PPrate: A Passive Capacity Estimation Tool

Taoufik En-Najjary
Institut Eurecom, France
Email: ennajjar@eurecom.fr

Guillaume Urvoy-Keller
Institut Eurecom, France
Email: urvoy@eurecom.fr

Abstract— **Knowledge of link capacities is essential, e.g., for ISPs to troubleshoot paths outside the boundaries of their networks. However, the vast majority of capacity measurement tools are based on active probing, which is not suitable for large scale studies of Internet paths characteristics. In this paper we present PPrate, a completely passive tool, that can extract capacity information of a path from the packet trace of a TCP connection. We validate PPrate using synthetic traces and real traces collected on PlanetLab. We compare PPrate with Pathrate, which is very accurate active tool, and show that they perform comparably. We finally apply PPrate on a large publicly available ADSL trace.**

I. Introduction

Measuring network characteristics such as *capacity* or *available bandwidth* is important for wide-area Internet services and distributed systems. The capacity of a path is defined as the maximum IP-layer throughput that a flow can get on the path. It is determined by the link with the minimum transmission rate. We refer to that link as the *narrow link* of the path, following the convention of [18]. The available bandwidth is the maximum rate at which a new flow can send without impacting the rate achieved by the existing flows.

As noted in previous work [6], [19], knowledge of network characteristics can be useful in many cases: ISPs can acquire a good picture about the characteristics of their own links and links outside the boundaries of their network, and plan their capacity upgrades. Also, multimedia servers would be able to determine appropriate codecs and streaming rate. So far, a large number of tools and techniques have been proposed to estimate the capacity of a network path [10], [15], [6], [8], [3], [19], [13], [11]. A vast majority of these techniques employ *active* approaches to probe the networks and estimate link capacities. Active estimation is based on the injection of measurement packets in the network. From an end-user's perspective, using active measurements to assess the capacity of a network path is a reasonable approach. However, for large scale study of Internet paths characteristics, active measurements become difficult because of the probe overhead and the need not to bias the characteristics being measured. Also, active measurements often assume access to both the sender and the receiver, which prevents large scale measurements; and they cannot be applied on the large set of Internet traces collected over the years by the traffic analysis community. A set of accurate, passive, capacity measurement tool is therefore required.

This paper presents PPrate, an accurate tool for the passive measurement of path capacity. PPrate uses algorithms similar to the ones of Pathrate [6] to passively discover capacities from traces collected at the sender or the receiver-side. PPrate uses packet or *ack* inter-arrival times to investigate questions about the link capacities along a path.

The paper is organized as follows. In Section II, we survey the related work about capacity estimation. In Section III, we further discuss the notion of packet pair dispersion and provide a short overview of Pathrate. We present PPrate in Section IV. We present validation results using simulations and Planetlab experiments in Section V. We demonstrate the use of PPrate on a publicly available ADSL trace in Section VI. Some conclusions and perspectives for future work are presented in Section VII.

II. Related Work

Many 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 in 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 possible. In practice, dispersion or compression of the packet pair might occur before or after the narrow link, due to cross traffic. Expansion leads to under-estimation and compression of dispersion leads to over-estimation of the narrow link capacity.

Paxson [16] shows that the packet pair dispersion distribution 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 [2], [15], [12].

To eliminate cross-traffic effects, various refinements have been proposed, including sending packet trains of various sizes [3], [16], and better filtering techniques to discard incorrect samples [13], [12]. The filtering problem 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 [10], Clink [8], and pchar[15]. However, delay measurements rely on ICMP time-exceeded messages from routers, which limits both the applicability and the accuracy of these tools (for a study of these tools, see [9]).

III. 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 present a short description of Pathrate.

A. Packet Pair dispersion

Formally, the dispersion of a packet pair can be described as follows. Consider a network path $\mathcal{P}$ defined by the 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 the cross-traffic invalidates this assumption. It has been observed that cross-traffic can causes compression or expansion, and subsequently, an under or over-estimation of the capacity.

