Capacity Estimation of ADSL links

Daniele Croce, Taoufik En-Najjary, Guillaume Urvoy-Keller and Ernst W. Biersack
EURECOM
Sophia Antipolis, France
Email: {croce,ennajjar,urvoy,erbi}@eurecom.fr

Most tools designed to estimate the capacity of an Internet path require access on both end hosts of the path, which makes them difficult to deploy and use. In this paper we present a single-sided technique for measuring the capacity without the active cooperation of the destination host, focusing particularly on ADSL links. Compared to current methods used on broadband hosts, our approach generates two orders of magnitude less traffic and is much less intrusive. Our tool, *DSLprobe*, exploits the typical characteristics of ADSL, namely its bandwidth asymmetry and the relatively low absolute bandwidth, in order to measure both downlink and uplink capacities and to mitigate the impact of cross-traffic. To further improve the accuracy, we study different ways to detect and filter cross-traffic packets and we show how to recognize and overcome limited uplink capacities. We validate our tool both on controlled hosts and on a wide variety of Internet hosts. Finally, we present a case study of two large ADSL providers.*

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks;
C.4 [**Computer-Communication Networks**]: Performance of Systems

General Terms

Measurement, Experimentation, Performance

Keywords

Capacity estimation, ADSL, measurement, asymmetry, non-cooperative, DSLprobe

*This work is supported in part by the NANODATACENTERS program (FP7-ICT-223850) of the EU.

ACM CoNEXT 2008, December 10-12, 2008, Madrid, SPAIN

1. INTRODUCTION

Capacity is the most important and most advertised characteristic of a network link. It is critical for achieving good end-to-end performance over the Internet. Many techniques have been developed in order to estimate the capacity of single links or the capacity of the bottleneck link of a given path [5, 8, 9, 12–15, 20]. Most of these techniques either rely on routers that reply to ICMP packets or require access to both end hosts of the path, significantly limiting and complicating their usage. None of them has been designed to work specifically in non-cooperative ADSL environments.

ADSL and broadband technology has become the standard access technology for over 200 million residential users [18]. The number of broadband subscribers has increased by 187% since Dec. 2004 and it is supposed to double in the next four years [23]. In many countries, more than half of all households are connected through a broadband access link. These networks have marked the success of the new era, allowing the widespread usage of bandwidth intensive applications such as video-on-demand, online gaming, peer-to-peer video streaming, distributed content delivery systems. The explosion of media content has thus pushed up the demand for higher capacities at the access links, which today represent the bottleneck on Internet paths [6]. Over 62% of broadband links are based on DSL technology.

Despite the importance of these networks, little research has been made on broadband links, mostly because of the lack of tools for analyzing them without the explicit cooperation of the end hosts or the ISPs. To the best of our knowledge, the only existing work presenting a large-scale characterization of broadband networks is the one presented in [6], where the capacity of the links was measured using buffer saturation, a quite intrusive technique. In this paper we present *DSLprobe*, a new single-ended tool for the capacity estimation of ADSL links, designed especially for non-cooperative environments[1]. DSLprobe sends two orders of magnitude less traffic and is less intrusive compared to the technique in [6]. In DSLprobe we exploit the fact that the capac-

[1]Similar to [4, 20], with "non-cooperative" we mean that access on the measured end-hosts is not required.

ity of the ADSL links is low compared to the available bandwidth of the backbone links. This allows to send high-speed trains of packets, reducing the likelihood of interference from cross-traffic, i.e. packets flowing with our probes through the same link(s). The probes sent are TCP ACKs, whose size can vary between 40 and 1500 Bytes. These ACKs are sent to the ADSL host in order to solicit TCP RSTs back to the source. Using the RSTs flowing back it is possible to measure the capacity of the ADSL link as well as to obtain some information on interfering cross-traffic. In DSLprobe we implement different techniques based on the analysis of the IP-IDs and on the Inter-Arrival Time (IAT) of the RSTs in order to detect and filter the cross-traffic packets interfering with the probe trains. We also take advantage of the capacity asymmetry of ADSL to measure both, the uplink and the downlink capacities. We investigate how to correctly set the size of the probes and pay attention to the overhead imposed by low layer protocols and to the capacity limits of the uplinks. Finally, we validate DSLprobe both on controlled ADSL hosts and on the Internet, and we provide a case study of two large ADSL providers.

2. BACKGROUND AND RELATED WORK

2.1 Capacity estimation techniques

Most existing tools estimate the capacity by sending a certain number of packets with particular size and distribution, measuring how these packets arrive after traversing the link. Generally, the packets traverse more than one link, so the tools either estimate the capacity of each link traversed, or the bottleneck capacity of the path *i.e.* the link with lowest capacity.

2.1.1 Variable Packet Size

The Variable Packet Size (VPS) technique aims at measuring the capacity of every link of the path. It was first described by [2] and used in tools like *pathchar* [12], *clink* [9] and *pchar* [15]. The basic idea in VPS is to measure the delay variation when increasing the size of the probes: for a generic link of capacity C_i, the transmission delay increases linearly with the size of the packet as $1/C_i$. Just like the *traceroute* tool, in VPS tools the TTL field is adjusted so to measure the RTT up to a certain hop. Then the size of the probes is modified and the RTT is measured again. If there is no queuing due to cross-traffic, the difference in delay is due solely to the difference in size and the capacity of the link is obtained by recursively measuring *all* previous links. VPS can thus suffer from error propagation. Additionally, in order to obtain delay samples unaffected by cross-traffic, several hundreds (or thousands! [9]) probes must be sent and only the sample with the minimum delay is considered. Finally, VPS suffers from layer-2 store-and-forward devices, which are invisible at layer-3 but add serialization delay [19].

2.1.2 Packet Pairs

The Packet Pair (PP) technique measures the bottleneck capacity of a path. When two packets are sent one after the other ("back-to-back"), they will be received at the end of the path spaced in time and, if there is no cross-traffic, the spacing (or *dispersion*) between the packets is inversely proportional to the capacity of the bottleneck link. The PP technique is implemented for example in *bprobe* [5], *Nettimer* [14], *SProbe* [20], *pathrate* [8] and *CapProbe* [13].

CapProbe, the most recent of these tools, sends a number of pairs and computes the capacity from the pair that has *minimum delay sum*: if d_{1i} and d_{2i} are the delays experienced by the two packets of pair i, CapProbe computes the capacity from the pair where $d_{1i} + d_{2i}$ is minimum. The assumption here is that if a pair is affected by cross-traffic, then the delay of one of the two packets will increase, otherwise it will remain at a minimum value. As we will explain, it is this condition that makes CapProbe (and PP techniques in general) unsuitable for non-cooperative ADSL environments.

2.1.3 Packet trains: Average Dispersion Rate

An approach that is conceptually similar to PP is to use Packet Trains. The idea is to send L back-to-back packets of size S and to measure at the receiver the Average Dispersion Rate (ADR) defined in [8] as $R = (L-1)S/\Delta$, where the *dispersion* Δ is the time between the arrival of the first and the last packet of the train. Again, if no cross-traffic is present, the dispersion of the train will be due solely to the bottleneck link and the ADR will be equal to the capacity. ADR estimations are used in *pathrate* [8] and *pathload* [7]. Compared to PP, the ADR is more robust to outliers and less sensitive to errors and timestamp granularity (the dispersion is measured over more packets) but the probability that a cross-traffic packet interferes with the train of probe packets is higher.

