Institut Eurécom
Department of Corporate Communications
2229, route des Crètes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-07-204

Passive Capacity Estimation: Comparison of existing tools

October 15th, 2007

Taoufik En-Najjary, Guillaume Urvoy-Keller

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : {ennajjar,urvoy}@eurecom.fr

[1]Institut Eurécom's research is partially supported by its industrial members: BMW Group Research & Technology - BMW Group Company, Bouygues Téécom, Cisco Systems, France Télécom, Hitachi, SFR, Sharp, ST Microelectronics, Swisscom, Thales.

Abstract

Measuring paths capacity in the Internet is a problem that has received a significant attention from the research community. Three options exist to estimate capacities: the active method, where measurement packets are sent in the network; the embedded method, where the application traffic is altered so as to give it a desired pattern; and the passive method that aims at extracting paths capacity from traffic traces. The active method is the most accurate, but also the most intrusive. The embedded method is useful for some specific applications, e.g., content distribution. The passive method is the only option in a number of scenarios, like the case of an ISP that wants to troubleshoot the performance of its customers.

In this paper, we introduce PPrate, a new passive capacity estimation tool, and compare its performance with the two other existing passive tools devised so far, namely Nettimer and MultiQ. We first compare those three passive tools to one state-of-the-art active tool, Pathrate, using Planetlab. In this specific environment, the three passive tools tend to behave in a similar way, though MultiQ is more prone to generate abnormally high estimates. We next investigate the performance of the passive tools in a more realistic and challenging environment, namely an ADSL platform. In this scenario, working with the TCP ack streams should uncover the ADSL modem downlink capacity, while the TCP data streams should uncover the servers capacity in client/server applications like FTP and HTTP. Our study reveals that Nettimer is unable to work properly on ack streams and tends to underestimate path capacities when working on data streams. In contrast, MultiQ works on both ack and data streams but tends to overestimate path capacities. Overall, PPrate offers a good compromise in most situations.

1 Introduction

Estimating the capacity of an Internet path[1] is an important issue that has received considerable attention in the last few years. Knowledge of the path capacity can be useful in many cases. ADSL providers can estimate the actual capacity of their customer, which is a function of both their subscription rate and the length of the line from the customer premise to the ADSL concentrator. More generally, an increasing trend for ISPs is to monitor and troubleshoot the performance of their users. Estimating the capacity of a path is, in this context, the first step toward locating the bottleneck of a connection [24]. Some applications might also benefit from the knowledge of the capacity of a path. Especially, multimedia servers would be able to select an appropriate codec and streaming rate. Researchers might also be interested in carrying a large scale study of Internet path characteristics [11] or simply to extract capacity information from the numerous publicly available traces.

So far, a large number of tools and techniques have been proposed to estimate the capacity of an Internet path [12, 18, 6, 8, 2, 22, 16, 14]. Those tools can be classified in three categories. First, active tools that rely on the injection of measurement packets in the network. This category encompasses the majority of existing tools like Pathchar [12], Pathrate [6] or Capprobe [14]. The second category consists of so-called embedded tools. Those tools alter the pattern of users traffic to estimate capacity without introducing additional measurement traffic. A typical example is TFRC Probe [4], which is an embedded version of CapProbe [14]. The third category consists of purely passive tools that aim at extracting capacity estimates from passively collected traces of traffic. To the best of our knowledge, two passive capacity estimation tools have been proposed so far: Nettimer [16] and MultiQ [15].

From an end-user's perspective, using active measurements to estimate the capacity of a network path is a reasonable approach. However, for a number of other scenarios, e.g. monitoring the performance of large set of users or extracting capacity estimates from a trace, a passive approach is the only option.

The focus of this paper is on the passive estimation of capacity. We present PPrate, a tool that passively measures the capacity of a path. PPrate is based on algorithms similar to the ones of Pathrate [6], which is a popular active capacity estimation tool. PPrate works on TCP traces collected at any measurement point in the network. The reason why we focus on TCP traffic is twofold. First, TCP has been adopted by the majority of new applications, e.g., p2p file sharing or podcasting, and thus remains the most popular transport protocol in the Internet today. Second, TCP is a natural candidate for capacity measurement techniques based on the packet dispersion principle (see Section 2) since TCP often injects packets in pairs in the network, due to the delayed acknowledgment strategy.

Contributions of this work are as follows:

[1]The capacity of a path is formally defined as the maximum IP-layer throughput that a flow can get on that path and is determined by the link with the minimum capacity among all links on a path. This link is called the *narrow link* of the path.

- We demonstrate that PPrate can work either on packet or *ack* inter-arrival times to investigate the capacity of a path, and requires only a few hundred kbytes for its estimation.

- We compare PPrate, Nettimer and MultiQ to Pathrate using Planetlab. Pathrate, being an accurate active tool, is used here as a benchmark. We show that all passive tools perform similarly on Planetlab, even though MultiQ tends to return occasionally abnormally high values and is less consistent than the two other passive tools.

- We show that in a more challenging and realistic environment, like an ADSL plateform, Nettimer is able to work on data streams only and tends to underestimate path capacities. We further observe that MultiQ continues to generate too high estimates for about 10% of the connections.

- We revisit the conventional wisdom that states that the dominant mode is often not the capacity mode in the histogram of packet pair inter-arrival times. We demonstrate on an ADSL and an Ethernet (campus) trace[2] that this conjecture can be over-pessimistic. We observe for this trace that the dominant mode is the capacity mode for 70% of the cases. Note, however, that a proper filtering of the data is crucial to obtain the above result; otherwise, the above percentage falls to 30%. In addition, for the 30% of cases where the capacity mode is not the dominant mode, large under and over estimations can be made.

- We evaluate the impact of compression on our example ADSL trace. Especially, we show that the fraction of packets compressed to values higher than the capacity of the path can be relatively large, though the speed-up factor (ratio between the observed packet inter-arrival time and the one dictated by the narrow link) is in general smaller than 2.

The rest of this paper is structured as follows. In Section 2, we survey the related work about capacity estimation. In Section 3, we introduce the notion of packet pair dispersion and provides an overview of Pathrate. We explain PPrate in details in Section 4. We compare all existing passive capacity estimation tools, namely PPrate, Nettimer and MultiQ with Pathrate using Planetlab in Section 5. In Section 6, we further compare the passive tools for traces collected on an ADSL platform. We discuss the impact of load on the weight of the capacity mode in the packet pair dispersion histogram in Section 7. A study of compression is provided in Section 8. We conclude in Section 9.

[2]The reason why we considered this campus trace and not only the initial ADSL trace is because (i) the former exhibits a average load of 60% at the link where measurement is performed whereas the load was only a few percent for the ADSL, and (ii) the conjecture concerning the dominant peak is more likely to be true for a loaded path.

2 Related Work

Most of the proposed capacity estimation schemes are based on the packet-pair dispersion principle. A packet pair, which consists of two packets sent back-to-back into the network, is dispersed at the narrow link according to the link capacity. If the packet pair reaches its destination experiencing no other perturbation, identification of the narrow link capacity is straightforward (see Section 3). In practice, due to cross traffic, expansion or compression of the packet-pairs might occur before or after the narrow link. Expansion leads to under-estimation and compression leads to over-estimation of the narrow link capacity.

Paxson [19] shows that the distribution of the packet pair dispersion can be multi-modal, and proposes the *Packet Bunch Modes* (PBM) technique to select a capacity estimate from these modes. More recently, the packet pair technique has been largely revisited, explaining the multiple modes that Paxson observed based on queuing delay and cross traffic effects [1, 18, 15].

To eliminate the effect of cross-traffic, various refinements have been proposed, including sending packet trains of various sizes, and filtering techniques to discard incorrect samples [2, 19, 16, 6]. The filtering is complicated by the multi-modality of the distribution of the packet-pair dispersion, and the observation that the dominant mode may not correspond to the capacity [6].

An alternative to the packet pair/train approach is to infer the narrow link capacity from the relationship between packet size and delay. Such an approach is used by pathchar [12], Clink [8], and pchar[18]. However, delay measurements rely on ICMP time-exceeded messages from routers, which limits both the applicability and the accuracy of these tools. For a comparison of these tools see [9].

Recently, an hybrid tool called CapProbe [14] has been proposed. It combines information on delays and dispersion of packet pairs to filter out samples distorted by cross-traffic. The key difference with the techniques mentioned above is that CapProbe uses delays only as an indicator of which packet pair sample to choose for estimating the path capacity. The authors have shown that this simple technique leads to results similar to the ones of Pathrate [6]. In [21], they proposed TCPprobe, an embedded version of CapProbe. TCPprobe needs either some modifications of the TCP protocol, or must be connected to both ends of the monitored path, whichever limits the applicability of this approach.

Nettimer [16] was the first tool to tackle the problem of passive estimation of path capacity. Recently, MultiQ[15] was proposed. MultiQ is a tool to passively discover the capacity of the narrow link either from sender or receiver side traces. Both tools work on the histogram of packet pairs and make use of advanced kernel density estimation techniques to estimate paths capacity.

3 Background

In this section, we first describe the packet pair technique on which a number of capacity estimation techniques such as Pathrate are based. We next give a short description of Pathrate.