Compression may 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 shorter time 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. Those capacity values $C = L/\Delta_P$ will thus form a certain distribution B. The challenge is to infer the 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 shows [6] that this not the case. Indeed, the use of packet train leads to the estimation of the so-called "Asymptotic Dispersion Rate" (ADR). The ADR has been shown to lie between the available bandwidth and the capacity of the path [6].

B. Pathrate

Pathrate is based on the dispersion of packet pairs and packet trains. To circumvent the problem of multi-modality of the packet-pair dispersion distribution, Pathrate uses packet pairs to uncover a set of possible "capacity modes". It further uses long packet trains to estimate the ADR and select the capacity mode. We provide a brief description of the main phases of Pathrate:

- Phase I: Packet pair probing. In this phase, Pathrate generates a large number (1000) of packet-pairs. The goal here is to discover all local modes in the packet-pair bandwidth distribution B[1]. One of the Phase I modes is expected to be the capacity of the path. The packets that Pathrate sends in Phase I are of variable size, in order to make the non-capacity local modes weaker and wider.
- Phase II: Packet train probing. In this phase, pathrate generates 500 trains of N packets and measure the packet trains dispersion[2]. Gradually, the resulting bandwidth distribution $B(N)$ of the packet train dispersion becomes unimodal, centered at the so called Asymptotic Dispersion Rate R. The capacity mode is selected as the strongest and narrowest phase I mode among those larger than R. In fact, two methods have been successively used in Pathrate to pick the capacity mode. We futher discuss this point in Section IV-B.

Pathrate was designed to be robust to cross-traffic effects, meaning that it can measure the path capacity even when the path is significantly loaded. However, the lack of scalability and the need to access to the two end points of the path, make Pathrate not suitable for a large scale study of Internet paths characteristics. To address those limitations, we present in the next section PPrate, a passive measurement tool, that use techniques similar to the ones of Pathrate.

IV. PPrate

A. Introduction

A requirement for a passive capacity measurement tool is to be able to handle TCP connections. Indeed, despite the emergence of new applications (e.g. p2p file sharing) that have deeply altered the Internet traffic characteristics, TCP is still the most popular transport protocol in the Internet today. In addition, TCP is a natural candidate for capacity measurement techniques based on the packets dispersion principle, as with the delayed *ack* strategy, a TCP sender often injects packets in pairs in the network.

PPrate takes as input a *tcpdump* trace, or a set of inter-arrival times obtained by other means, and automatically estimates the capacity of the narrow link. PPrate can work with measurements collected at the receiver side or at the sender side. 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 next present the more intricate sender side case where one can only indirectly observe the packet pairs arriving at the receiver using the ACK streams. In the last part of this section, we discuss the influence of the TCP layer and the application on top of TCP on the quality of the estimation.

[1]Note that we use two equivalent representations of the packet pair dispersion histograms. Either we plot directly the histogram of the inter-arrival times IAT_i or we plot the bandwidth histogram of $8.MSS/IAT_i$, where MSS is the maximum segment size in bytes of the connection.

[2]Defined as the time elapsed between the arrival of the first and the last packet of the train at the receiver.