2.1.4 Buffer saturation

Another way to estimate the capacity is to saturate the buffer with a stream of packets at high rate and to measure the amount of traffic that traverses the link. If the rate of the flow is very high, cross-traffic packets will find no place in the buffer and will be dropped, leading to good estimates. Indeed, the probability that a cross-traffic packet traverses the link decreases when the rate of the probes increases. Buffer saturation can thus be very intrusive. If the cross traffic consists of TCP flows, causing persistent loss to these flows can severely affect their throughput or even kill the connection completely.

2.2 The MPI tool

In [6] Dischinger et al. from the Max Planck Institute of Software Systems used a tool, which we refer to as the *MPI tool*, to analyze different types of broadband networks. The aim of the MPI tool is to characterize

the network access of *non-cooperative* ADSL and Cable hosts with respect to various metrics including allocated bandwidth, buffer size, round trip times, traffic shaping, and queue management policy. In order to measure all these characteristics and to account for the differences between Cable and ADSL access, the MPI tool is very general and includes various types of tests.

To estimate the bottleneck capacity, the MPI tool saturates the buffer of the access link. All measurements affected by cross-traffic are discarded. Using measurements to many different end hosts, it is shown that for broadband access networks the last hop is always the bottleneck and that backbone links can support flows of 10 Mbps without significant loss. To measure the capacity of the broadband link a flow with rate of 10 Mbps is sent and the capacity is computed by counting the amount of packets that traverse the path without being dropped. Both the downlink and uplink measurements lasted 10s which means that in total over 15,000 packets were sent towards the ADSL host. They did not provide information on when the tool was failing. The main thrust of that paper [6] was not to describe in detail the techniques used to measure the various metrics but to present the results of the empirical measurements towards a large number of broadband hosts. The authors of the MPI tool where kind enough to provide us the code of their tool, which allowed us to study its operation and to compare the capacity estimates obtained by the MPI tool with the ones of DSLprobe (see Sec. 8).

As we will show, the advantage of DSLprobe is that, focusing on the capacity estimation of ADSL links, allows to obtain excellent results without the need of saturating the access link buffer. DSLprobe generates two orders of magnitude less packets and drastically reduces the intrusiveness of the measurements. Moreover, in DSLprobe we introduce sophisticated techniques to filter or mitigate the impact of cross-traffic whereas the MPI tool only tries to detect cross-traffic. Finally, the MPI tool fails when the link is too much asymmetric.

3. WHY ARE PACKET PAIRS INFEASIBLE?

In the PP technique, capacity is estimated directly from the IAT of the two packets. It is thus extremely important to discard all pairs compressed or expanded by cross-traffic. The solution proposed in CapProbe [13] is probably the most simple and effective since it assures that the PP from which the capacity is computed is the one that suffered lowest delay (and possibly no queuing at all). In the ADSL environments, however, the IAT of packet pairs is not always constant but can vary with some granularity. This variability around specific values has already been noticed in [3] and an example is given in Fig. 1. The figure shows how the IAT of 40 Byte packets traversing the uplink of an ADSL link can vary significantly around the average. For this experiment, the packets were captured from a well-connected host, there was no cross-traffic (both hosts were under our control) and we used UDP packets. Both machines where idle and the timestamp resolution was 1 us.

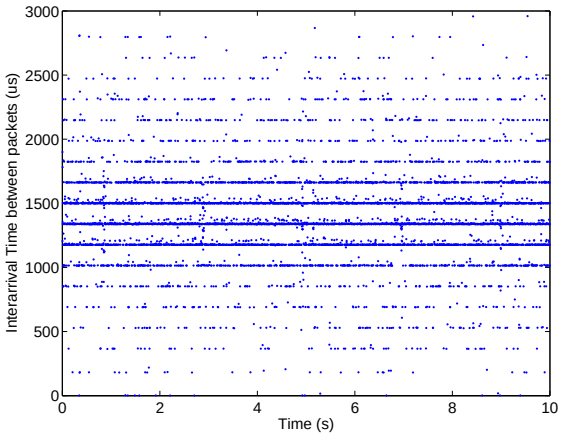

Figure 1: Inter-Arrival Time on an ADSL link. UDP packets are sent through the uplink of the ADSL and captured from a well-connected host.

The immediate consequence of this observation is that, even in absence of cross-traffic, the delay seen by a PP varies significantly around its mean[2]. It thus becomes very difficult to distinguish and filter out PPs distorted by cross-traffic. The problem of finding the real (average) link capacity is mitigated when using packet trains: using a larger number of packets, indeed, averages out the IAT variations. Of course, the problem of cross-traffic remains unchanged (actually it is even worsened with long train durations). As we will show, however, cross-traffic can be filtered out effectively in order to obtain accurate measurements.

4. ACTIVE MEASUREMENTS IN NON-COOPERATIVE ENVIRONMENTS

In a *non-cooperative* environment, it is not possible to capture the probes arriving at the *receiver*. A solution to this problem is to use probes that will be "reflected" back to the sender when they arrive at the receiver. This idea has already been adopted in literature [4, 6, 11, 20] in various forms and for different purposes using ICMP, UDP or TCP packets. Of particular interest for us are TCP probes, which have the peculiarity that the packets sent back to the sender are 40 Bytes long RSTs (or SYN/ACKs depending on the probes used) independently of the size of the probes sent to the receiver. As it will become clear soon, this

[2]To exploit the capacity of the local loop, ADSL providers implement complex signal processing and error correction algorithms and nowadays are providing together with Internet connection additional services such as IPTV or VoATM. This could lead to delay variations and to the granularity of the IAT shown in the figure. The investigation of the exact cause of these effects is beyond the scope of this paper.

possibility of sending large probing packets downstream but getting back small packets is fundamental for measuring, both, the uplink and downlink of the ADSL.

DSLprobe estimates the capacity by sending a train of TCP ACKs to a non-cooperative host and evaluates the dispersion of the train of the corresponding RSTs sent back to the source. Ideally, if no other packets interfere with the RSTs flowing back, the statistical properties of the ACK train will be preserved so that the dispersion of the train of RSTs will perfectly reflect the dispersion of the ACKs on the forward path. In reality, however, the measurement will be influenced by the characteristics of both, the forward and the reverse path. For example, if the RSTs traverse a low capacity uplink, the dispersion measured will be the one of the reverse path and not of the forward path. Additionally, cross-traffic can significantly compress or expand the trains biasing the measurement. When measuring ADSL hosts, in particular, the main challenges are:

- **Unknown Layer 2 overhead**. At Layer 2 (L2), protocols such as ATM, PPPoE or PPPoA that carry the higher layer packets can add significant overhead, especially when packets are small. The exact amount of overhead depends on the network architecture of the ISP and is generally not known.
- **High asymmetry.** If the capacity of the uplink is much lower than the downlink, the RSTs flowing back can be constrained by the uplink. This can bias the results when measuring the downlink.
- **Cross-traffic on the downlink.** Downlink cross-traffic causes an underestimation when estimating the downlink capacity. As we will show, downlink cross-traffic does not affect uplink measurements.
- **Cross-traffic on the uplink.** Because of the lower capacity of the uplink, large packets can take several tens of milliseconds to be transmitted. Thus, cross-traffic packets on the uplink can significantly bias the measurements.
- **Loss** is likely to occur when a link is congested and the buffer is nearly full. When loss is detected, we use smaller trains to reduce both the impact on the buffer and the probability of lost probes.

In the next section, we will discuss how to account for L2 overhead and correctly measure the forward and reverse paths even in high asymmetric conditions, while in Sec. 6 we will explain how to deal with cross-traffic.