3.1 Packet Pair dispersion

Formally, the dispersion of a packet pair can be described as follows. Consider a network path $\mathcal{P}$ defined by a sequence of link capacities $\mathcal{P} = C_0, C_1, \ldots, C_P$, where C_0 is the capacity of the sender and C_P the capacity of the last link before the receiver. Let i_0 be the index of the narrow link of the path. A sender emits a pair of probe packets (packet pair) back-to-back, each of the same size L. The dispersion Δ_i of the packet pair after link i is the time interval between the complete transmission of the first and the second packet at link i.

Assuming no cross traffic on the path, $\Delta_i = \max(\Delta_{i-1}, L/C_i)$, with $\Delta_0 = L/C_0$. In addition, we obtain that $\Delta_i = \Delta_{i_0}$ for $i \geq i_0$ as by definition, $C_{i_0} = \min_i(C_i)$. The narrow link capacity can then be calculated as: $C = L/\Delta_{i_0}$. In practice, interference with cross-traffic invalidates this straightforward computation method: Cross-traffic can cause compression or expansion, and subsequently, an under or over-estimation of the capacity.
Compression can happen when the narrow link is not the last link in the path. If the first packet of a pair queues at a post-narrow ($i > i_0$) link, while the second experiences queuing for a shorter time period than the first one, the dispersion between the packets decreases and the capacity is then over-estimated. On the other hand, if the dispersion of the packet pair at the destination is larger than the one introduced at the narrow link, the capacity is under-estimated. An increase of the dispersion may happen when cross-traffic packets are inserted in between the probe packets. Such a phenomenon can happen anywhere on the path, before, at, or after the narrow link.
The dispersion Δ_P observed at the receiver side varies when we repeat the experiment many times. The capacity values $C = L/\Delta_P$ will thus form a certain distribution B. The challenge is to infer the true path capacity $\hat{C}$ from this distribution.

It may seem at first sight that using packet trains, instead of packet pairs, makes the capacity estimation more robust to random noise caused by cross-traffic. However, Dovrolis et al have shown that this not the case [6]. Indeed, the use of packet train leads to the estimation of the so-called "Asymptotic Dispersion Rate (ADR), which lies between the available bandwidth and the capacity of the path [6].

3.2 Pathrate

Pathrate is based on the dispersion of packet pairs and packet trains. To circumvent the problem of multi-modality of the distribution of the packet-pair dispersion, Pathrate uses packet pairs to uncover a set of possible "capacity modes". It further uses long packet trains to estimate the ADR and to select the capacity mode. In fact, two methods have been successively used in Pathrate to pick the capacity mode [7, 6]. The main phases of Pathrate are:

- Phase I: Packet pair probing. Pathrate generates a large number (1000) of packet-pairs. The goal here is to discover all local modes in the distribution of the packet-pair bandwidth B. One of the Phase I modes is expected to be the capacity of the path. Packets sent by Pathrate in Phase I are of variable size, in order to make the non-capacity local modes weaker and wider.
- Phase II: Packet train probing. During this phase, Pathrate generates 500 trains of N packets and measures the dispersion of the packet trains, which is defined as the time elapsed between the arrival of the first and the last packet of the train at the receiver. Gradually, the resulting bandwidth distribution $B(N)$ of the packet train dispersion becomes unimodal, centered at the so called *Asymptotic Dispersion Rate* R. In the first release of Pathrate [7], The capacity mode is selected as the first phase I mode larger than R, and in the second release [6], the capacity mode is selected as the strongest and narrowest phase I mode among those larger than R.

We next present PPrate, a passive capacity estimation tool whose core algorithms are similar to the ones of Pathrate.

4 PPrate

PPrate takes as input a set of packet inter-arrival times extracted from a TCP connection, and automatically estimates the capacity of the narrow link. PPrate can work with measurements collected at any point along the path from the sender to the receiver. In the remaining of this section, we first present the receiver side case, where one can observe arrivals of the packet pairs sent by the sender. We also present the more intricate sender side case, where one can only indirectly observe the packet pairs arriving at the receiver using the ack stream at the sender side. We next discuss the more general case where the measurement point is located anywhere between the sender and the receiver. In the last part of this section, we discuss the influence of the TCP protocol and the application on top of TCP on the quality of the estimation.

4.1 Receiver side algorithm

Assume a packet trace captured at the receiver side. The packet inter-arrival times can be seen as a time series of packet pair dispersions (similar to the case of active probing), which we can use to form the bandwidth distribution of the path, which is multi-modal in general. We present in Figure 1 an example of such

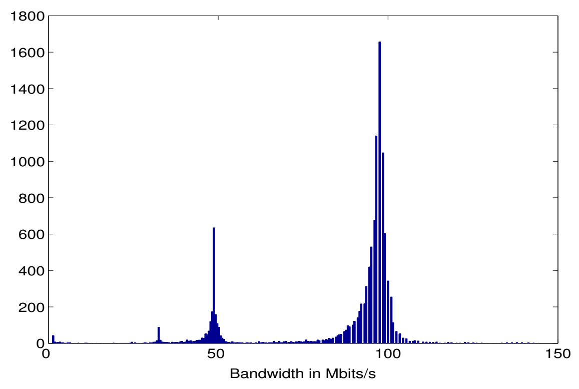


Figure 1: Example of multi-modal Bandwidth distribution

an histogram. Local modes in the distribution of the packet pair dispersion are candidate values for the capacity of the path.

The challenge, here, is to select the local mode corresponding to the narrow link capacity. Similarly to Pathrate, PPrate relies on a lower bound of the capacity to guide its choice of the capacity mode. Pathrate uses the ADR as a lower bound. With PPrate, we sum consecutive inter-arrival times, taken $N-1$ by $N-1$, to form the dispersion histogram $B(N)$ of trains of size N. We expect that, as in the active probing case [6], increasing N will make $B(N)$ unimodal. By analogy with active probing (see Section 3.2), we refer in the remainder of this paper to the center of this *unique* mode as the ADR. Note that while Pathrate can obtain an accurate estimate of the ADR by varying the train size N and by insuring that packets are sent back to back, PPrate is affected by the the TCP sender and the loss process on the path, when sending consecutive packets. As a consequence, we can only obtain a lower bound of the ADR. Experiments however reveal that the N values required to obtain unimodal distributions are in general less than 10 (see Section 6.3).

Algorithm 1 shows the pseudo-code of PPrate. Initially, *PPrate*, constructs the distribution of the packet pair dispersion B, and scans it for local modes. For the detection of local modes, the *bandwidth resolution*, or *bin width* is an important parameter. In general, a bad choice of bin width may conduct to erroneous results [3]. In this work, we set the bin width to 5% of the interquartile range [3] of the capacity measurements. Thus, a wider distribution of measurements leads to a larger bin width, in accordance with standard statistical techniques for density estimation [23]. If B is unimodal, PPrate decides that this mode represents the narrow link capacity of the path. If there are multiple modes, PPrate starts a second phase in order to estimate the ADR. To do so, PPrate groups the inter-arrival time samples $N-1$ by $N-1$, constructs the packet trains dispersion $B(N)$ and scans it for

[3]Let p_{75} (resp. p_{25}) be the 75^{th} (resp. 25^{th}) percentile of a distribution, then the interquartile IQR is defined as $IQR = p_{75} - p_{25}$.

modes. PPrate starts with $N = 3$, and repeats the procedure until $B(N)$ becomes unimodal. PPrate chooses that mode as the Asymptotic Dispersion Rate R. For the few cases where $B(N)$ does not become unimodal, we choose, similarly to Pathrate, the strongest mode in the packet pair histogram as a capacity estimate. The next step is to select in the packet pair dispersion histogram, the mode that corresponds to the capacity. We implemented and evaluated the two methods proposed in [7] and [6]. In [7], the authors propose to select the first peak larger than R. In [6], they propose to select as the capacity mode, the strongest and narrowest mode of B among those larger than R. Our evaluation of the two methods led us to the conclusions is that the latter method is more robust than the former (see [10] for details). PPrate implements the second version of the algorithm.

```
Compute connection inter-arrivals from packet trace file
Compute packet pair dispersion $B$
Find the Modes
if there is only one mode, at value M then
    Output narrow link capacity $C = MSS \times 8/M$
    Exit
else
    $N$=3
    while Number of modes $> 1$ do
        Group inter-arrivals $N$ by $N$
        Compute packet train dispersion $B(N)$
        Find Modes
        $N = N + 1$
    end
    Output single mode $M$
    Compute $ADR = MSS \times 8/M$
    Find the mode $M_c$ larger than ADR among modes of $B$
    Output narrow link capacity $C = MSS \times 8/M_c$
end
```

Algorithm 1: Pseudo-code for PPrate

4.2 ACK based algorithm

If the traffic was not captured at the receiver side, it is impossible to know the dispersion of the packet pair arriving at the receiver. However, we can use the dispersion of the pure *acks* as an approximation of the packet pair dispersion at the receiver. Indeed, pure *acks* are 40-bytes packets, having no payload data, and generally acknowledge two data packets due to the delayed acknowledgment strategy. To "cancel" the effect of delayed *acks* we divide by two the inter-arrival times between successive *acks* that acknowledge two MSS worth of bytes. Once the multi-modal distribution of the path is constructed, the capacity mode is selected as described above. Note that working on the *ack* stream captured at the sender side constitutes a worst-case for the estimation of the path capacity. More generally, the accuracy of the estimation based on the *ack* stream increases when the measurement point is closer to the receiver.