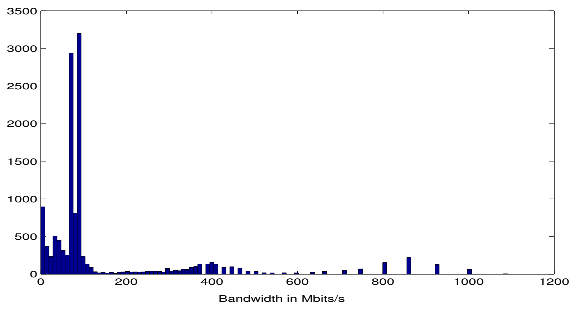

Fig. 1. Raw Bandwidth Histogram

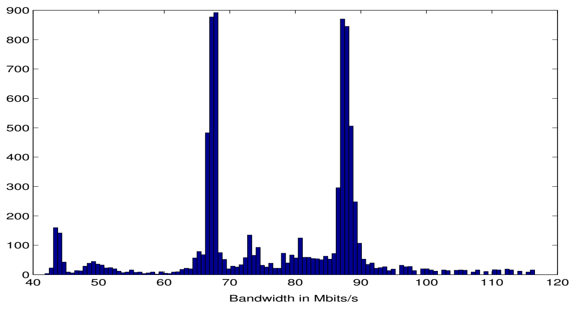

Fig. 2. Preprocessed Bandwidth Histogram

B. Receiver side algorithm

Assume a packet trace captured at the receiver side. The packets inter-arrival times can be seen as a time series of packet pair dispersions (similar to the case of active probing). So we can use it to form the bandwidth distribution of the path, which is multi-modal in general. We present in Figure 2 an example of such histogram. Preprocessing of the histogram (data cleaning) will be detailed in Section IV-D. The local modes in the packet pair dispersion distribution are candidate values for the capacity of the path. The challenge, here, is to select the local mode corresponding to the narrow link.

The main idea here is that by aggregating the inter-arrival durations N by N, we can build the distribution of packet trains $B(N)$; and, as in the active probing case [6], by increasing N, make $B(N)$ unimodal. By analogy, we refer to the center of the *unique* mode as the ADR.

Algorithm 1 shows the pseudo-code of PPrate. Initially, *PPrate*, constructs a packet pair dispersion distribution 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 [4]. 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 [20]. 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 aggregates the inter-arrival times samples N by N, constructs the packet trains dispersion $B(N)$ and scans it for modes. PPrate starts with $N = 3$, and repeats the procedure till $B(N)$ becomes unimodal. PPrate chooses that mode as the Asymptotic Dispersion Rate R. 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 following conclusions:

- If there is a large number of samples to form the packet pairs dispersion histogram, say over 5000, there is no significant discrepancy between the results returned by the two methods;
- Conversely, if there are less than 5000 samples, the estimation returned with the first method is more prone to errors.

