emulating a constant bit rate service over TCP, that no such abstraction is provided at the socket level API is unfortunate.

8. Conclusions

This paper considers mechanisms that influence TCP throughput above and beyond the traditional well-known TCP control loops. Mechanisms identified in this paper are: *host limiting*, *application pacing* and *receiver shaping*. These three mechanisms together with traditional TCP control govern the throughput available to TCP. In the observed data over *half* of all inbound TCP traffic is governed by one of these three mechanisms and not traditional TCP flow control. We can only speculate exactly how typical the data set studied is, however, this makes it more important that the tools and techniques used in this paper are employed on wider and more recent data sets.

Various developments have improved throughput over the period of the study. OS upgrades have reduced the impact of host limitation. In addition the network under study has been improved resulting in significantly improved throughput over the duration of the study. This is particularly notable for smaller flows. However, evidence of throughput limiting effects independent from available end-to-end capacity was also uncovered. In the data set studied the majority of the traffic is primarily governed by mechanisms not related to loss, delay or measured of the network itself. There is significant evidence of traffic being shaped by modification of the receiver advertised window in a bid to curtail congestion. Application-driven rate limitation alone accounted for 40% of all inbound traffic observed in 2011. In summary, the conventional model that TCP throughput is driven by congestion control algorithms responding to loss and/or delay no longer appears true for the majority of traffic in this study.

[1] F. M. Ramos, F. Song, P. Rodriguez, R. Gibbens, J. Crowcroft, I. H. White, Constructing an IPTV Workload Model, in: Proc. of ACM SICOMM Poster Session, 2009.

[2] A. Rao, A. Legout, Y. Lim, D. Towsley, C. Barakat, W. Dabbous, Network characteristics of video streaming traffic, Proc. CoNEXT.

[3] D. Antoniades, E. P. Markatos, C. Dovrolis, One-click hosting services: a file-sharing hideout, in: Proc. of ACM SIGCOMM IMC, 2009.

[4] J. Sanjuàs-Cuxart, P. Barlet-Ros, J. Solé-Pareta, Measurement based analysis of one-click file hosting services, Journal of Network and Systems Management 20 (2) (2012) 276–301.

[5] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, Measurements, Analysis, and Modeling of BitTorrent-like Systems, in: Proc. Internet Measurement Conf, 2005.

[6] R. Braden, Requirements for internet hosts, RFC1122 (1989).

[7] P. Kanuparthy, C. Dovrolis, ShaperProbe: end-to-end detection of ISP traffic shaping using active methods, in: Proc. Internet Measurement Conf, 2011, pp. 473–482.

[8] J. Araujo, R. Landa, R. Clegg, G. Pavlou, K. Fukuda, A longitudinal analysis of internet rate limitations, in: Proc. INFOCOM, 2014, pp. 1438–1446.

[9] CAIDA, Cooperative Association for Internet Data Analysis, http://www.caida.org/.

[10] Routeviews, Project webpage, http://www.routeviews.org/.

[11] R. Oliveira, B. Zhang, D. Pei, L. Zhang, Quantifying path exploration in the internet, IEEE/ACM Transactions on Networking 17 (2) (2009) 445–458.

[12] N. Hu, L. Li, Z. Mao, P. Steenkiste, J. Wang, Locating internet bottlenecks: Algorithms, measurements, and implications, Proc. SIGCOMM (2004) 41–54.

[13] R. Fontugne, P. Borgnat, P. Abry, K. Fukuda, MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking, Proc. CoNEXT (2010) 8.

[14] B. Ager, W. Mühlbauer, G. Smaragdakis, S. Uhlig, Web content cartography, in: Proc. Internet Measurement Conf, 2011, pp. 585–600.

[15] A. Dhamdhere, C. Dovrolis, Twelve years in the evolution of the internet ecosystem, IEEE/ACM Transactions on Networking 19 (5) (2011) 1420 – 1433.