5. ASYMMETRY AND OVERHEAD ON ADSL

In this section, we study how to exploit the properties of the ADSL to correctly measure both uplink and downlink capacities. Some of these concepts were already used in *ABwProbe* [4], though to measure a *different characteristic* (the available bandwidth) and facing different problematics, or in the MPI tool. In DSLprobe, we use these ideas in a different way, employing completely new techniques and methods.

5.1 Measuring both uplink and downlink: when asymmetry becomes an opportunity

Consider a generic path between two hosts, one being our measuring host, the other a non-cooperative ADSL host. Figure 2 represents a typical scenario for our measurements. Let now C_{down}, C_{up} and $s_{A/R}$ be respectively the downlink capacity, uplink capacity, and the ratio between the ACK size S_{ACK} on the forward path and the size S_{RST} of the corresponding RSTs on the reverse path ($s_{A/R} = S_{ACK}/S_{RST}$). Let MTU denote the size of the Maximum Transmission Unit, which is typically close to 1500 Bytes. Since ACKs can have any size, i.e. $40 \leq S_{ACK} \leq$ MTU, while S_{RST} is always 40 Bytes, we get $1 \leq s_{A/R} \leq$ MTU/40. At first, lets suppose there is no cross-traffic and that the size of the ACK probes is equal to the size of the RSTs ($S_{ACK} = S_{RST} = 40$, which then implies $s_{A/R} = 1$). Since every ACK will generate a RST flowing back, the rate R_{ACK} of the ACKs on the forward path and the rate R_{RST} of the corresponding RSTs on the reverse path will be equal. Thus, using $s_{A/R} = 1$ the dispersion measured with our train will match the link that has the lowest capacity, which is typically the ADSL *uplink*. If, instead, we want to measure the *downlink* of the ADSL host, we must increase the size of our ACK probes so that the load generated by the ACKs exceeds the downlink capacity before the corresponding RSTs saturate the uplink.

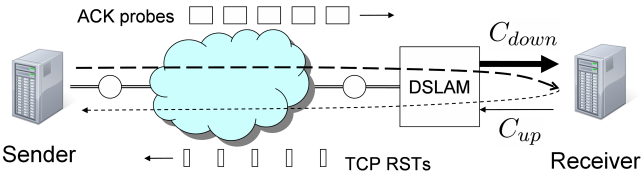

Figure 2: Non-cooperative estimation: the measuring host sends TCP probes with the ACK flag on. The receiver will reply with TCP RSTs.

To correctly measure the downlink, we must have $R_{RST} < C_{up}$ so that the RSTs are not constrained by the uplink. If we increase the size of the ACK probes traversing the forward path to, say $S_{ACK} = 1500$ Bytes, then on the uplink the rate of the RSTs will be $s_{A/R} = 1500/40 = 37.5$ times lower than the corresponding load on the downlink due to the ACKs, because $R_{RST} = R_{ACK}/s_{A/R}$. Since for most ADSL links the ratio $c_{d/u} = C_{down}/C_{up}$ between downlink and uplink capacity is generally lower than 15 (see [6]) the dispersion of the ACKs on the downlink will be sufficiently high so that the RSTs do not saturate the uplink.

5.2 Accounting for Layer 2 overhead

A major problem in computing $s_{A/R}$ precisely comes from L2 overheads. Most ADSL access platforms use ATM (together with encapsulation protocols such as PPPoA or PPPoE [22]): IP packets are segmented and carried in 53 Byte (comprising header and payload) ATM cells. This segmentation has great impact on the overhead, especially in case of small IP packets. While an MTU size IP packet is normally fragmented in 32 cells (1696 Bytes total, $\sim$ 13% overhead), a 40 byte RST will be carried in one or even 2 ATM cells for a total size of 106 Bytes. Indeed, in almost all ADSL architectures a 40 Bytes IP packet will be fragmented in 2 ATM cells [22], thus the number of Bytes transmitted at the physical layer will be $106/40 = 2.65$ times higher compared to the IP layer. This is important and must be taken into account for both the uplink and downlink measurements: for the uplink, because all results obtained with $s_{A/R} = 1$ (using 40 byte probes) must be corrected to account for this overhead; for the downlink, because at L2, the ratio between the size of an MTU packet (32 cells) and the size of a RST (2 cells) will reduce $s^{IP}_{A/R} = 37.5$ down to $s^{ATM}_{A/R} = 32/2 = 16$. Since usually the L2 infrastructure is not exactly known, it is good practice when measuring the downlink to be conservative and allow extra margin when setting the ACK size, i.e. to make ACKs as large as possible (MTU).

In some particular cases, however, increasing the ACK size might not be enough to overcome too big an asymmetry between downlink and uplink capacity. In this case, we *interleave* ACKs with other probes (such as TCP RSTs) that do not generate replies from the ADSL host. This way, it is possible to increase the downlink load further, reducing the amount of RSTs and therefore the load on the uplink: for example, if 30% of the probes sent do not generate a reply, then R_{RST} will be reduced by 30%, just as if $s_{A/R}$ increased.

6. CAPACITY ESTIMATION IN PRESENCE OF CROSS-TRAFFIC

6.1 Main ideas and approach

The great problem in correctly estimating the capacity is to filter or reduce the interference between cross-traffic and the probes. As we will explain in detail in this section, we exploit the following features to obtain accurate measurements:

- **High-speed trains**. We send our probes as fast as possible (our measuring hosts are generally connected with 100Mbps Ethernet links) in order to minimize the impact of cross-traffic. Indeed, the smaller the duration of a train, the lower the chance of cross-traffic interference.
- The IP identifier (**IPID**) is one of the fields in the IP header and is usually incremented for each packet sent. We use IPIDs to detect cross-traffic packets entering the train of RSTs on the reverse path (sent by the receiver).
- **Inter-Arrival Time (IAT) analysis** is undertaken to detect special cross-traffic interactions with the probes or when IPIDs are not usable.
- **Sequence numbers** are used to match the probes sent with the corresponding RSTs. Since TCP requires a RST to have as sequence number the acknowledgment number contained in the ACK that has triggered it, in each probe we use different acknowledgment number in order to relate ACKs and RSTs and to detect loss.

As we will explain in detail, in DSLprobe

1. We send trains of 40 byte ACK packets to measure the **uplink** capacity. Cross-traffic on the uplink is filtered using IPIDs or – if IPIDs are not available – analyzing the IATs.
2. We then send trains of MTU[3] size ACKs to measure the **downlink** capacity. Downlink cross-traffic is mitigated by using high rate trains. Uplink cross-traffic can *not* be filtered but must be detected in order to discard affected measurements. We learn about uplink cross-traffic by analyzing the IATs of the RST packets. By comparing the values with the uplink capacity estimated previously, we discover trains that are affected by cross traffic.

6.2 ADR as an approximation of the capacity

Dovrolis *et al.* [8] have demonstrated that, in the case of one-hop persistent cross-traffic (the worst case), the rate R_i of a train of packets traversing a link i with capacity C_i, does not depend on the train length but only on the utilization u_i of link i and on the train rate R_{i-1} at which the train left the previous link $i-1$. In particular, assuming a fluid model of cross-traffic and denoting with $A_i = C_i(1-u_i)$ the available bandwidth (avail-bw) of link i, the train rate after passing through link i will be

$$R_i = \begin{cases} C_i/(1 + u_i C_i/R_{i-1}) & \text{if } R_{i-1} \geq A_i \\ R_{i-1} & \text{otherwise} \end{cases} \quad (1)$$

Thus, if the link is not used ($u_i = 0$) and the train rate is sufficiently high ($R_{i-1} \geq C_i$), then $R_i = C_i$ and the train rate will represent the capacity of the link. If we extend this to all the links of a path, a train of packets sent back-to-back at the maximum speed possible will be received at the destination at rate $R_i = min\{C_i\}$,

[3]We use ACKs of 1488 Bytes because some ISPs set the MTU size to less than 1500 Bytes to account for protocols such as PPPoE or PPPoA. In DSLprobe the ACK size can be changed at will.