The main reason behind the above observations is that the first method is more likely to pick a non significant peak in the histogram as the capacity peak. This situation is less likely to occur when the number of samples increases as non significant modes should disappear. Considering the case of Figure 2, if R is 70 Mbits/s, the first algorithm might pick the mode at value 80 Mbits/s as the capacity mode, whereas the second method will chose the peak at 90 Mbits/s. We will further illustrate the two above observations on the experiments we made on Planetlab in Section V-B.

```
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
    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

C. Sender Side Algorithm

If the traffic was not captured at the receiver side, it is impossible to know the time dispersion of the packet pair

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

arriving at the receiver. However we can use the dispersion of the pure *acks* (that carry no data) as an approximation of the packet pair dispersion at the receiver. Indeed, pure *ack* packets are 40-bytes packets, having no payload data, and generally acknowledge two data packets due to the delayed *ack* strategy. We "cancel" the effect of delayed *acks* by dividing by two the inter-arrival times of *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.

D. Impact of TCP layer and application on top

PPrate uses as input packets or *acks* 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 layer 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 case of the TCP layer. The main hindrance caused by TCP is the constraint imposed by the receiver advertised window on the maximum observable size of the trains. Indeed, if PPrate can't observe long enough packet trains, it will be difficult to correctly estimate the Asymptotic Dispersion Rate R. In practice, it turns out that the packet trains sizes that we require to form R is relatively small as compared to commonly observed advertised window sizes in the Internet, as we will see on simulation results in Section V and on real traces in Section VI.

As for the application on top of TCP, it can 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] that uses the two directions of a TCP connection to transfer data. In this case, PPrate can be applied at the receiver side, upon reception of the packet pairs from the other party, but not on the sender side as *acks* are likely to be piggybacked. Note however than when a peer has completed the download of the file and is only acting as a source of data for the other peers in the BitTorrent session, PPrate can be applied at the sender side and at the receiver side.

Second, the application can generate idle periods on the TCP channel if 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 queries on the same server. Long idle periods will lead to long packet inter-arrival time samples. This will 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 values that are larger than $min(p_{75} + IQR, p_{95})$. The latter formula means that we remove either the samples that are far away from the 75$^{\text{th}}$ 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 any application effect but against some measurement errors [17] that lead to extremely small interarrival times. To illustrate the impact of those filtering mechanisms, consider the bandwidth histogram in Figure 1 obtained from some raw data. We observe on this histogram abnormaly high bandwidth values, up to 1 Gbits/s. Figure 2 depicts the same histogram once raw data has been filtered.

