Study of a non intrusive and accurate method for measuring the end-to-end useful bandwidth

Mathieu Goutelle, Primet, Pascale Vicat-Blanc

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON n° 5668

Study of a non intrusive and accurate method for measuring the end-to-end useful bandwidth

Mathieu Goutelle and Pascale Primet
(LIP, INRIA RESO)

october 2003

Research Report N° 2003-48

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr

Study of a non intrusive and accurate method for measuring the end-to-end useful bandwidth

Mathieu Goutelle and Pascale Primet (LIP, INRIA RESO)

october 2003

Abstract

Studies and tools development for applications sensitive to data rates is a very active research field for distributed application performance optimization. The research community works to propose tools for measuring the end-to-end performance of a link between two hosts. Delay measurements provide a first approximation but aren't sufficient enough because the delay isn't a relevant metric. A bandwidth evaluation method would give a more realistic view of the raw capacity but also of the dynamic behaviour of the interconnection, when we want to evaluate the transfer time of an amount of data.
Among all the existing methods, there are some differences according to the measurements strategies and the evaluated metric. We first describe the available bandwidth measurements and then the total capacity measurements approaches. Among all the presented methods, none of them can evaluate both metrics, while giving an overview of the link topology. By using a hop-by-hop packet pair method, we show that we can provide such informations with a fine analysis of the measurements.
In this report, we detail our proposition of a solution for an hop-by-hop measurement of the capacity and available bandwidth. This method has been validated in simulation, then implemented in Linux and validated experimentally. We compare this method with others to define its limits and the future utilisations on the newly developed tool.

Keywords: IP network measurements, capacity evaluation, available bandwidth, *Packet Pair* method, `tracerate`.

Résumé

Dans un contexte de réseaux longue distance (Internet par exemple), de nombreuses recherches sont menées pour proposer des outils de mesure de la performance d'une liaison entre deux extrémités. Les mesures de délai fournissent une première approche mais sont insuffisantes car le délai n'est pas une métrique suffisamment pertinente. Une méthode d'évaluation du débit d'une liaison donnerait une vision plus réaliste de l'interconnexion, lorsque l'on cherche à évaluer le coût de transfert d'une quantité de données.
Parmi les solutions existantes, on peut distinguer les méthodes selon le type de mesure et le paramètre évalué. Nous nous intéressons d'abord aux mesures de débit disponible, puis aux méthodes d'évaluation de la capacité d'un chemin. Cependant, parmi ces solutions, aucune ne permet à la fois d'évaluer la *capacité* et le *débit* disponible en donnant un aperçu de la topologie. En adoptant une méthode de découverte saut par saut de la capacité grâce à la technique *Packet Pair*, nous montrons qu'il est possible de fournir de telles informations, grâce à une analyse fine de la distribution des mesures.
Dans ce document, nous détaillons notre proposition d'une nouvelle solution pour la mesure, segment par segment, de la capacité et du débit disponible d'un chemin. Cette méthode a été validée en simulation, puis implémentée dans Linux et évaluée expérimentalement. Nous confrontons finalement cette méthode à celles étudiées précédemment pour définir ses limites et les perspectives d'utilisation du nouvel outil développé.

Mots-clés: Métrologie des réseaux IP, mesure de capacité, mesure de débit disponible, méthode *Packet Pair*, `tracerate`.

1 Introduction

1.1 Presentation and context

In the context of Internet-like network, *i.e.* using IP at network level, the informations about network performances are very sparse. Since there is no control channel, the simplicity of this network layer drive to develop complex mechanism with an high participation of end hosts.

The scientific community has been working hard to provide external means for evaluating the end-to-end network performance. A lot is using the delay between two hosts to evaluate the *network distance* between two hosts [FJJ+01, NZ02]. This metric is very easy and quick to measure, but insufficient when you want to know the time of a data transfer. The capacity or available bandwidth evaluation along a path gives a far more accurate vision of the interconnection.

There is a lot of applications for such a service: *e.g.* rate adaptation of a data source, verification of a service guaranteed by a SLA, data source selection in multi-site applications, parameters tuning of transport protocol or network monitoring. It can be used also in the monitoring of an overlay network on top of IP networks.

1.2 Problematic

Let us take as an example a computer grid, architecture for sharing computing and storage resources through Local and Wide Area Network (LAN & WAN) interconnection (typically over IP). In this context, a better knowledge of the network is a major issue. The distributed computing specialists need indeed to evaluate the communication costs to propose efficient scheduling mechanisms in a computer grid. This cost can be rather easily evaluated in a super computer (internal communications) or in a cluster (dedicated network). But in a shared WAN, it is often unknown and subject to a fast and completely random dynamic, because it depends on the whole streams crossing the network. To know a way to evaluate the network perfomances at time t to build a kind of *cost function* of a data transfer is important. By the way, a transport protocol or a network administrator may need to localize the link bottleneck and evaluate its capacity and load.

There are some techniques being developed. They are completely suitable for measuring end-to-end bandwidth. They can be very simple or based on more theoretical principles, depending on the evaluated metric. There are indeed some different concepts beyond the word "bandwidth". We will try to dispel any ambiguity about this. Nethertheless, none of this solutions can determine in a non intrusive manner the bottleneck of a path and its localization, in a high-performance environment.