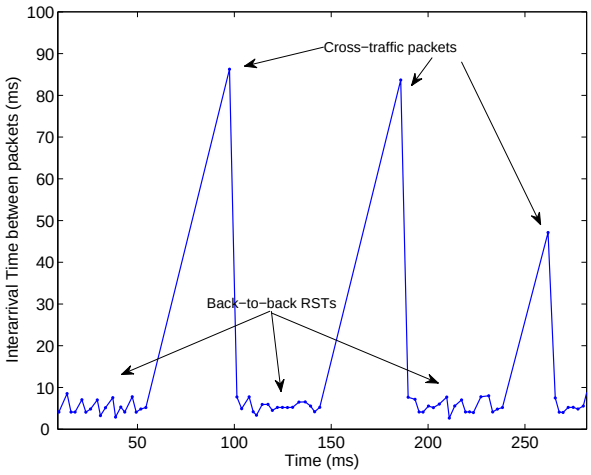

Figure 3: Uplink estimation (ACKs are 40B). IAT of the RSTs under *uplink* cross-traffic.

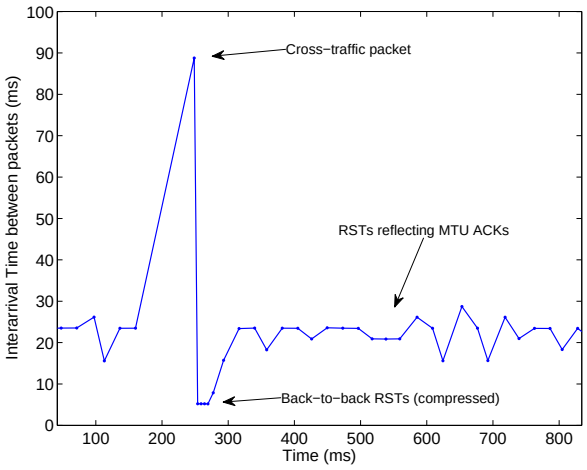

Figure 4: Downlink estimation (MTU ACKs). IAT of the RSTs under *uplink* cross-traffic.

which represents the maximum rate achievable on the link with lowest capacity. If the links are loaded, instead, the packets are dispersed at each hop and the train rate deviates from the capacity proportionally to the quantity of cross-traffic and to the train rate at the previous hop.

In order to obtain a good estimate of the bottleneck capacity, we thus need the cross-traffic error term $u_i C_i / R_{i-1}$ to be as small as possible. Now, suppose we send from a well connected host a train of packets at rate $R_0 = 100$ Mbps. Since u_i is *very small for all backbone links* [10], the dispersion will be very low and the rate of the train arriving to the ADSL downlink will be close to R_0. Such high rate implies that packets will arrive at the downlink ADSL buffer very close to each other, limiting the possibility for other packets to interfere with the train, which helps achieving a good estimate. For example, if the ADSL link has 5 Mbps downlink capacity, Eq. 1 guarantees that even in case of full link utilization ($u = 1$) the error term will be at most 5%. The idea is thus to use the average dispersion rate ADR to estimate the downlink capacity.

If we finally drop the fluid cross-traffic assumption, the effect of finite-sized cross-traffic packets is averaged along the train and has no real impact on the measurements, as the results will prove.

6.3 Effect of cross-traffic and train length

The design of DSLprobe is based on the observation that the backbone links have much higher capacity (and available bandwidth [10]) than the ADSL links. Thus, a train of packets sent at high rate from the measurement host will also arrive at high rate to the ADSL downlink buffer, suffering very little cross-traffic. However, the impact of cross-traffic on the ADSL *uplink* can be very destructive because of the low uplink capacity. Figs. 3 and 4 give simple examples of the impact of large uplink cross-traffic packets. These figures depict the IAT of the RSTs flowing back when we measure the uplink (fig. 3) or the downlink (fig. 4) and illustrate how the IATs are perturbed by cross-traffic packets.

Regarding the length of the trains, there is an important trade off to be considered: on one hand, the impact of a cross-traffic packet is less influent when adopting longer packet trains; on the other hand, the smaller the train, the lower the probability of having packets interfering. Moreover, long trains suffer higher chances of packet loss and are more intrusive. Similarly to the ADR estimation implemented in Pathload [7], in DSLprobe both uplink and downlink estimations are initially done using 50 ACKs. In case of loss, the measurement is discarded and the number of probes is reduced to 25, 10, and eventually 5 packets[4]. For a general discussion on the length of packet trains we refer the reader to the work in [8]. Finally, since the trains are very small, the risk of overloading network devices is negligible.

7. THE ALGORITHM

7.1 Estimating the uplink capacity

Since the capacity of the uplink is much lower than the capacity of the downlink, using "symmetrical" probes ($s_{A/R} = 1$) the dispersion caused by the uplink will be the highest even when the downlink is heavily con-

[4]As explained later, we might also shorten the train when measuring the downlink capacity under uplink cross-traffic. We prefer using shorter trains and repeating the measurement rather than trying to select usable sub-parts of the trains, which would complicate the analysis. Note that, in the worst case, the total amount of traffic generated is 90 packets, which amounts to less than 140 KByte.

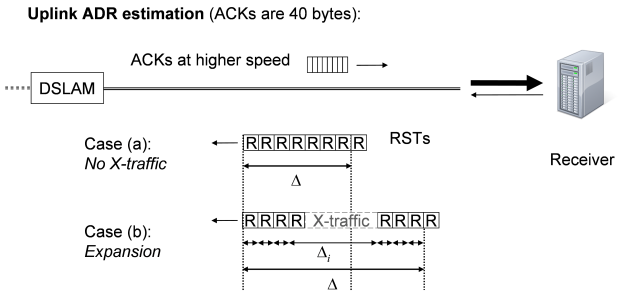

Figure 5: Impact of *uplink* cross-traffic on the *uplink* measurement. Using the IPIDs or analyzing the IAT, it is possible to filter out these packets and to correct the measurement.

gested[5]. It is thus possible to estimate the uplink capacity by sending ACK probes of 40 Bytes being careful to then correct the results to account for L2 overheads, as explained in Sec. 5.1.

Before arriving to the uplink, the probes traverse the downlink and the train rate is significantly reduced from R_0 to approximately C_{down}. The train duration will thus be longer and the chances to have cross-traffic packets interfering will be higher. For this reason, the uplink measurement can suffer from cross-traffic more than the downlink measurement. Large packets, in particular, can disperse the RST probes significantly causing an uplink capacity underestimation. Fig. 5 shows the expansion caused by a large uplink cross-traffic packet on the ADR measurement. Let Δ be the total dispersion of the train, i.e. the time between the arrival of first and the last RSTs, and Δ_i the IAT between RST packet i and RST packet $i+1$. Since the arrival times are known to us as we receive the RSTs, we can detect which of the IATs have been inflated by cross-traffic and the ADR can be easily corrected by subtracting Δ_i from Δ and decreasing the train length by 1.