Third, the application can enforce some rate limitations. This is the case for instance of many p2p applications like E-donkey and BitTorrent. We note however that many different techniques exist to limit the rate of an application [21]. For instance, the application can deliver small amounts of data to the TCP layer and set the PUSH flag to force TCP to output packets smaller than the MSS. This can be observed for some E-donkey clients [21]. Another option for the application is to send bursts of bytes (that will generate bursts of MSS packets at the TCP layer) separated by idle periods. This can be observed with some BitTorrent clients [21]. 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 [21]. We have already integrated both methods in a single tool and conducted some preliminary experiments on some p2p traffic. However, in the context of this paper, we will focus on some applications that should not apply rate limitations, such as FTP, SCP and HTTP transfers.

V. VALIDATION

In this section, we validate our tool using both simulations and experiments over the Internet using Planetlab.

A. Validation on Simulated Network

In this section we investigate the accuracy of PPrate via simulation. In our experiments, we used the ns-2 [1] simulator, with the configuration shown in Figure 3, consisting of a 9-hop path.

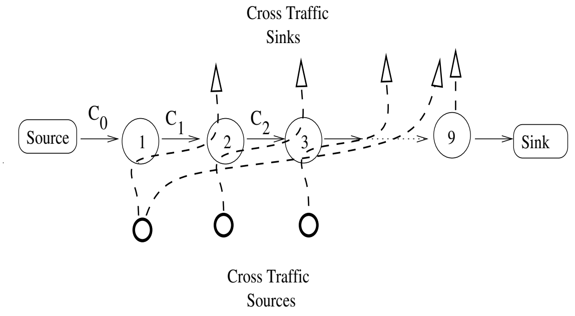

Fig. 3. Simulation configuration.

We ran several simulations for different loads and narrow link capacity values. Each simulation consists of a 30 second FTP transfer from a source to a sink. In order to produce Long Range Dependant (LRD) cross-traffic, we used a number of Pareto sources with shape parameter $\alpha = 1.9$. The aggregation of many Pareto sources with $\alpha < 2$ has been shown to produce LRD traffic [22]. Cross-traffic is both path-persistent and non-persistent. Cross-traffic is path-persistent when its packets follow the same path as the FTP packets, whereas the non-persistent cross-traffic packets exit one hope after they enter the path. This scenario is similar to the one used in [6], [11].

We first investigate the accuracy of the estimation made with PPrate for different loads and narrow link capacity values. We next discuss the impact of the advertised window value of the FTP transfer on the PPrate algorithm.

Cross-traffic effect: We first investigate the cross-traffic effect. To do so, we vary the narrow link capacity (link between nodes 2 and 3 in Figure 3) from 5 Mbits/s to 100 Mbits/s, and its load from 30% to 80%.

Tables I and II summarize the results of the PPrate estimation using data packets and *ack* packets. Overall, we observe that PPrate always outputs accurate results for the whole range of load and narrow link capacity values that we consider. A comparison between the two tables shows that the procedure we devised to handle *ack* streams (esp. when delayed acknowledgments are used as it is the case here) enables to obtain accurate estimates of the capacities. Note however that the load on the reverse path is low in the context of our simulations. We further exemplify the use of PPrate with *ack* streams in Section VI. We also observe that the estimation error might be either positive or negative (over or under estimations of the narrow link capacity). This is to be expected as PPrate, like Pathrate is based on the dispersion of packet pairs that is prone to both types of errors (see Section III).

Set capacities (Mbits/s)	Narrow link load		
	30%	50%	80%
5	5	5	5
10	10	10	10
20	20	19.5	20
40	39.1	39.9	42.5
55	54.8	57.3	59
100	100	97.5	99

TABLE I
CAPACITY MEASUREMENTS ON DATA INTER-ARRIVALS

Set capacities (Mbits/s)	Narrow link load		
	30%	50%	80%
5	5	5	5
10	10	10	10
20	20	18.5	19
40	39	39	41.1
55	54.8	54	53
100	100	99.3	98

TABLE II
CAPACITY MEASUREMENTS ON *acks* INTER-ARRIVALS

TCP advertised window effect: As we aggregate several inter-arrivals to estimate the ADR, we can expect that the TCP advertised window value will affect the accuracy of PPrate. To investigate this issue, we vary the advertised window of the receiver of the FTP stream as a function of the bandwidth delay product of the path. We plot in Figure 4 the ratio of the capacity estimation made when the advertised window is set to $x\%$ of the bandwidth delay bandwidth product of the path to the capacity estimation made without any advertised window limitation (i.e. $x = 100$). The value of the advertised window has no