4.3 Upstream capacity

When a packet trace is captured in the middle of the network, PPrate can also be applied to data packets to estimate the **upstream capacity**. Indeed, consider a packet trace T collected at network link L_T, and suppose that f is a TCP flow in T, and S_f is the flow source (Figure 2). The **upstream capacity** of flow f is the minimum link capacity between S_f and L_T.

The work of Jiang et al [13] was the first attempt to passively estimate the upstream capacity based on packet dispersion: The authors employ an histogram-based technique of capacity samples given by the packet pair dispersion to estimate the upstream capacity. To do so, they first drop the 50% smallest capacity samples in order to keep only the packet pairs sent back-to-back due to the delayed ack mechanism. Then, they estimate the upstream capacity as the center of the strongest mode in the capacity histogram. However, recent work on active probing and capacity estimation [6, 7] has shown that the true capacity is a local mode of the distribution, often different from its global mode. We will quantify this difference on example scenarios in Section 7.

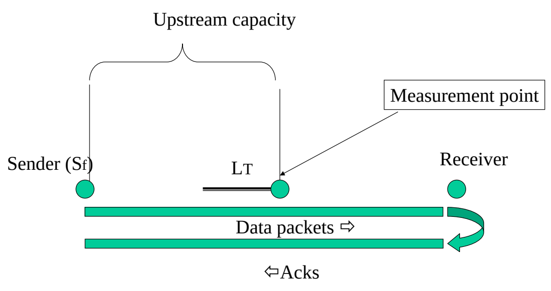


Figure 2: The upstream capacity is measured between the sender and the measurement point

4.4 Impact of TCP and the application on top of TCP

PPrate uses as input packet or *ack* inter-arrival times from TCP connections over which PPrate has no control. Hence, the accuracy of the capacity estimate might be biased by the TCP protocol and also by the application running on top of

TCP. We list in this section the factors that might lead to errors and discuss how to handle them.

Let us first consider the TCP protocol itself. The receiver advertised window will limit the maximum observable train size. Losses also impact the maximum observable train size as it reduces the congestion window. Both a small advertised window and a decrease of the congestion window impact PPrate, as if PPrate cannot observe long enough packet trains, it might be difficult to correctly estimate the Asymptotic Dispersion Rate R. However, in practice, it turns out that the packet train size needed to obtain R is relatively small as compared to commonly observed advertised window sizes in the Internet (see Section 6).

The application on top of TCP can also impact PPrate in a number of ways. First, the application can send data in both directions of a TCP connection. This is, for instance, the case with BitTorrent [5], where the acknowledgments are often piggybacked. In this case, PPrate can be applied to the data streams of the two directions to estimate the upload capacity of both BitTorrent clients. Indeed, the upload capacities of the clients should constitute the narrow link with high probability if those are ADSL clients.

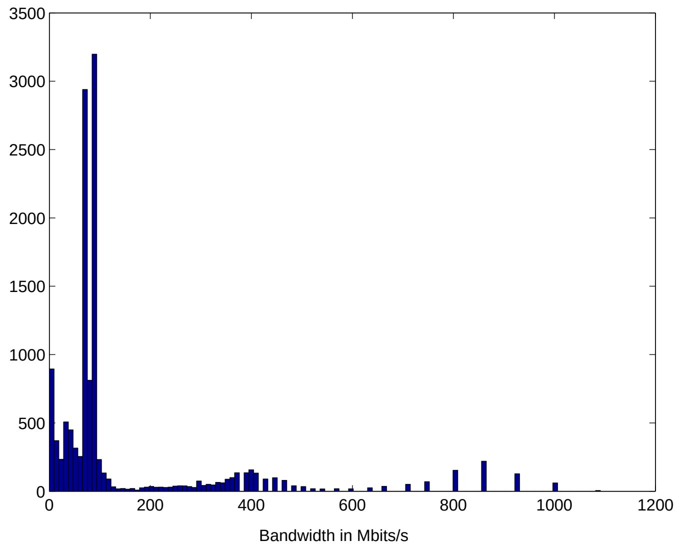


Figure 3: Raw Bandwidth Histogram

Second, the application can generate idle periods when no data is to be exchanged. This is for instance the case with persistent HTTP connections, when the client is reading a page between two requests to the same server. Long idle periods lead to large packet inter-arrival time samples. This can affect both, the histogram of packet pair dispersions and the histograms used to infer R, leading to an underestimation of the narrow link capacity. As a counter-measure, we filter the largest inter-arrival time values prior to any computation of histogram. To do so, we remove all samples that are larger than $min(p_{75} + IQR, p_{95})$. The latter formula

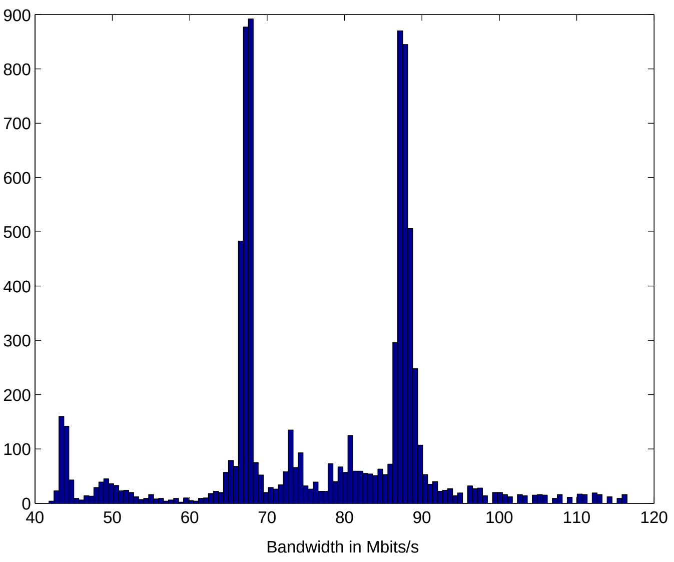


Figure 4: Preprocessed Bandwidth Histogram

means that we remove either the samples that are far away from the 75th percentile of the distribution if they represent more than 5% of the samples, or 5% of the samples. In addition, we filter the values in the distribution smaller than $p_{25} - IQR$, i.e. the values that are far from the core of the distribution. The idea behind this second filtering is not to fight against some application effects but against some measurement errors [20] that lead to extremely small inter-arrival times. To illustrate the impact of those filtering mechanisms, consider the bandwidth histogram in Figure 3 obtained from some raw data. We observe in this histogram abnormally high bandwidth values, up to 1 Gbits/s. Figure 4 depicts the same histogram once raw data has been filtered.

Third, the application can enforce some rate limitations. This is the case of many p2p applications like E-donkey and BitTorrent. Different techniques exist to limit the rate of an application [25]. For instance, the application can deliver small amounts of data to the TCP layer and set the PUSH flag to force TCP to send packets smaller than the MSS. This can be observed for some E-donkey clients [25]. Another option for the application is to send bursts of bytes (that will generate bursts of MSS size packets at the TCP layer) separated by idle periods. This can be observed with some BitTorrent clients [25]. More generally, there is a need for a method that automatically detects the periods in a TCP connection where the application is limiting the transfer rate of the connection. Such a method has been proposed in [25]. In the context of this paper, we will focus on some applications that do not apply rate limitations, such as FTP, SCP and HTTP transfers.

5 Planetlab Experiments

In this section, we compare the existing passive capacity estimation tools, namely PPrate, Nettimer and MultiQ to one active tool, Pathrate. We used Planetlab, which enables us to be connected to the two ends of a path; a mandatory feature for an active tool like Pathrate. Our focus is on receiver side experiments, i.e., we capture the data stream at the receiver side. We were not able to collect any useful (ack) traces at the sender side as, on average, there was one ack for 3 to 4 data packets while the delayed acknowledgment strategy would recommend 2.

5.1 Receiver side experiments

We selected 33 paths between randomly chosen PlanetLab nodes. On each path, we first perform one *scp* transfer (of a 20 Mbytes file), and simultaneously dump the data packet stream at the receiver side. Note that it is not possible to collect traces at both end points of a path on PlanetLab. Immediately after the *scp* transfer, we run Pathrate[4] on the same path. The overall procedure lasts a few hours for each path, as it takes 15 to 30 minutes for Pathrate to return an estimate and 1 to 5 minutes for each *scp* transfer.

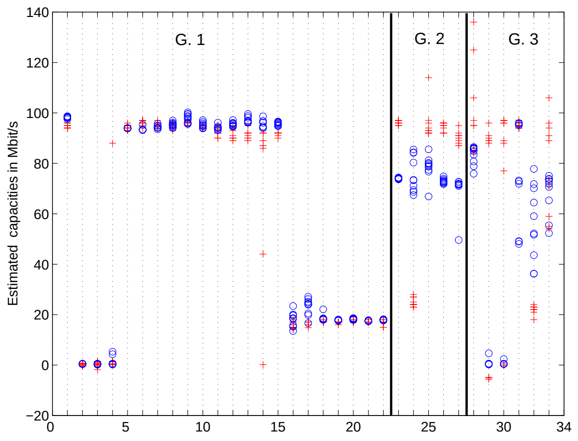
Figures 5(a) to 5(c) summarize the results for the 33 paths that we considered. A given index on the x axis corresponds to a single path. For each figure we use crosses for Pathrate and circles for the considered passive tool.