[16] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, Internet inter-domain traffic, SIGCOMM Comp. Comm. Rev. 40 (4) (2010) 75–86.

[17] K. Cho, K. Fukuda, H. Esaki, A. Kato, Observing slow crustal movement in residential user traffic, Proc. CoNEXT (2008) 12.

[18] Y. Zhang, L. Breslau, V. Paxson, S. Shenker, On the characteristics and origins of internet flow rates, SIGCOMM Comp. Comm. Rev. 32 (4) (2002) 322.

[19] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, Inferring TCP connection characteristics through passive measurements, Proc. INFOCOM 3 (2004) 1582–1592.

[20] K. Lan, J. Heidemann, A measurement study of correlations of internet flow characteristics, Computer Networks 50 (1) (2006) 46–62.

[21] S. Shakkottai, R. Srikant, N. Brownlee, A. Broido, K. C. Claffy, The RTT distribution of TCP flows in the internet and its impact on TCP-based flow control, Tech Report Cooperative Association for Internet Data Analysis (CAIDA).

[22] S. Ha, I. Rhee, L. Xu, CUBIC: A new TCP-friendly high-speed TCP variant, ACM SIGOPS Operating Systems Review 42 (5) (2008) 64–74.

[23] B. Veal, K. Li, D. Lowenthal, Passive online RTT estimation for flow-aware routers using one-way traffic, Proc. Passive Active Measurements Conf (2005) 121–134.

[24] R. Lance, I. Frommer, Round-trip time inference via passive monitoring, ACM SIGMETRICS Performance Evaluation Review 33 (3) (2005) 32–38.

[25] F. Qian, A. Gerber, Z. Mao, S. Sen, O. Spatscheck, W. Willinger, TCP revisited: a fresh look at TCP in the wild, Proc. Internet Measurement Conf.

[26] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, Anatomy of a Large European IXP, Proc. SIGCOMM.
URL `http://www.eecs.qmul.ac.uk/~steve/papers/ixp-sgcm.pdf`

[27] S. Alcock, R. Nelson, Application flow control in youtube video streams, SIGCOMM Comp. Comm. Rev. 41 (2) (2011) 24–30.

[28] K. Cho, K. Mitsuya, A. Kato, Traffic data repository at the WIDE project, in: Proc. USENIX ATC, 2000.
URL `http://dl.acm.org/citation.cfm?id=1267724.1267775`

[29] H. S. Martin, A. McGregor, J. G. Cleary, Analysis of internet delay times, in: Proc. Passive Active Measurements Conf, 2000.

[30] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, Measurement and classification of out-of-sequence packets in a Tier-1 IP backbone, IEEE/ACM Transactions on Networking 15 (1).

[31] S. Rewaskar, J. Kaur, F. Smith, A performance study of loss detection/recovery in real-world TCP implementations, Proc. IEEE ICNP.

[32] MaxMind, MaxMind GeoLite City, http://www.maxmind.com/app/geolitecity (2012).
URL `http://www.maxmind.com/app/geolitecity`

[33] J. Nagle, Congestion control in IP/TCP internetworks, RFC896 (1984).

[34] V. Jacobson, R. Braden, D. Borman, Rfc1323: TCP extensions for high performance, RFC1323 (1992).

[35] Microsoft, Description of the Receive Window Auto-Tuning feature for HTTP traffic on Windows Vista-based computers, http://support.microsoft.com/kb/947239.

[36] AppEx Networks, AppEx IPEQ (IP End-to-End QoS), http://www.appexnetworks.com/white-papers/IPEQ.pdf.

[37] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling tcp throughput: A simple model and its empirical validation, SIGCOMM Comp. Comm. Rev. 28 (4).

[38] R. G. Clegg, J. T. Araujo, R. Landa, E. Mykoniati, D. Griffin, M. Rio, On the relationship between fundamental measurements in TCP flows, in: Proc. of IEEE ICC, 2013.