impact when the ratio is equal to 1. We plot in Figure 4 the case when the estimation is performed on the *ack* stream, which constitutes a worst case from the tool point of view. We observe that only for very small values of the advertised window, the estimation becomes inaccurate. When working on the data stream, where we have two times more samples, the estimation remains accurate (ratio of 1) even if the advertised window is equal to 5% of the delay bandwidth product of the path. This result suggests that one in general needs small train sizes (N values) to infer the ADR of the path. We further investigate this issue in Section VI.

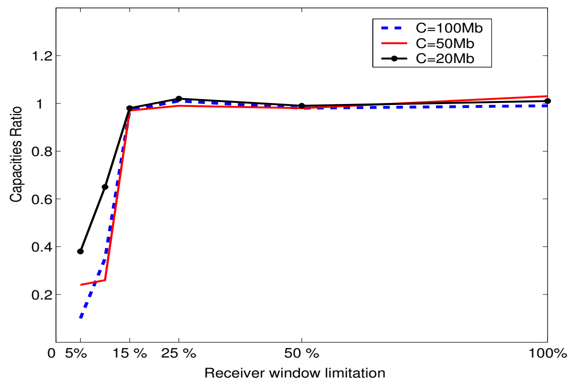

Fig. 4. Ratio of PPrate estimations made with $x\%$ and without receiver window limitations

B. Planetlab Validation

PPrate is derived from Pathrate. We thus need to compare the two tools with one another. Our methodology consists in comparing the estimations given by Pathrate and PPrate on the same paths under similar conditions. We use Planetlab[4] machines. Planetlab is an attractive platform for the comparison of Pathrate and PPrate as it enables us to connect on the two sides of a path, which is a requirement to run Pathrate.

We selected 33 paths between PlanetLab nodes. On each path, we first conduct one *scp* transfer (of a 20 Mbytes file), and collect a tcpdump trace at the receiver side. Note that PlanetLab does not allow collecting traces at both end points of a path. Immediately after the *scp* transfer, we run Pathrate on the same path. This procedure is repeated 10 times for each path. As Pathrate returns two estimates for each run [6], we use their average. As it takes on average 15 to 30 minutes for Pathrate to output an estimate and 1 to 5 minutes to perform an *scp* transfer, the total number of measurements span over a few hours for each path.

Figure 5 summarizes the results for the 33 paths that we considered. A given index on the x axis corresponds to a

[4]http://www.planet-lab.org/

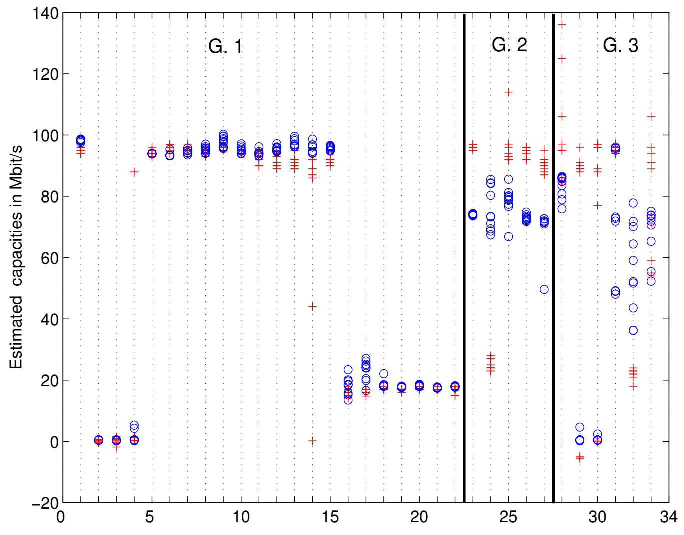

Fig. 5. This graph shows the bandwidth reported by PPrate (o) and Pathrate (+) along 33 paths on PlanetLab.

single path. For each path, we use circles to depict the 10 estimates returned by PPrate and crosses for the 10 estimates returned by Pathrate. The method proposed in [6] is used to pick the capacity mode. Results when the method in [7] is used, are similar to the ones in Figure 5 due to the large number of samples (approximately 12,000) obtained with the *scp* transfer, as stated in Section IV-B. To ease interpretation of the results, we have formed 3 groups of paths labeled as group 1, 2 and 3 on Figure 5. In group 1, we put the paths for which each tool returns a consistent estimates and, in addition, the estimates of the two tools are consistent with one another. This group consists of 22 paths, that is over two third of the paths. Group 2 consists of paths for which each tool return consistent estimates but the two tools do not fully agree with one another. A relatively small number of paths (4) fall in this group. Group 3 consists of the paths for which one or both tools return non consistent estimates. We note that both tools might exhibit a high variance in its estimates. It is difficult to comment on the paths in this group as many factors might explain why the tools do not work properly. We hypothesize that a possible reason is the use of slices and the rate limitations on Planetlab nodes. We leave for future work a more in-depth investigation of those issues, and we note that other studies of Pathrate over Planetlab nodes have also observed a number of problems [14].

The main conclusion that we draw from those experiments is that for a significant number of cases, Pathrate and PPrate fully or partially agree with one another.