To ease interpretation of the results, we have formed 3 groups of paths labeled as Group 1, 2 and 3. The grouping stems from the comparison between Pathrate and PPrate in Figure 5(a) and we use the same grouping for Figures 5(b) and 5(c). Group 1 contains the paths for which Pathrate and PPrate return consistent estimates (i.e., each tool returns approximately the same values for the 10 experiments) and, in addition, the two tools agree with each other. There are 22 paths, that is two thirds of the paths, in Group 1. Group 2 consists of paths for which each tool returns consistent estimates but the two tools do not agree with one another. A relatively small number of paths (4) fall in this group. Group 3 consists of seven paths for which one or both tools return inconsistent estimates.

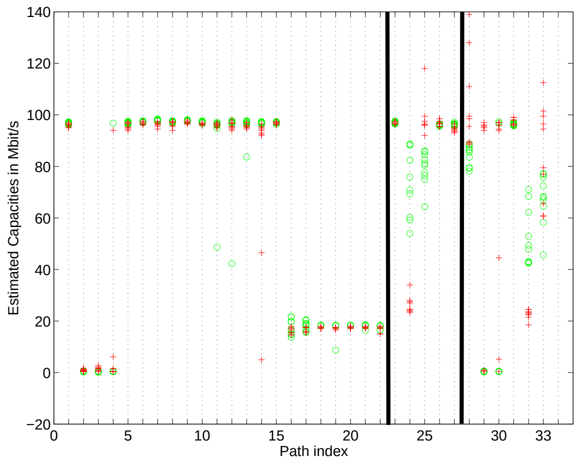
Overall, we observe from Figures 5(b) and 5(c), that for each group, PPrate, Nettimer and MultiQ behave similarly to Pathrate. Note that for the case of MultiQ the scale on the y-axis goes up to 800 Mbits/s, as we wanted to emphasize the tendency of MultiQ to return, from time to time, large values that should be, with high probability, overestimates. Note however that if we rescale Figure 5(c) to a maximum of 140 Mbits/s like the two others, all three figures become qualitatively similar. All tools can exhibit a high variance in their estimates for the paths in Group 3. It is difficult to comment on the paths in this group as many factors might explain why the tools do not work properly. A possible reason for these high variances is the use of slices and rate limitations on Planetlab nodes[5]. We note that

[4]Pathrate returning two estimates for each run [6], we use their average.

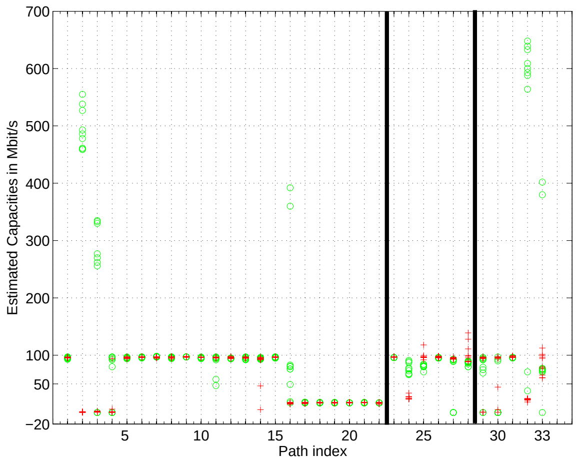
[5]http://www.planet-lab.org/doc/BandwidthLimits.php



(a) Pathrate(+) vs PPrate(o)



(b) Pathrate(+) vs Nettimer(o)



(c) Pathrate(+) vs MultiQ(o)

Figure 5: PlanetLab Experiments

previous studies of Pathrate over Planetlab nodes have also observed a number of problems [17].

The main conclusion that we draw from those experiments is that for a significant number of cases, the 3 passive tools fully or partially agree with our reference tool, namely Pathrate. We note however that MultiQ tends to generate larger estimates than the other tools, from time to time. While we do not know the exact value of the path capacity, it seems extremely unlikely to be significantly larger than 100 Mbits/s as suggested by MultiQ. This observation strongly suggests that MultiQ can sometimes overestimate the path capacity.

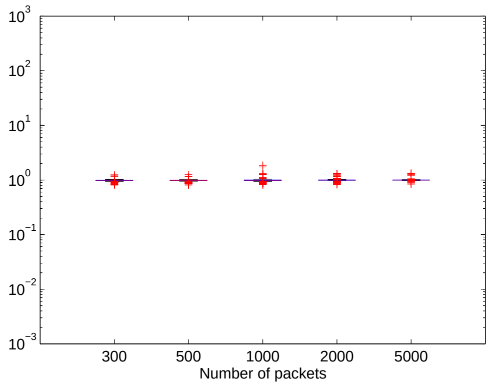
5.2 Consistency

The number of samples required to perform a correct estimation of the capacity of a path is a crucial point for passive tools as they can be applied on traces for which it is not possible to control the duration of a transfer. As we do not know the exact value of the paths capacities on Planetlab, we will address the above issue in an indirect manner by focusing on the consistency of each tool. We proceed as follows. We consider the 330 *scp* transfers that we did (10 experiments per path with a total of 33 paths) and applied, for each transfer, PPrate, Nettimer and MultiQ on the first 300, 500, 1000, 2000 or 5000 first packets of the transfer (the whole transfers contain about 12000 packets). We next compute for each case, the ratio between the estimate obtained with the first x packets and the estimate obtained using all the packets. The consistency of a tool will be measured as the fraction of ratios close to 1.

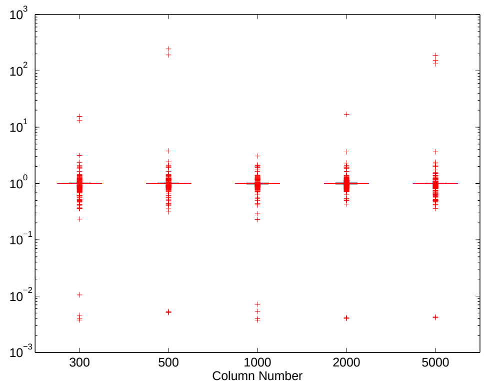
We plot in Figures 6(a) to 6(c) the boxplots[6] of the estimations for the different x values and for all the transfers on all the paths, returned by PPrate, Nettimer and MultiQ respectively.

We first observe from Figures 6(a) to 6(c) that the boxplots for each tool have their upper and lower edges close to 1, meaning that most of the time, the estimate obtained by a given tool with $x < 120000$ samples is consistent with the one obtained with 12000 samples. In the boxplot representation, extreme values, i.e., values that are far from the core of the distribution, are marked with crossed. With respect to those extreme values, we observe that Nettimer and even more MultiQ are more prone to generate extreme values (i.e., large under or over estimations) than PPrate. Indeed, Nettimer and MultiQ exhibit large outliers even when using 5000 samples, as compared to PPrate whose maximum value is 1.8 and minimum value is 0.8. Identifying the reason behind those observations is not straightforward. The three tools rely on the histogram of packet pairs dispersion to infer the

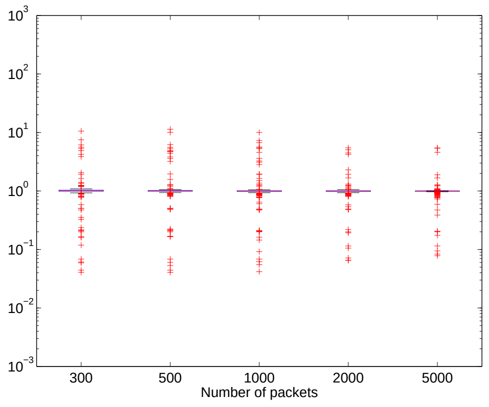
[6]A boxplot is a graphical representation of a distribution where the upper line of the box is the 75^{th} percentile p_{75} of the distribution, the lower line is the 25^{th} percentile p_{25} and the central line is the median of the distribution. In addition, two lines are added on each side of the box at $+/-1.5IQR$. 50% of the mass of the distribution in the box while, intuitively, one expects most of the samples to lie between the two outer lines. Values outside those boundaries are marked with a cross as they are extreme values of the distribution.



(a) PPrate



(b) Nettimer



(c) MultiQ

Figure 6: Boxplots of ratio between the estimates obtained with x packets and the ones obtained with the full FTP transfer

capacity of the path. However, they differ in the filtering technique they use to uncover the capacity mode. Nettimer and MultiQ both rely on kernel density-based technique to filter the dispersion histogram. PPrate uses the ADR to guide the search for the capacity mode, after a preliminary phase where outliers due either to the application on top or to measurement errors (context switching in the OS) are removed. It is difficult to assess a priori the relative merits of both approaches. We conjecture that the larger number of extreme values generated by Nettimer and MultiQ might stem from the kernel density estimation technique used in Nettimer and MultiQ that, in general, require a quite large number of samples to rip the full benefit of the method.

6 ADSL Traces Analysis

In this section, we further compare PPrate, Nettimer and MultiQ on traces from an ADSL access network[7] made available by the University of Twente. We focus our comparison on two specific traces. The first trace consists of FTP connections while the second trace consists of HTTP connections. We use only connections with more than 300 packets, which represent 20% of the observed *ack* streams for the FTP trace and 8% of the *ack* streams for the HTTP trace. Distributions of the connection sizes in packets, see Figure 7, indicate that the majority of the connections are large (more than 1000 packets), the FTP connections being globally larger than the HTTP connections.

All traffic was collected by a probe close to the ADSL clients. The only information we have on the clients is that their access link capacities range from 256 kbits/s to 8 Mbits/s. By working on the ack stream collected by the probe, passive capacity estimation tools should be able to estimate the downlink capacity of the clients, as the latter should constitute the bottleneck of the path (Section 6.2). Conversely, working on the data streams allows us to infer the upstream capacity, i.e. the capacity of the portion of the path between the sender and the measurement point, see Figure 2.