We detect cross-traffic packets on the uplink in two ways: 1) by checking if the IPIDs of the RSTs are consecutive (see [1] for more details) and 2) verifying that the IAT is not 2 times larger than the median IAT of the train, i.e. $\Delta_i \leq 2\Delta_{med} = 2 \times median\{\Delta_i\}$. This is consistent with the fact that all packets interfering with the RSTs will increase the IAT by *at least* a factor of 2 because cross-traffic packets should be fragmented in at least 2 ATM cells, the same size as the RSTs. Note that we take the median IAT and not the minimum because of the effects in Fig. 1. We use this second test in case the IPIDs are not usable for cross-traffic detection which is the case, for example, when the measured host is running Linux. Algorithm 1 provides an overview of the uplink measurement implemented in DSLprobe.

[5]This is because packets dispersed on the downlink are rebuffered on the uplink or, more rigorously, because from Eq. 1 to obtain $R_{down} < A_{up}$ (and thus $R_{down} < C_{up}$) the ADSL must be almost symmetric with $c_{d/u} \approx 1$.

Algorithm 1 Uplink estimation algorithm

```
S_ACK = 40; //we need s_A/R = 1;
L = {50, 25, 10, 5}; //train lengths
for (j = 1; j ≤ 4; j = j + 1) do
  Send L[j] ACKs @ 100Mbps; receive RSTs;
  if (loss) then
    continue; //use shorter train
  else
    break;
  end if
end for
Measure dispersion Δ and IATs {Δ_i};
/* Remove IATs affected by cross-traffic */
if (usable IPIDs) then
  eliminate packets with non-consecutive IPID;
else
  Δ_med = median{Δ_i}; // median IAT
  eliminate packets with Δ_i > 2Δ_med;
end if
update Δ and L (without affected IATs);
R = (L − 1)S_ACK/Δ; //compute ADR
C_up = 2.65 × R; //adjust for L2 overheads
```

7.2 Estimating the downlink capacity

Measuring the downlink capacity is a difficult task for two reasons: first of all, because we must verify that the RSTs flowing back are not constrained by the uplink ($s_{A/R}$ sufficiently high); secondly, because after being dispersed on the downlink, packets must traverse the uplink without cross-traffic interference.

7.2.1 *Highly asymmetric links*

The first essential condition so that the RSTs flowing back are not altered by the uplink is that $s_{A/R} > c_{d/u}$. This can be checked either by (i) varying $s_{A/R}$ and measuring if the rate of the RSTs changes accordingly (the rate will not change if $s_{A/R}$ is too small) or, (ii) in case the uplink capacity is already measured, by verifying that the rate at which the RSTs are received is lower than the maximum uplink capacity. Since we do estimate the uplink capacity, we prefer this second option which is faster and does not require additional measurements. Let $T_{min} = S_{RST}/C_{up}$ be the minimum IAT between RSTs allowed by the uplink. We suspect the ADSL link to be too much asymmetric when the median IAT of the RSTs flowing back is less than $1.3 \times T_{min}$. In other words, if the RSTs flowing back start congesting the uplink (only less than 30% of the uplink capacity is left), we declare $s_{A/R}$ to be too small compared to $c_{d/u}$. A 30% margin should be enough to account for the noise introduced by other links on the path. In case this margin is not respected, we redo the measurement

alternating ACKs and UDP[6] packets so that only half of the probes generate a RST, thus doubling $s_{A/R}$, or even two UDP packets for every ACK (but generally $s_{A/R} = 32$ is sufficiently large for all ADSL links).

7.2.2 Uplink cross-traffic detection

The distortion caused by cross-traffic packets on the uplink is the most important and difficult problem to assess. The transmission of an MTU size packet on the uplink can cause a significative expansion (or compression) of the RSTs flowing back, biasing the capacity measurement. The interactions between RSTs and uplink cross-traffic can be complex and some examples are given in Fig. 6. The main results of these interactions are train *expansion* and train *compression*. Expansion occurs whenever the last packet(s) of the train are queued behind cross-traffic and the dispersion Δ of the train is inflated (case (b) in Fig. 6). Compression occurs when the queue is not empty at the arrival of the first RST packet so that the initial dispersion is reduced (case (c) in Fig. 6). It can also happen that the first and the last packets are unaffected even if the RSTs in the middle suffer from cross-traffic (case (d)). Let now N be the number of RSTs affected by a cross-traffic packet. The value of N depends on $s_{A/R}$, on the ratio $s_{X/R} = S_X/S_{RST}$ between the size of the cross packet S_X and the RSTs size, and on the ratio between the downlink and uplink capacity following the formula $N = s_{X/R} \times (C_{down}/s_{A/R})/(C_{up} - C_{down}/s_{A/R})$. For example, if $c_{d/u} = 6$ and both the cross-traffic packet and the ACKs are MTU size ($s_{A/R} = s_{X/R} = 16$), then about 10 out of the 50 packets of the train will be affected. This implies that already very few cross-traffic packets can completely disrupt the dispersion of the train, altering the results.

To precisely filter out the effects of cross-traffic, we must thus know both the place where the cross-traffic packet is located with respect to the RSTs and also the size of this packet. While detecting the presence of cross-traffic packets can be done with techniques based on IPIDs or IAT analysis (like the ones described for the uplink measurement), estimating the exact size of a cross-traffic packet becomes unfeasible. Note, however, that the ADR measurement is certainly correct if *all* the IATs of the packets are preserved (case (a) in Fig. 6), but it is also correct if the *total* dispersion of the train, between the first and the last packet, is not altered (see case (d)). This is much less restrictive because it requires only that the head and the tail of the train traverse the uplink unaffected. Note that the IPIDs do not allow to detect cross-traffic causing compression of the head of the train (i.e. packets already in queue just

[6]Many hosts are nowadays located behind a NAT. Since in our evaluation we target hosts running eMule, our ACKs and UDP probes are sent towards the ports opened by this application (we explain how in Sec. 8.2), significantly increasing the probability of the host being reached (many hosts use static port mapping).

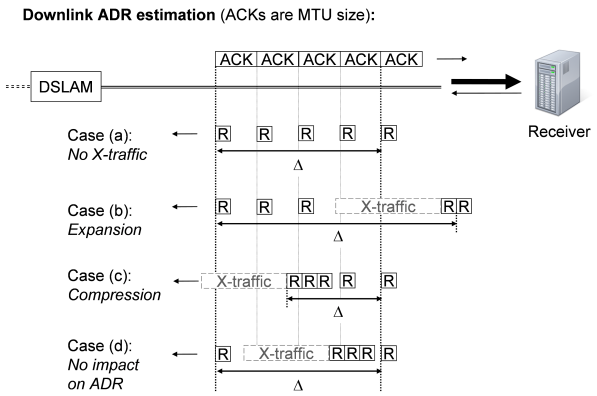

Figure 6: Impact of *uplink* cross-traffic on the *downlink* measurement. Cross-traffic packets can compress or expand the train duration.

before the train arrives, see case (c)) so an analysis of the IATs is necessary. Of great help in this case is the knowledge of the uplink capacity: RSTs received at the uplink capacity will be a clear sign of buffering (see the example in Fig. 4) with IAT equal to S_{RST}/C_{up} instead of $S_{ACK}/C_{down} = s_{A/R}S_{RST}/C_{down}$, which is higher if $s_{A/R}$ is correctly tuned. The same analysis can be done on the tail packets to check if cross-traffic has altered their spacing. So, if the IAT of the head or tail packets is too close to S_{RST}/C_{up} (compression) or if there is an IAT which is too high (expansion) we discard the measurement and we try again with a smaller train of packets[7].