Another issue that we want to address with those Planetlab experiments is the impact of the number of samples used by PPrate on the consistency of the estimation. To tackle this issue, 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 on the first 300, 500, 1000, 2000 or 5000 first packets of the transfer. We next compute for each case, the ratio between the estimate obtained with the

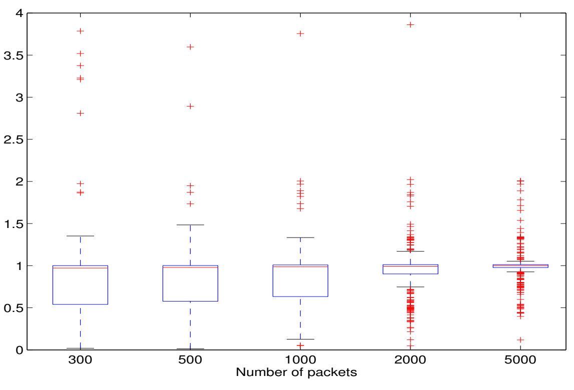

Fig. 6. Boxplots of ratio between the estimates obtained with x packets and the ones obtained with the full FTP transfer - version 1 of PPrate

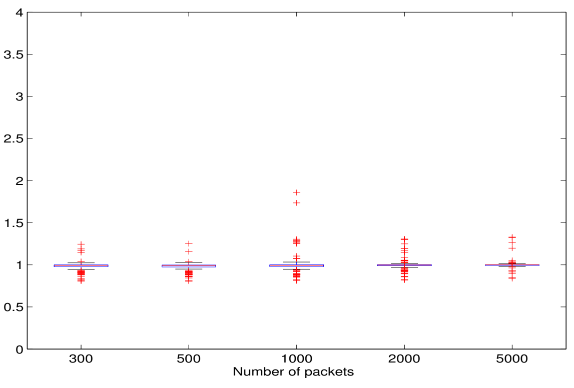

Fig. 7. Boxplots of ratio between the estimates obtained with x packets and the ones obtained with the full FTP transfer - version 2 of PPrate

first x packets to the estimate obtained using all the packets. Ideally, if the ratio is one for all x values, this means that PPrate provides a consistent estimate whatever the number of samples used is. We plot in Figures 6 and 7 the boxplot[5] for the different x values for all the transfers on all the paths. The difference between Figures 6 and 7 is the method used to pick the capacity mode. In Figure 6, we use the method from [7] (first peak larger than R), while in Figure7 we use the method from [6] (largest peak larger than R). We first observe from both figures that as x increases, the distribution of the estimates obtained with the first x packets are consistently closer and closer to the estimates obtained with all the samples. However, we also observe that the second version of PPrate is able to work with a significantly smaller number of samples than the first one. Indeed, the estimates returned by the second version of PPrate for $x = 300$ are already close to the ones obtained with all samples as most of the mass is close to 1 in the boxplot. Those results are consistent with the explanations

[5]A boxplot is a graphical representation of a distribution where the upper line of the box is the 75th percentile p_{75} of the distribution, the lower line is the 25th 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 those two lines. Values outside those boundaries are marked with a cross as they are extreme values of the distribution.

provided in Section IV-B on the robustness of both versions of the algorithm.

VI. ADSL Traces Analysis

We applied PPrate on two traces extracted from the same public set of traffic traces of an ADSL access network[6]. The first trace consists of FTP connections while the second trace consists of HTTP connections. All the traffic was collected by a probe located 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. We applied PPrate on the *ack* streams of the FTP and HTTP connections which should, with high probability, flow from the ADSL clients to servers outside the ADSL platform as we can reasonably expect most of the FTP and HTTP servers targeted by those connections not to be hosted by ADSL clients. We selected the *ack* streams with more than 300 samples, which represent 20% of the observed *ack* streams for the FTP connections and 8% of the *ack* streams for HTTP connections. We plot in Figure 8 the distribution of the capacities estimated by PPrate on both the FTP and HTTP connections. We first observe from Figure 8 that there is a quite good agreement between the distribution of capacities for the FTP connections and the HTTP connections. Note that we could not expect a perfect matching between those two curves as the anonymisation of the traces prevent 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 sample connection. We also observe from Figure 8 that approximately 72% of the estimated capacities fall in the interval [256kbits/s, 8Mbits/s]. In addition, we do observe for these capacity values some peaks around values close to 500 kbits/s, 1 Mbits/s and 8 Mbits/s, which are typical capacities of ADSL clients. The reason why 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. As for the overestimation, they represent less than 5 % of the FTP connections and less than 10% of the HTTP connections. On the other hand, values that fell below 256 kbits/s represent around 18% of the values. As discussed in Section IV-D, those underestimations might be due to the application on top of TCP. We expect to obtain better results by first isolating periods in connections where the application does not limit the rate of the connection [21]. We leave this topic as future work. However, we note that the FTP traffic is more likely than the HTTP traffic to consist only of traffic that is limited be the network and not by the application; and we can indeed observe that the smallest capacity estimation obtained with the FTP traffic is 15 kbits/s for the FTP traffic while it is 0.6 kbits/s for the HTTP traffic.