6.1 TCP data streams

Due to the measurement setting in the ADSL traces of the University of Twente, working on the TCP data streams should uncover the upstream capacity (see Section 4.2). Given the low utilization often observed in core networks, the upstream capacity should often equal the capacity of the server in client server applications like FTP and HTTP.

We applied PPrate, Nettimer and MultiQ on the data stream of FTP and HTTP connections. Figures 8 and 9 represent the cumulative distribution functions of the estimates of the three tools for the HTTP and FTP traces.

[7]M2C Measurement Data Repository: http://m2c-a.cs.utwente.nl/

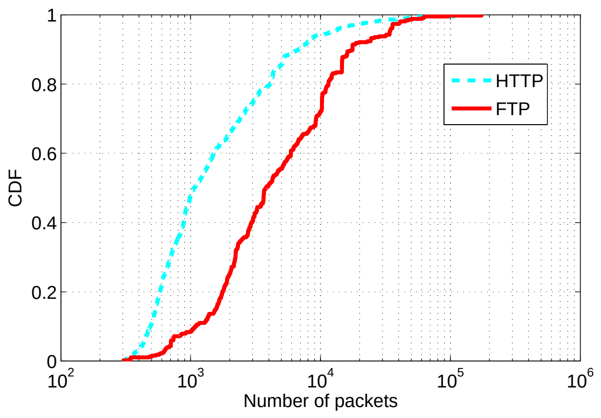


Figure 7: CDF of the number of packets of HTTP and FTP connections

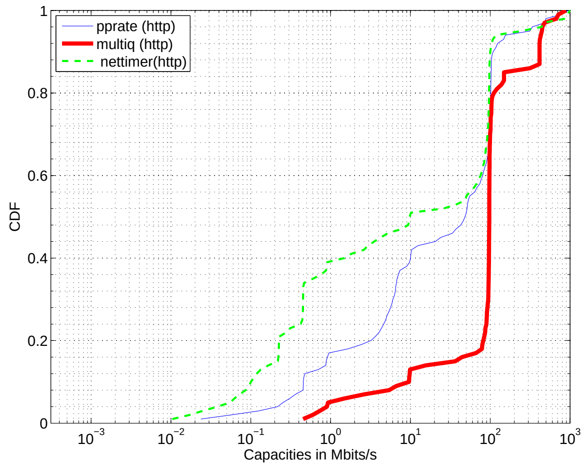


Figure 8: Path capacity estimates for the data streams of the HTTP TRACE

6.1.1 HTTP trace

Figure 8 shows the distribution of the estimates for HTTP connections. We observe peaks at 10 and 100 Mbits/s for the three tools, in accordance with the intuition that HTTP servers have an Ethernet or FastEthernet network interface.

In contrast to FTP connections (see Section 6.1.2), PPrate and MultiQ distributions of capacities for HTTP traffic are very different. MultiQ returns an estimate of 100 Mbit/s for 75% of the connections, while PPrate return 100 Mbit/s for only 25% of the connections. To understand this phenomenon, we plot in Figure 10(a) a scatter plot of the results of PPrate and MultiQ. To ease interpretation, we highlight four regions in Figure 10(a) : region 1 where MultiQ return values larger than 200 Mbit/s, which represent 14.3% of the connections; region 2 where both tool return

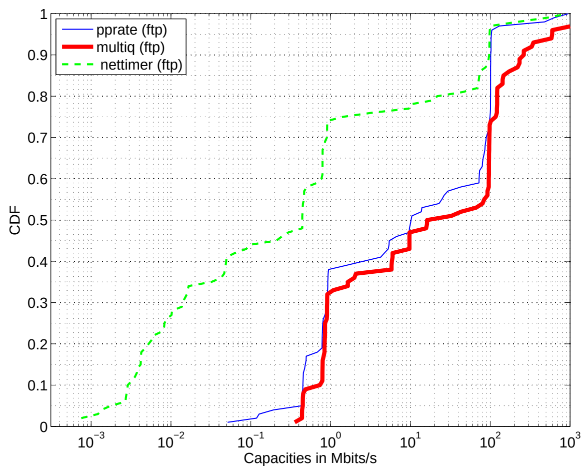


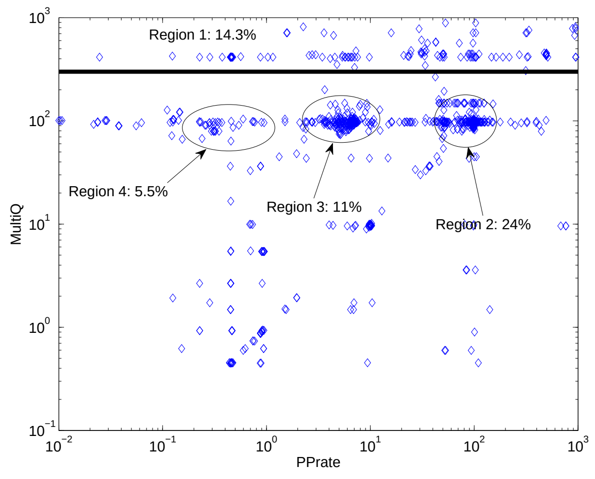
Figure 9: Path capacity estimates for the data streams of the FTP TRACE

values around 100 Mbit/s; region 3 where PParte returns values around 10 Mbit/s and MultiQ returns values around 100 Mbit/s and region 4 where PPrate returns values around 1 Mbit/s and MultiQ returns values around 100 Mbit/s. Region 1 corresponds, with a high probability, to cases where MultiQ overestimates path capacities. Region 2, which corresponds to about 24% of the cases, corresponds to cases where both PPrate and MultiQ agree with each other. Regions 3 and 4, which encompass around 16.5% of the samples, correspond to cases where MultiQ return capacity values that are one or two orders of magnitudes larger than the ones of PPrate. To understand the difference between PPrate and MultiQ, we exhibited capacity histograms for connections in regions 1,3 and 4.

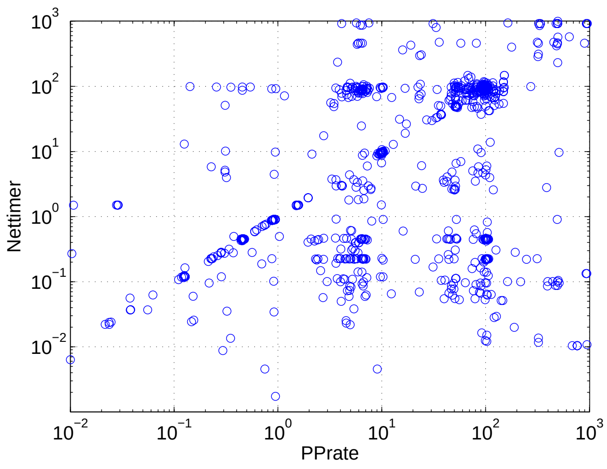
Figure 11(a) shows an example of capacity histogram in the first region. In this example, PPrate returns a capacity of 50 Mbit/s while MultiQ returns an estimate of 400Mbit/s. Although we do not know the exact value of the capacity, 50Mbit/s is a reasonable value for an HTTP Server. However, it is clear that MultiQ fails to estimate the capacity in this case as there no capacity samples larger than 150 Mbit/s in this example (note that we presented the raw capacity histogram and not the filtered histograms where large samples could have been filtered out).

Figure 11(b) shows an example of capacity histogram in region 3. In this example, MultiQ elects the second peak as the capacity peak, while PPrate chooses the first peak. PPrate excludes the peak around 100 Mbit/s as its size is an order of magnitude smaller than the dominant peak. It is difficult to know which tool is right. Note however that the first peak is so narrow that it is not unlikely that the second peak be due to compression of some of the packets at a 100 Mbits/s link along the path.

Figure 11(c) shows an example of capacity histograms in region 4. This case is similar to the one depicted in Figure 11(a): there is no mass at the value returned

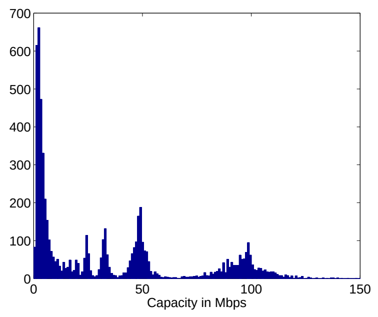


(a) PPrate vs MultiQ

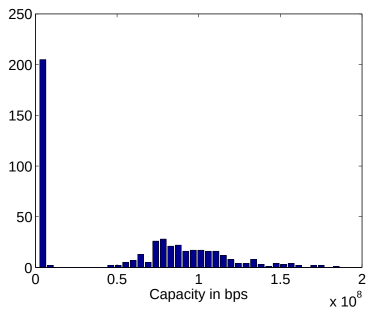


(b) PPrate vs Nettimer

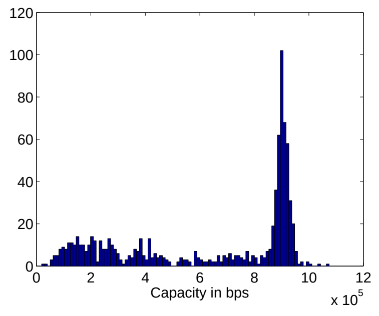
Figure 10: Scatter plots for HTTP connections



(a) PPrate ∼50 Mbits/s, MultiQ ∼400 Mbits/s



(b) PPrate ∼10 Mbits/s, MultiQ ∼100 Mbits/s



(c) PPrate ∼1 Mbits/s, MultiQ ∼100 Mbits/s

Figure 11: Examples of Capacity Histograms

by MultiQ. It is difficult to diagnose the cause behind this strange behavior of MultiQ. Note that it can't be any obvious programming error as one fails to identify a consistent behavior, e.g, a consistent overestimation, of MultiQ. Indeed, there are a number of cases, e.g. region 2 of Figure 10(a) for the HTTP traffic, where MultiQ agrees with PPrate and both tools return reasonable results.