We thus apply two test conditions on the 5 head and 5 tail packets of the train[8] to detect measurements affected by cross-traffic. We consider the ADR as valid if the following conditions are true. The first applies to both, the head and tail packets, the second condition on the tail packets only: C1) the median IAT is at least 30% higher than the capacity of the uplink and C2) the largest IAT is less than 3 times the median IAT. The first condition is intended to detect **compression** of the head or tail packets and states that packets must respect what was already verified for the entire train: the IAT between packets must be at least 30% higher than the minimum IAT allowed by the uplink. C1 alone detects most of the affected trains. C2 accounts for cross-traffic packets altering the very last packets of the train and thus impacting only few packets (the median IAT is unaffected).

Consider for example a cross-traffic packet entering

[7]The probability of having compression on the head packets depends on the arrival process (and not on the train length) but smaller trains have higher chances to traverse the uplink without interference from cross-traffic packets in the middle, thus reducing the probability of expansion.

[8]Theoretically, two packets should be sufficient to detect anomalies in the IAT but for sake of robustness we extend the IAT analysis to the first and last 10% of a train.

Algorithm 2 Downlink estimation algorithm

S_{ACK} = MTU; //thus $s_{A/R} = 16$;
$L = \{50, 25, 10, 5\}$; //train lengths
for $(j = 1; j \leq 4; j = j + 1)$ **do**
  Send train of $L[j]$ ACKs @ 100Mbps; receive RSTs;
  if (loss) **then**
    continue; //use shorter train
  end if
  Measure dispersion Δ and IATs $\{\Delta_i\}$;
  $\Delta_{med} = median\{\Delta_i,\ i{=}1, ..., L[j]\}$; //median IAT
  $\Delta_H = median\{\Delta_i,\ i{=}1, ..., 5\}$; // median of head
  $\Delta_T = median\{\Delta_i,\ i{=}L[j]{-}4, ..., L[j]\}$; //tail med.
  $\Delta_{maxT} = max\{\Delta_i,\ i{=}L[j]{-}4, ..., L[j]\}$; //tail max
  if $(\Delta_{med} < 1.3 \times C_{up})$ **then**
    /* not enough uplink capacity! */
    $s_{A/R} = 32$; //interleave ACKs and UDP probes
    repeat measurement;
  end if
  if $(\Delta_H < 1.3C_{up})$ or $(\Delta_T < 1.3C_{up})$ or $(\Delta_{maxT} > 3\Delta_{med})$ **then**
    /* bad measurement! */
    continue; // reduce train length
  else
    /* Δ is unaffected */
    $R = (L[j] - 1)S_{ACK}/\Delta$; // compute ADR
  end if
end for
$C_{down} = median\{R\}$; //return median of good ADRs

the train just in front of the last packet, which will cause no compression on the RSTs, but delay the last RST. This will inflate the train dispersion significantly while only one IAT is changed (the median will probably not change). C2 takes into account this expansion of the tail packets while still allowing some variation on the IAT (up to three times the median, cf. Fig. 1). Algorithm 2 provides an overview of the downlink measurement implemented in DSLprobe.

The tests described above are designed to detect the vast majority of cross-traffic, but interactions of smaller magnitude might still be present in the measurements. To be conservative and further improve accuracy, we repeat the measurement 10 times waiting 500ms between each measurement[9]. We then take the median estimated capacity – not the maximum because trains might suffer compression. Considering that for typical ADSL capacities a packet train takes about 100ms to go through, and adding another 100ms RTT, the total duration of these measurements is approximately 7 seconds. However, as we will show in the evaluation, fairly good results can be obtained also without repeating the measurements, so depending on the precision and the speed requirements of the user, one packet train can be enough and the measurement time is reduced to few hundred milliseconds (which is two orders of magnitude less than the MPI tool).

Finally, we must point out that the downlink estimation is not feasible if the uplink is severely congested: in fact, under this circumstance, the RSTs will always be buffered and the probability of having no queuing for both, the head and the tail packets, tends to zero. If R_{RST} represents the load generated by the RSTs on the uplink and A_{up} is its avail-bw, the downlink capacity can only be measured if $R_{RST} < A_{up}$. However, in our large-scale experiments DSLprobe provided at least one good measurement in over 75% of the cases. Note that a way to drastically reduce the number of RSTs on the uplink (thus reducing the probability of interference) would be to send ACKs only for the head and the tail packet while using UDP probes for all the rest of the train. Then, only two RSTs would be generated and the train dispersion is given by the IAT between the two RST packets. However, in this case we would loose *all* information on the rest of the train and we would not be able to detect compression or expansion with the techniques described above.

[9]In case of loss, some on going connections might have lost some packets too. When this happens, we pause for 3 additional seconds to let the connections recover.

8. EVALUATION

In this section we validate the accuracy of DSLprobe both "in-lab" on some ADSL hosts under our control and then towards 1244 hosts on the Internet. We also compare our tool against the MPI tool and provide a case study of two large ISPs providing ADSL services.

8.1 Validation on controlled hosts

We tested DSLprobe from a well-connected host towards ADSL hosts under our control. Tab. 1 shows the results obtained measuring one of these links. The capacity of the downlink is 4.14 Mbps while the capacity of the uplink was 0.62 Mbps. We tested DSLprobe with

X-traffic	Downlink			Uplink		
Downlink	C_{down}	success	CoV	C_{up}	success	CoV
none	4.14	10/10	0.6%	0.62	10/10	1.8%
1 Mbps	4.10	10/10	0.9%	0.62	10/10	2.2%
2 Mbps	4.07	10/10	1.3%	0.62	10/10	2.6%
3 Mbps	4.06	10/10	1.4%	0.62	10/10	2.9%
4.1 Mbps	4.05	10/10	2.7%	0.62	10/10	3.0%
Uplink	C_{down}	success	CoV	C_{up}	success	CoV
none	4.14	10/10	0.6%	0.62	10/10	1.8%
100 Kbps	4.14	10/10	3.2%	0.61	10/10	2.8%
150 Kbps	4.12	7/10	4.3%	0.62	10/10	3.4%
200 Kbps	4.09	6/10	5.7%	0.61	10/10	3.8%
300 Kbps	-	0/10	-	0.61	10/10	5.2%
400 Kbps	-	0/10	-	0.62	10/10	4.3%
610 Kbps	-	0/10	-	0.63	10/10	2.7%

Table 1: Cross-traffic impact. Median estimated downlink and uplink capacity (in Mbps), number of successful measurements and Coefficient of Variation (CoV) of the measurements.

different levels of downlink and uplink cross-traffic. We used $s_{A/R} = 16$ for the downlink measurements, thus almost 50% of the uplink capacity was consumed by the RSTs (S_{ACK} =MTU and $R_{RST} = 259$ kbps). The cross-traffic was UDP and the measuring host had a 100 Mbps Ethernet link connection with at least 90 Mbps avail-bw.

The top part of the Table shows the effect of downlink cross-traffic. We inject MTU size cross-traffic packets (the most harmful because they cause the highest delay) from a third well-connected host directed to the ADSL host. We then run ten independent uplink and downlink measurements and take the median value as the capacity estimate. In Tab. 1 we also include the number of successful measurements and the Coefficient of Variation (CoV) of the different measurements expressed in percentage as $CoV = 100 \times std\_deviation/mean$. As expected, the asymmetric properties of the ADSL allow the uplink to be measured correctly without suffering from downlink cross-traffic. For the downlink estimation, even with high congestion the underestimation caused by cross-traffic is minimal, less than 3%, confirming the theoretic results discussed in Sec. 6.2.