PPrate needs to estimate the ADR of a path to correctly infer its capacity. The ADR is computed for the smallest N value for which the dispersion of trains of size N is unimodal.

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

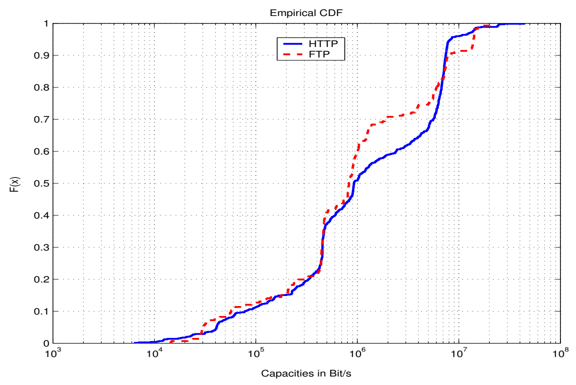

Fig. 8. Capacity estimates for the *ack* streams of FTP and HTTP connections

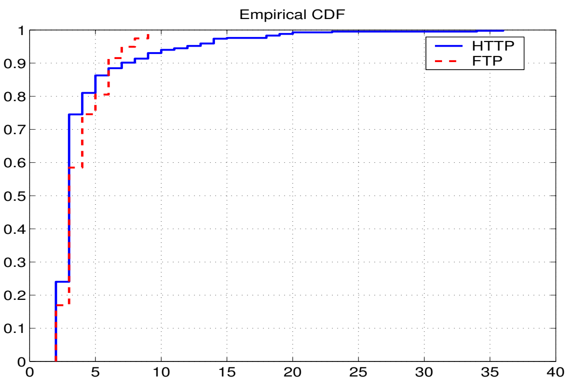

Fig. 9. N values used to estimate the ADR

Intuitively, if N is small as compared to the advertised window of the path, we can expect the estimation of the ADR to be made correctly. We present in Figure 9 the distribution of N for all the connections that we considered. We distinguish in the figure between the FTP and HTTP connections. Overall, we observe that for both types of traffic, 80% of the ADR have been estimated with trains of less than 5 samples and over 93% with less trains of less than 10 samples. As we apply PPrate on the *ack* streams, 5 (resp. 10) samples correspond to 10 (resp. 20) data packets. These are relatively small values if one considers that an advertised window of 65 Kbytes corresponds to approximately 43 packets of 1500 bytes.

VII. Conclusion

In this paper we studied the feasibility of a passive estimation of the capacity of an Internet path. We propose PPrate, a tool based on the Pathrate algorithm, one of the more robust and accurate active capacity estimation tool. Simulations show the high accuracy of the proposed tool under different load conditions. Experiments on PlanetLab further demonstrate that the accuracy of PPrate is comparable to the one of Pathrate. We also exemplify the use of PPrate on a large ADSL trace.

As future work, we plan to study the possibility to do capacity measurements ”on line”, and the ways to incorporate PPrate’s techniques into Internet applications (e.g. p2p systems) that would benefit from rapid and accurate capacity estimations. PPrate may be downloaded from: `http://www.eurecom.fr/~ennajjar/`.

References

[1] "The Network Simulator - ns-2 : http://www.isi.edu/nsnam/ns/".

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

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

[4] 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.

[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] V. Jacobson, "Pathchar: A tool to Infer Charcteristics of Internet Paths", *ftp://ftp.ee.lbl.gov/pathchar/*, 1997.

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

[12] 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, New York, NY, USA, 2004, ACM Press.

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

[14] 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.

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

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

[17] V. Paxson, "Strategies for sound internet measurement", In *IMC'04*, pp. 263–271, New York, NY, USA, 2004, ACM Press.

[18] R. S. Prasad, M. Murray, C. Dovrolis, and K. C. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools", *IEEE Network*, 17(6):27–35, November 2003.

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

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

[21] M. Siekkinen, G. Urvoy-Keller, and E. W. Biersack, "On the Impact of Applications on TCP Transfers", RR-05-139, Institut Eurecom, October 2005.

[22] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar traffic modeling", *SIGCOMM Comput. Commun. Rev.*, 27(2):5–23, 1997.