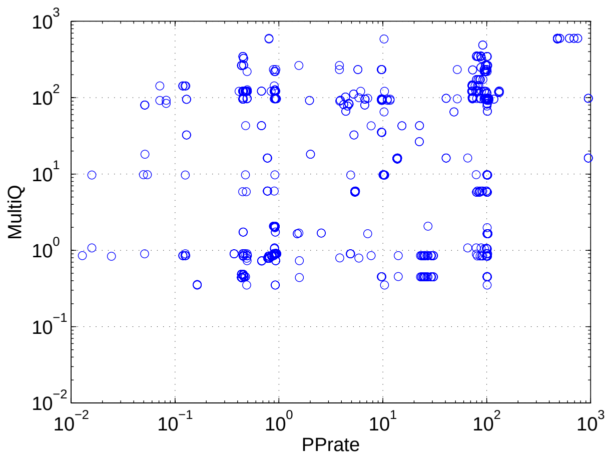
Let us now focus on Nettimer vs PPrate for the case of the HTTP trace. We first note the tendency of Nettimer to under estimate the capacity. Indeed, about 50% of the estimates are less than 1Mbits/s which far from typical values for HTTP servers. The scatterplot of Figure 10(b) indicates that the major difference between PPrate and Nettimer is that in a number of cases, PPrate estimates capacity values around 10 or 100 Mbit/s while Nettimer returns values smaller than 1 Mbit/s. Inspection of specific histograms (as for the case of MultiQ) reveals that Nettimer often elects a minor peak as the peak of the capacity. However, in contrast to MultiQ, Nettimer elects an existing peak.

6.1.2 FTP trace

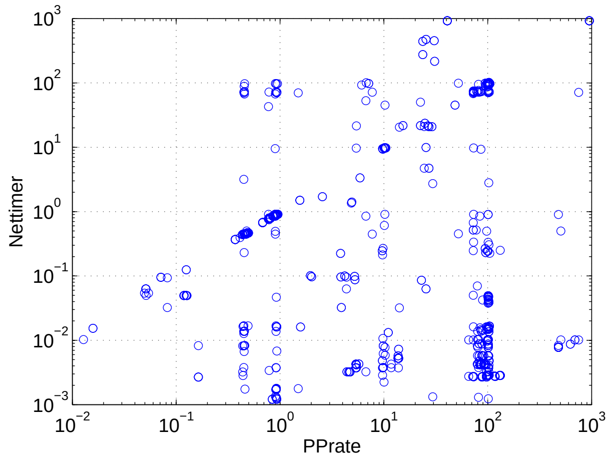
Figure 9 shows the distribution of the estimates returned by PPrate, MultiQ and Nettimer for FTP servers. Note that we expect to have better results for FTP connections as the application should not limit the rate of the connections. PPrate and MultiQ seem to agree with each other, even though we note that 20% of the values returned by MultiQ are larger than 200 Mbit/s. Inspection of specific capacity histograms suggests that MultiQ overestimates path capacities in those cases. Scatterplot (Figure 12(a)) of the results returned by MultiQ and PPrate shows that the cumulative distribution functions of Figure 9 was misleading. Indeed, in a number of cases, PPrate returns values close to 100 Mbit/s while MultiQ returns values close to 1 Mbit/s and vice versa. The situation here turns out to be often similar to the one in Figure 11(b), and it is difficult to determine which tool is right.

The case of Nettimer is significantly different. Indeed, the distribution returned by Nettimer on FTP traffic, see Figure 9, is very different from the ones of both PPrate and MultiQ. Nettimer returns very small values as compared to the other tools. About 80% of its estimates are less than 1Mbits/s, and more than 30% are less than 10kbits/s. Even if we assume that the users access FTP servers located on machines of other ADSL users (in other ADSL networks), those capacities remain relatively small as compared to standard values of uplink ADSL link, even in 2004. Inspection of the scatterplot of Nettimer vs. PPrate (Figure 12(b)) where we observe that when Nettimer returns extremely small values, PPrate returns values close to 100 Mbits/s. Visual inspection of some histograms in this category reveals that Nettimer elects too small peaks as capacity peaks. Indeed, the mass around 100 Mbit/s in those cases forms a clear, i.e. both strong and narrow, peak and strongly suggests that Nettimer understimates path capacities.

Overall our conclusions for the comparison of the three tools when applied on the data streams of FTP and HTTP trace are:



(a) PPrate vs MultiQ



(b) PPrate vs Nettimer

Figure 12: Scatter plots for FTP connections

- Nettimer seems to often underestimate path capacities.
- In contrast, MultiQ tends to sometimes overestimates path capacities. This is in line with what we observed in Section 5.
- There is a number of cases where determining which tool is actually right is difficult, if not impossible. This further justifies our use of Planetlab to run experiment in a environment where we can compare the passive tools to an active tool. However, Planetlab by itself is not enough and experiments in a challenging environment like an ADSL platform, allow to underscore specific behaviors of the tools.

6.2 TCP ack streams

Our focus in this section is on the capacity estimated returned by Nettimer, MultiQ and PPrate when applied on the TCP ack streams of M2C ADSL dataset.

We first observe that the vast majority of FTP and HTTP servers observed in our traces should not to be hosted by the ADSL clients we observe. As a consequence, *ack* streams of the FTP and HTTP traces should reveal the bottleneck of the path from servers in the Internet to ADSL clients, which should be with high probability the downlink of the clients[8].

Figure 13 represents the cumulative distribution functions of capacity estimates of the 3 tools for the two types of traffic. The two vertical bars mark the lower bound (256 kbits/s) and the upper bound (8 Mbits/s) on the clients download capacities.

We observe from Figure 13 that Nettimer is way off as compared to the two other tools, as over 75% of the values returned by Nettimer for the FTP and HTTP traces are below the lower bound of 256 kbits/s. Note however that Nettimer was not optimized to work on ack streams, and all the results in [16] were obtained for data streams.

There is quite a good agreement between the distribution of capacities for the FTP connections and the HTTP connections for PPrate. Results are less satisfactory for MultiQ. Note that we could not expect a perfect matching between those two curves (for a given tool) as anonymization of the traces prevents a precise identification of the clients and thus it is possible that for some clients of the ADSL network, we do have only one FTP or one HTTP connection (or many samples for one type of application and none for the other type)

We also observe that approximately 70% (resp. 85%) of the estimated capacities fall in the interval [256kbits/s, 8Mbits/s] with PPrate (resp. with MultiQ). For both tools, we do observe peaks around values close to 500 kbits/s, 1 Mbits/s and

[8]We assume here that the uplink of the clients should not slow down ack streams as the ratio of size between acks and packets is in general larger than the ratio of uplink to downlink capacity. It is of course possible that the uplink constitutes the bottleneck if, e.g., a p2p application is saturating the link. We however note that p2p applications often use rate limiters to avoid those situations. A recent study on ADSL users has observed that utilization of the uplinks are in general low [24].

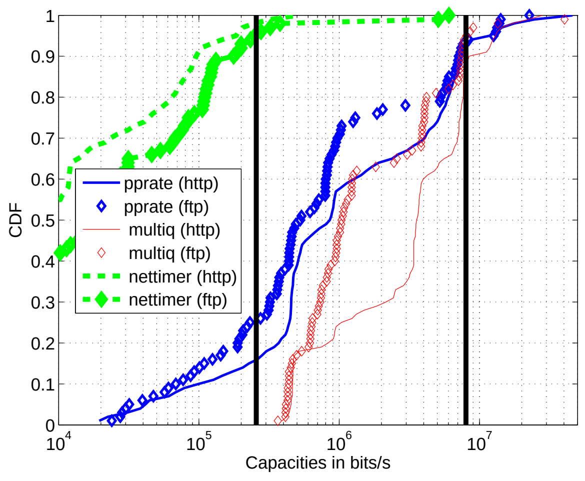


Figure 13: Path capacity estimates for the *ack* streams of the FTP and HTTP traces

8 Mbits/s, which are typical capacities of ADSL clients. The reason why we do not observe mass only at those characteristic values might be due to the fact that the actual ADSL capacity decreases with increasing distance of the phone line between the customer premise and the ADSL concentrator. It is however puzzling that the smallest values returned by MultiQ are around 400 kbit/s, and not closer to 256 kbits/s. As in the previous section where we focused on the upstream capacity, we observe a better agreement between PPrate and MultiQ for the FTP than for HTTP traffic. We applied the same technique as we did in the previous section to try to understand why MultiQ returns larger values than PPrate. Observation of some specific histograms revealed again that MultiQ sometimes sees mass where it should not in the capacity histogram.