The lower half of Tab. 1 shows the impact of *uplink* cross-traffic on the measurements. Even when the uplink is completely saturated all the uplink estimations were successful and of excellent precision, proving the effectiveness of the cross-traffic filtering. The downlink estimation also gives very good results, provided there is enough avail-bw: as the cross-traffic rate on the uplink increases and approaches the limit condition where $A_{up} = R_{RST} = 260$ kbps, the probability of having the head and the tail of the RSTs free of interference decreases up to the point where all measurements suffer some compression or expansion (in our case from 300 kbps above). In this case, DSLprobe successfully detects compression and expansion of the RST and discards the measurements affected by cross-traffic.

Overall, with both downlink and uplink cross-traffic, the CoV between the ten different measurements is always very low, meaning that *each single packet train provides fairly good estimates*. Since the total duration for the uplink and downlink measurements is approximately 14s, one might prefer using only one measurement, with reduced precision, but obtaining a good estimate in few hundreds milliseconds (one RTT plus the transmission time of the train). Just to give an example, in the experiments above the maximum error for the downlink measurements in the worst conditions (with 200 kbps uplink cross-traffic) was less than 150 kbps or 3.6% of the downlink capacity.

8.2 Validation on Internet hosts

We have then tested DSLprobe towards 1244 ADSL hosts connected through various ISPs in the US and Europe. We used the MPI tool to establish a "ground truth", comparing the results obtained with the ones

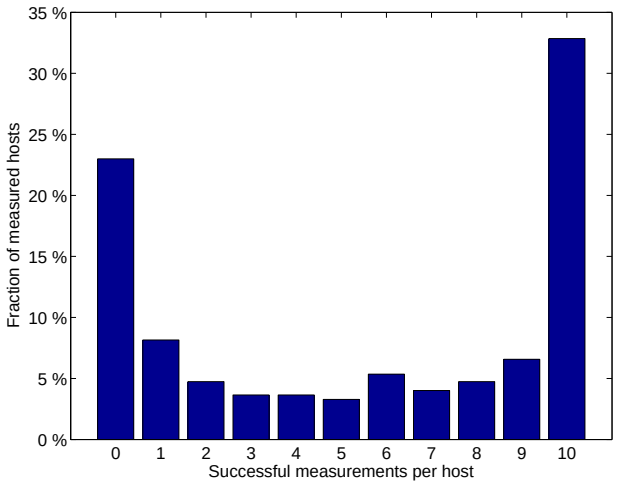

Figure 7: Number of successful *downlink* measurements (out of 10).

# of packets	unsucc.	5	10	25	50
Downlink	29.1%	17.8%	8.5%	9.8%	34.8%
Uplink	0.1%	1.1%	2.5%	7.5%	88.8%

Table 2: Train length of the measurements. Fraction over *all* measurements (10× # of hosts).

of DSLprobe. We used the KAD crawler of [21] to obtain IP addresses of hosts participating to the Kademlia DHT [17] and then used the Maxmind database [16] to discover the ISP associated to the IP addresses.

We selected 9 different ISPs providing ADSL services in Europe and US and measured both downlink and uplink ten times. Out of 1244 probed hosts, 822 were answering to our probes (66.1%). For the downlink capacity, on 499 hosts we obtained at least 6 out of 10 estimates free from cross-traffic. Fig. 7 shows the histogram of the number of trains not affected by uplink cross-traffic, while Tab. 2 summarizes the distribution of the train lengths. From the table, we see that for the uplink estimation about 90% of the measurements succeeded with trains of 50 packets. Since for the uplink we shorten the length of the train only in case of loss, this means that the vast majority of the trains suffer no loss at all. On the downlink, our measurements are successful in over 70% of the cases and we can also derive that: 1) almost 35% of the trains were not affected by cross-traffic or loss (50 packets); 2) in 10% of the cases there was little loss or light cross-traffic (25 packets); 3) using small trains (5 or 10 packets), another 25% of the measurements were successful; 4) 30% failed because of congested uplinks. The interesting thing, also combining these results with the histogram in Fig. 7, is that the downlink measurements are often always successful or not successful at all indicating that *on the uplink the ADSL is either unused or completely congested*.

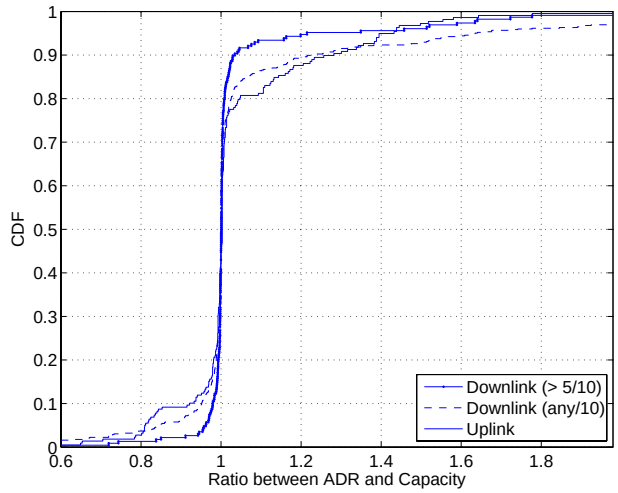

Figure 8: Accuracy of DSLprobe : comparison with the MPI tool. Results for the downlink improve about 10% by filtering out those hosts with less than 6 good measurements out of 10. Uplink measurements were always successful.

In order to verify the accuracy of DSLprobe, we measured the same hosts with the MPI tool *just after* DSLprobe. The MPI tool was successful against 504 of these same hosts and the number of hosts on which both tools produced results (the intersection) was 372.

Fig. 8 shows the ratio between the capacity obtained by DSLprobe and the estimate provided by the MPI tool. For both, uplink and downlink, 85% of the results deviate less than 20% from the capacity estimate obtained with the MPI tool. For the downlink, considering only those hosts with at least 6 out of 10 successful measurements, almost 90% of the downlink estimates fall within 5% of the capacity obtained with the MPI tool. The uplink estimation looks less accurate but in absolute terms the errors are quite small: with the precision showed in Fig. 8 the estimate of an uplink of 250 kbps will have with high probability an error of less than 40 kbps.

8.3 Discussion

While DSLprobe has been designed specifically to measure the capacity of ADSL links, the MPI tool is generic for ADSL and Cable networks and is intended to estimate various other link characteristics in addition to capacity. Designing fine-tuned tools for a specific environment rather than generic tools having broader applicability is a choice that has often to be taken. In the case of the MPI tool, the price to pay for this higher versatility is a higher impact on the broadband links, with persistent buffer saturation. Over 54,000 packets and about 30 MBytes of traffic are generated in only one minute, whereas DSLprobe estimates the capacity sending two orders of magnitude fewer packets even when being conservative and repeating the measurement 10 times.

As it has been shown in the evaluation, for almost 60% of the hosts probed, the MPI tool was not able to estimate the capacity (compared to 33% of DSLprobe). This was either because the ADSL was too asymmetric (the MPI tool can not measure links with $c_{d/u} > 16$) or because the host did not reply or stopped responding to the probes. Additionally some ISPs (France Telecom for example) have protections on the downlink buffer and start dropping the packets when there is a long term saturation, which causes the MPI tool to fail. With DSLprobe we aim to obtain precise capacity measurements without saturating the buffers. Cross-traffic can disturb or prevent us from measuring the capacity, but in the end DSLprobe obtains at least one good measurement in two thirds of the hosts tested and at least 6 good measurements in 40% of the cases, the same success rate as the MPI tool. Sending even more trains will obviously increase the chances of success. Congested links might still be difficult to measure but we prefer renouncing to measure these hosts rather than becoming intrusive.