6.3 Asymptotic Dispersion Rate (ADR)

PPrate needs to estimate the ADR of a path to correctly infer its capacity. The ADR is computed for the smallest value of N such that the dispersion histogram of trains of size N is unimodal. Intuitively, if N is small as compared to the advertised window of the considered connection, we can expect the estimation of the ADR to be made correctly. We present in Figure 14 the distribution of N for our FTP and HTTP traffic traces. Overall, we observe that for both types of traffic, 80% of the ADRs have been estimated with trains of less than 5 samples and over 93% with trains of less than 10 samples. When working on TCP data streams, one sample corresponds to one data packet, while for TCP ack streams it corresponds to 2 data packets. This means that in the worst case, i.e. the ack stream case, one needs the equivalent of 20 data packets to estimate the ADR. Given that a typical advertised window is 65 kilobytes, i.e. 43 data packets of 1500 bytes, we observe that estimating the ADR should not be an issue in general.

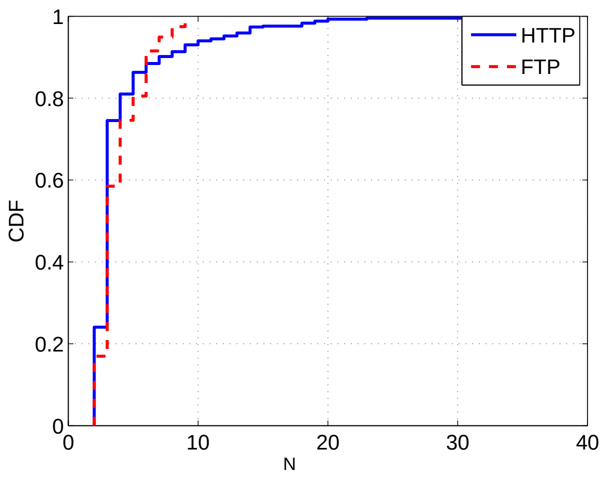


Figure 14: Number of samples N used to estimate the ADR

7 How far is the path capacity mode from the strongest mode?

It is well known [6, 19] that the mode corresponding to the true capacity of a path can differ from the strongest mode in the capacity histogram, especially if the path is significantly loaded. Figure 15 represents the capacity histogram, obtained by simulation, of a TCP connection that spans over a loaded path. The strongest mode is around 25 Mbits/s while the true capacity corresponds to the small mode at 40 Mbits/s.

In order to quantify how far the capacity mode is to the strongest mode in operational scenarios, we used two different datasets, both made available by the M2C project of the university of Twente[9]. The first dataset is the ADSL trace already used in Section 6. The second dataset consists of measurements collected on a ($3 \times$ 100Mbit/s) Ethernet trunk, with 2000 students connected using 100 Mbits/s Ethernet access link. For the ADSL dataset, the average estimated load is around 15% while for the Ethernet access network, the estimated average load is about 60% (as reported on the web site of the M2C project).

Let us first consider the ADSL trace. We focus here on the HTTP traffic only, as the results for the FTP traffic are qualitatively similar. For each connection and each direction of the traffic, we estimate the capacity on the corresponding part of the path (see Figure 2). We also extract each time the strongest mode of the histogram. Figure 16 represents the cumulative distribution functions of the ratio between a PPrate estimate and the center of the strongest mode for each direction of traffic. A ratio close to one means that the two techniques lead to a similar estimate. We observe from Figure 16 that for both directions, the naive method

[9]http://m2c-a.cs.utwente.nl/repository/, locations 1 and 4.

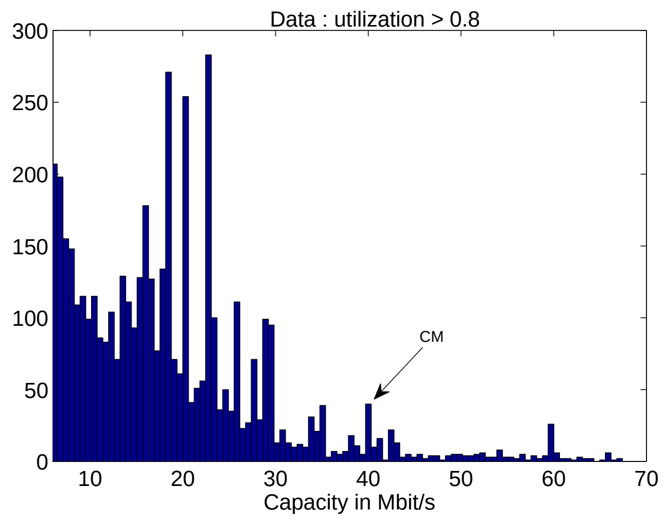


Figure 15: A sample of Capacity histogram obtained from a heavily loaded path

(that consists in picking the strongest mode) provides satisfactory results for about 80% of the cases when working on the TCP ack streams and around 60% when working on the TCP data stream. We further observe from Figure 16 that the naive method can only underestimate the capacity of the path (we assume here that the value returned by PPrate is accurate). The reason behind this observation lies in the search technique used by both Pathrate and PPrate, which is to pick the strongest mode **greater** than the ADR of the path (see Section 3.2). As a consequence, the estimate returned by PPrate can only be larger or equal to the strongest mode of the capacity histogram. We hypothesize that the reason why the naive method works apparently better for the TCP ack stream as compared to the TCP data streams is that it is often the downlink capacity of the client that is estimated in the former case. Since the measurement probe is close to the client, the strongest mode should relate to the downlink capacity as long as the uplink is not too loaded. On the other hand, the TCP data streams are analyzed once data packets have traversed many hops from the HTTP server to the client, which gives more chance to compression and expansion to alter the shape of the capacity histogram.

For the Ethernet trace, we selected the TCP ack streams with more than 300 samples, which represent 2% of the total TCP ack streams observed, but 97.4% of the bytes. Note that as the Ethernet technology offers symmetric access links, working on the TCP ack stream is similar to working on the TCP data stream. For each connection, we collected the capacity estimate returned by PPrate, along with the strongest mode of the capacity histogram. To assess the impact of the initial preprocessing stage of PPrate (see Section 4.4), we further compute the strongest mode for the raw data for each connection.

For each connection, we formed two ratios. The first one is the ratio between

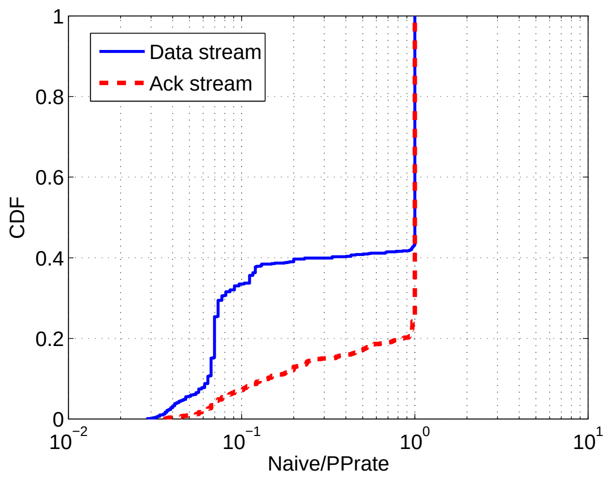


Figure 16: Ratio of PPrate estimates to the strongest mode for HTTP traffic.

the estimation given by the strongest mode in the raw data and PPrate; and the second one between the filtered data and PPrate. Figure 17 shows the cumulative distribution functions for the two ratios. We first observe that filtering has a significant impact on the results as 70% of the capacity estimates given by the strongest mode agree with the estimates of PPrate, while this percentage falls down to 30% without filtering. However, while the naive technique returns correct estimates for almost 70% of the cases, the error made with the remaining 30% cases can be extremely large (a factor of ten or above for almost 20% of the cases). Those results are similar to the ones observed for the case of the TCP data streams of the ADSL trace (see Figure 16). We do not have a complete explanation for this phenomenon, but we note that both types of traces have in common that the estimation is made once the stream has crossed a significant section of the Internet. This again emphasizes the impact of compression and expansion on the shape of the capacity histogram.

8 Compression Analysis

We focus in this section on the compression of packet pairs for the HTTP and FTP connections considered in Section 6. A packet pair is said to be compressed if its dispersion is smaller than what would be obtained if the two packets arrive back-to-back at the narrow link and if their dispersion remains the same afterward. We expect that most of the compression that we observe is on the forward (data) path as the *ack* streams are captured close to the receiver. Note however that if clients generate high volumes of peer-to-peer traffic, it is possible that the *ack* stream of the HTTP and FTP connections have to compete with peer-to-peer data streams for the upstream link capacity of the client. As we have no means to associate

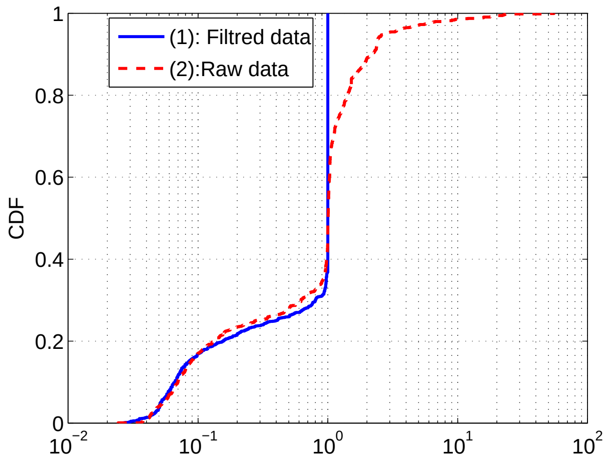


Figure 17: CDF of the ratio of PPrate estimations with: (1) the estimation given by the strongest mode of the filtered data, (2) estimation given by the strongest mode of the raw data

connections with an origin client machine, we can not check if there are such flows competing for the clients uplinks.

Our objective here is to characterize the compression experienced by the connections at the specific point where data is collected. For each connection, we proceed as follows. We pick the modes in the packet-pair histogram used by PPrate, which are on the right (larger capacity modes) of the peak that corresponds to the capacity of the path, according to PPrate. We then form two metrics. The first one is the fraction of the total mass carried by those modes, refered to as the **compressed mass**. The second one is the weighted average capacity corresponding to those modes divided by the capacity of the path. Indeed, each mode i can be associated to a given capacity S_i and carries a certain fraction of the mass m_i. We first calculate the weighted average capacity $\sum S_i \times m_i / \sum m_i$, and next divide the result by the capacity of the path. We refer to the this metric as the **average compression**.

Figures 18 and 19 present the results for both the compressed mass and the average compression. For both of the figures, we provide the results for the FTP and HTTP connections. A first observation is that for the two types of traffic, the results are roughly similar. The actual difference in the tail of the distributions might be due to the fact that we have far less FTP samples than HTTP samples. From Figure 18, we observe that for about 15% of the connections, the compressed mass is 0, meaning that no packet pair experienced compression. For the remaining cases, we observe that a significant fraction of the samples can experience some compression. Figure 19 provides some insight on the average compression. Overall, we observe that for 90% of the FTP connections and 80% of the HTTP connections, the average compression is smaller than 2, while values over 5 are clearly rare.

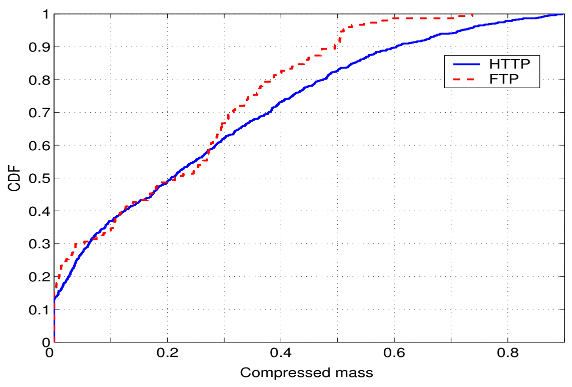


Figure 18: CDF of the *compressed mass* for FTP and HTTP coneections

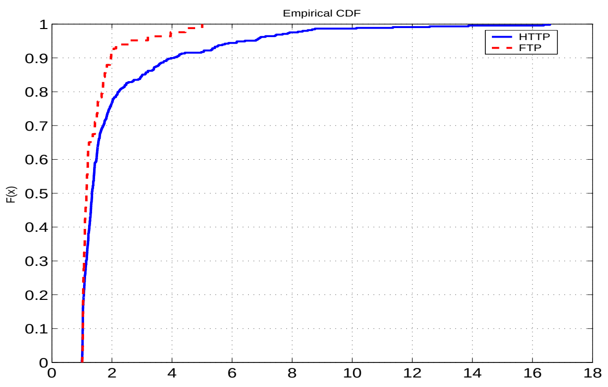


Figure 19: CDF of the *average compression* for FTP and HTTP connections

The main conclusion that we draw from the comparison between Figures 18 and 19 is that while a significant fraction of the samples can be compressed at speeds higher than the path capacity, those speeds are in general close to the path capacity as 80% of these speeds for HTTP and 90% for FTP are smaller than 2 times the narrow link capacity.

9 Conclusion

In this paper, we introduced PPrate, a passive capacity estimation tool. PPrate can extract path capacity from packet traces using either a TCP data or TCP ack stream, collected at any location in the network.

We have compared PPrate to two other passive tools, namely MultiQ and Nettimer. To the best of our knowledge, no other passive tools, with a publicly available implementation, have been proposed so far. We first compared PPrate, Nettimer and MultiQ to one popular active tool, Pathrate using Planetlab. The main conclusion we draw from this experiment is that the three passive tools return results in line with the ones of PPrate, even though MultiQ tends to overestimate paths capacity as compared to Nettimer and PPrate, and both MultiQ and Nettimer appear to be less consistent than PPrate.

We next applied PPrate, Nettimer and MultiQ to publicly available ADSL traces of HTTP and FTP traffic. Results turn out to significantly differ from the ones obtained using Planetlab. Nettimer, which is the oldest tool, appears to be unable to work correctly when processing ack streams and to underestimate path capacities when working on data streams. In contrast, MultiQ is able to work both on ack and data streams but tends to overestimate paths capacity in a number of cases.

We also demonstrated on the same ADSL dataset and an additional LAN dataset that using the strongest mode as an estimate of the path capacity is applicable in 70% of the cases, though over or underestimations are observed in the remaining case. In addition, we underscore the crucial impact of preprocessing the raw capacity histogram to obtain satisfactory results.

As future work, we plan to develop an embedded flavor of PPrate, so as to incorporate PPrate's techniques into Internet applications, e.g., p2p systems, that could benefit from rapid and accurate capacity estimations. The matlab code of PPrate may be downloaded from: `http://www.eurecom.fr/~ennajjar/`.

References

[1] R. Caceres, N. Duffield, and A. Feldmann, "Measurement and Analysis of IP Network Usage and Behavior", *IEEE Communications Magazine*, 2000.

[2] R. Carter and M. Crovella, "Dynamic Server Selection using Bandwidth Probing in Wide-Area Networks", BU-CS-007, Boston University, 1996.

[3] P. Chaudhuri, J. Marron, J. Kim, R. Li, V. Rondonotti, and J. de Una Alvarez, "SiZer: Which features are "really there"?", http://www.stat.unc.edu/faculty/marron/DataAnalysis/SiZer.

[4] L.-J. L.-J. Chen, T. Sun, G. Yang, M. Y. Sanadidi, and M. Gerla, "Monitoring access link capacity using TFRC probe", *Computer Communications*, 29(10):1605–1613, 2006.

[5] B. Cohen, "Incentives Build Robustness in BitTorrent", In *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, 2003.

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

[7] C. Dovrolis, P. Ramanathan, and D. Moore, "What Do Packet Dispersion Techniques Measure?", In *INFOCOM'01*, pp. 905–914, Los Alamitos, CA, April 22–26 2001, IEEE Computer Society.

[8] A. Downey, "Clink: A tool for Esimating Internet Link Characteristics", *http://rocky.wellesley.edu/doweney/clink/*, 1999.

[9] A. Downey, "Using Pathchar to Estimate Link Characteristics", *In Proceeding of SIGCOMM'99*, 1999.

[10] T. En-Najjary and G. Urvoy-Keller, "PPrate: A Passive Capacity Estimation", In *IEEE e2emon*, Vanvouver, CANADA, 2006.

[11] T. Isdal, M. Piatek, A. Krishnamurthy, and T. Anderson, "Leveraging BitTorrent for End Host Measurements", In *PAM 2007*, 2007.

[12] V. Jacobson, "Pathchar", http://www.caida.org/Pathchar/ Source: ftp://ftp.ee.lbl.gov/pa thchar.

[13] H. Jiang and C. Dovrolis, "The effect of flow capacities on the burstiness of aggregated traffic", In *Passive and Active Measurements 2004*, 2004.

[14] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi, "CapProbe: A Simple and Accurate Capacity Estimation Technique", *In Proceeding ACM SIGCOMM*, 2004.

[15] S. Katti, D. Katabi, C. Blake, E. Kohler, and J. Strauss, "MultiQ: automated detection of multiple bottleneck capacities along a path", In *IMC '04*, pp. 245–250, 2004.

[16] K. Lai and M. Baker, "Nettimer: A Tool for Measuring Bottleneck Link Bandwidth", *In Proceeding of USENIX*, 1999.

[17] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring Bandwidth between PlanetLab Nodes", In *Passive and Active Measurements 2005*, March 2005.

[18] B. Mah, "pchar: A tool for measuring Internet Paths Charcteristics", *http://www.employees.org/bmah/Software/pchar/*, 2000.

[19] V. Paxson, *Measurements and analysis of End-to-End Internet Dynamics*, Ph.D. Thesis, University of California, Berkely, April 1997.

[20] V. Paxson, "Strategies for sound internet measurement", In *IMC'04*, 2004.

[21] A. Persson, C. A. C. Marcondes, L.-J. Chen, M. Y. Sanadidi, and M. Gerla, "TCP Probe: A TCP with built-in Path Capacity Estimation", *In Proceeding of he 8th IEEE Global Internet Symposium*, 2005.

[22] S. Saroiu, P. Gummadi, and S. Gribble, "Sprobe: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments", *http://sprobe.cs.washinton.edu*, 2002.

[23] D. Scott, *Multivariate Density Estimation: Theory, Practice and Vizualisation*, Prentice Hall, 1992.

[24] M. Siekkinen, D. Collange, G. Urvoy-Keller, and E. W. Biersack, "Performance Limitations of ADSL Users: A Case Study", In *Proc. Passive and Active Measurement: PAM 2007*, April 2007.

[25] M. Siekkinen, G. Urvoy-Keller, and E. W. Biersack, "On the Interaction Between Internet Applications and TCP", In *Proc. 20th International Teletraffic Congress: ITC*, June 2007.