8.4 Case study of two ISPs

We also compared the policies of two large European ISPs, namely Free (France) and Telefonica (Spain). The results are based on 352 different hosts, measured with our tool. Fig. 9 shows the downlink capacity estimated by DSLprobe. The two ISPs have different policies: like most other ISPs, Telefonica allocates fixed downlink rates matching the advertised offers, while Free allocates no fixed bandwidth (as much bandwidth as the local loop allows). This is quite different from the results given in [6] where most ISPs adopt the same policy as Telefonica. In Fig. 10, the results are similar for the uplink except that over 30% of the hosts belonging to Free have uplink capacity limited to 1 Mbps.

9. CONCLUSIONS

In this paper, we presented DSLprobe, a new single-ended tool for measuring the capacity of ADSL links without the active cooperation of the remote host. The probes are composed of TCP ACKs sent without an established connection in order to solicit TCP RSTs from the receiving host back to the source. In order to measure both, downlink and uplink capacities of the ADSL, DSLprobe takes advantage of the relatively low absolute bandwidth and the bandwidth asymmetry typical of ADSL. We study how to adjust the probes size, carefully taking into account the amount of overhead imposed by the L2 protocols.

To obtain precise measurements, DSLprobe exploits the much higher capacity of the backbone links to significantly reduce the probability of having interfering cross-traffic packets. To detect or filter the cross-traffic packets still interfering with the probes, we implement

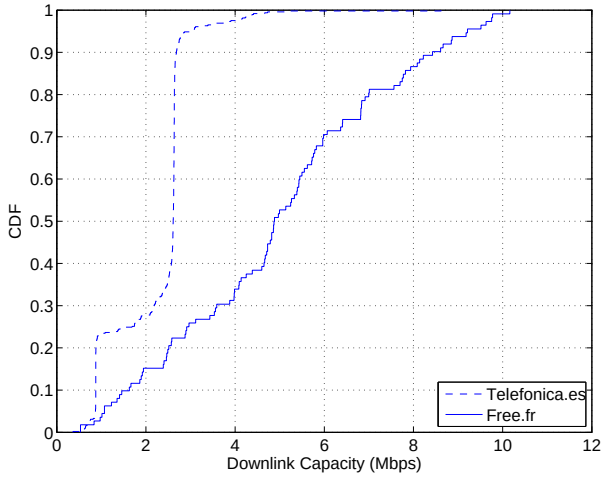

Figure 9: Comparison between the downlink capacity allocated by two ISPs.

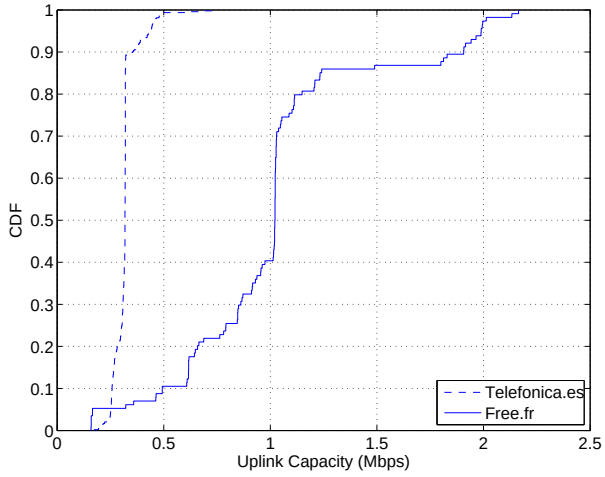

Figure 10: Comparison between the uplink capacity allocated by two ISPs.

several strategies based on IPIDs and the analysis of the IAT of the RSTs. Compared to current tools such as the MPI tool, DSLprobe sends two order of magnitude fewer packets, which significantly reduces the intrusiveness of the measurements. Finally, we demonstrated the accuracy of DSLprobe both on controlled hosts and on the Internet and provided a case study of two large ADSL providers.

Acknowledgments

We are very thankful to Marcel Dischinger for providing us the source code of the MPI tool, which greatly simplified the evaluation of DSLprobe. We also thank Moritz Steiner for providing the IP addresses of ADSL hosts.

10. REFERENCES

[1] J. Bellardo and S. Savage. Measuring packet reordering. In *IMW '02: 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 97–105, New York, USA, 2002.
[2] S. Bellovin. A best-case network performance model, 1992.
[3] A. Broido, R. King, E. Nemeth, and K. Claffy. Radon spectroscopy of inter-packet delay. *IEEE High-Speed Networking Workshop*, 2003.
[4] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. Biersack. Non-cooperative Available Bandwidth Estimation towards ADSL links. In *Proc. Global Internet Symposium 2008*, Apr. 2008.
[5] M. Crovella and R. Carter. Dynamic server selection using bandwidth probing in wide-area networks. In *Proc. of IEEE INFOCOM*, 1997.
[6] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proc. Internet Measurement Conference (IMC'07)*, Oct. 2007.
[7] C. Dovrolis and M. Jain. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput. In *ACM SIGCOMM*, Pittsburgh, USA, Aug. 2002.
[8] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Trans. Netw.*, 12(6):963–977, 2004.
[9] A. Downey. Using pathchar to estimate link characteristics. *In Proc. of SIGCOMM'99*, 1999.
[10] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-level traffic measurement from the Sprint IP backbone. *IEEE Network Magazine*, Nov. 2003.
[11] A. Haeberlen, M. Dischinger, K. P. Gummadi, and S. Saroiu. Monarch: A tool to emulate transport protocol flows over the internet at large. In *Proc. Internet Measurement Conference (IMC'06)*, Oct 2006.
[12] V. Jacobson. Pathchar: A tool to infer charcteristics of internet paths. *ftp://ftp.ee.lbl.gov/pathchar/*, 1997.
[13] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi. CapProbe: A simple and accurate capacity estimation technique. In *Proceedings ACM SIGCOMM*, 2004.
[14] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proc. of USENIX*, 1999.
[15] B. A. Mah. pchar: A tool for measuring internet paths characteristics. 2000. *http://www.employees.org/bmah/Software/pchar/*
[16] Maxmind. *http://www.maxmind.com/*
[17] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-peer informatiion system based on the XOR metric. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 53–65, Mar. 2002.
[18] OECD. Oecd broadband statistics, June 2007. *http://www.oecd.org/sti/ict/broadband*
[19] R. Prasad, C. Dovrolis, and B. Mah. The effect of layer-2 store-and-forward devices on per-hop capacity estimation. *INFOCOM 2003.*
[20] S. Saroiu, P. K. Gummadi, and S. D. Gribble. Sprobe: A fast technique for measuring bottleneck bandwidth in uncooperative environments. *http://sprobe.cs.washinton.edu*, 2002.
[21] M. Steiner, T. En-Najjary, and E. W. Biersack. A Global View of KAD. In *Proc. of the Internet Measurement Conference (IMC)*, 2007.
[22] D. Van Aken. Encapsulation overhead(s) in ADSL access networks, Dec. 2000. *www.thomsontelecompartner.com/getfile.php?id=525*
[23] L. Windsor Oaks Group. Annual market outlook report, March 2006. *www.broadbandtrends.com/Report%20Summary/2006/BBT%20GlobalBBOutlook2006%20061110%20TOC.pdf*