This document is organized as follow: we first see in section 2.1 how we can know the available bandwidth between two hosts. We then study (section 2.2) the methods for evaluating the capacity of a path. From this state of the art, we developed in section 3 a proposition of a new methodology providing a measurement tool of both capacity and utilization rate of each edges of a complex WAN path between two machines. The section 4 is dedicated to its implementation and its validation with simulations. Finally, we conclude (section 5) with a critical study of our tool, named `tracerate`.

1.3 Definitions

A *link* is an arc between two adjacent network equipements. A *path* is then a serie of links between two distant network equipements. The hop n of a path between two nodes is the subset of links from the first node to the n^{th} node of the path.

Before describing bandwidth measurements in a network, we need to give a more precise definition of some concepts behind this expression. There is indeed a lot of different metrics covering the word "bandwidth" [LTC+03]. These quantities are homogeneous to an amount of data per unit of time. We distinguish four distinct metrics here:

The capacity ($\mathscr{C}$) It is the maximal rate a link or a path can provide, without any cross-traffic. It depends mainly on the underlying link technology. It is typically a static property of a

link, but cannot be applied to a path because of routing dynamic ;

Utilization ($\mathscr{U}$) It represents the whole current traffic on a link or a path. This property is dynamic. We note $u = \mathscr{U}/\mathscr{C}$ the utilization rate ;

The available bandwidth ($\mathscr{A}$) Given a particular utilization and capacity on a link, $\mathscr{A}$ is the maximal extra amount of data per unit of time which can go through that link. We can deduce this metric from the two previous one ($\mathscr{A} = \mathscr{C} - \mathscr{U}$) and *vice-versa*. It keeps the dynamic characteristic of $\mathscr{U}$;

The achievable bandwidth ($\mathscr{A}'$) This notion slightly differs from the previous one because it considers the maximal extra rate at an upper level (application level), according to the value of $\mathscr{A}$, $\mathscr{C}$ and characteristics of end hosts (operating systems, CPU load, type of application, *etc.*) and transports protocols used.

We must also highlight that all these definitions are valid for both a link and the whole path. In the path case, the value is often the minimum or maximum (depending of the considered parameter) on the whole set of links of the path. On the other hand, the value of these metrics increases or decreases each time you go through the layers of TCP/IP architectural model. This comportment is caused by the encapsulation, the size of the headers and the overhead of the layer crossing. It is then important to always precise the layer from which the value is measured.

2 State of the art

2.1 Available bandwidth measurements

A lot of methods exists to determine the available bandwidth along a path between two nodes of a network. We classify them into *intrusive* and *non intrusive* methods. It is somehow clear that non intrusive methods are although intrusive, since there is no control channel in IP networks and since measurements are conducted in-band. The intrusiveness of a method is evaluated on its capacity of producing a result while sending the minimum amount of extra traffic. On the other hand, we exclude from our scope *passive* methods, using SNMP counters of networks equipments (*e.g.* packet counters in a router), since they only produce a local and not end-to-end measurement.

2.1.1 Intrusive measurements

Intrusive measurements of available bandwidth consist in initiating a relatively long transfer between an emitter and a receiver and measuring the obtained rate. With this, you can determine the performance a user can get during a tranfer. This measurements are doubtful because they create a non negligible traffic and hurt every other connection sharing a link with the probed path.

Therefore, this method is widely used, *e.g.* in the grid projects, to monitor the interconnections quality (delay, rate and loss) between grid nodes for many reasons. First they give relatively accurate results and take into account various parameters: load on end-hosts, transport protocols dynamic (*e.g.* TCP), quality of service, *etc.*, in addition to the link utilization. These dependencies become a drawback because all these parameters have a fast evolution dynamic and so the validity of a measurement is very short. Because of the impact of a measurement on the network, it can't be too often repeated. For the same reason, such measurements can't be done in a high bandwidth network, since you must be able to congest the measured link and this is far from easy with bandwidth greater than gigabit/s. Generally, 10 s of saturating traffic must be generated, *i.e.* 10 Gbit of data sent.

To conclude on this class of methods, we can cite some of the tools used to perform this kind of measure. The most widely used is certainly `iperf`[1]: it allows to tweak various parameters of the protocols and various options for the measurement (protocol, parallel stream, *etc.*). There is

[1] http://dast.nlanr.net/Projects/Iperf/

also the standard Unix™ and Linux tool called `ttcp`[2] and the `netperf`[3] tool, developped by HP. Let us cite the NWS software which can provide satisfactory rate informations between two points when the link is medium-size (around 100 Mbits/s) and relatively loaded [PHB02].

2.1.2 Non intrusive measurements

The *non instrusive* methods try to limit as much as possible the amount of data sent in the network. They often need an analytic model of the network and use a clever analysis of measurements and network behaviour during the probing.

One of this methods is the one developed by the `pathload` tool [JD02]. The principle is to send, from one machine to another one, a packet train at a given rate $\mathscr{D}$. If the rate $\mathscr{D}$ is greater than the available bandwidth $\mathscr{A}$, the inter-packet delay (or *jitter*) increases due to the bottleneck. On the contrary, if $\mathscr{D}$ is lesser than $\mathscr{A}$, the jitter remains around zero during the whole measurement. If $\mathscr{D}$ is around $\mathscr{A}$ (a little greater), the jitter will remain around zero while the bottleneck queues are not build and will then grow. The `pathload` methodology is to inject traffic in the network, to detect the increasing or non-increasing trend in the jitter and to adjust the emission rate for the next measurement according to this. The rate adjustement use a dichotomy like method, until the interval is sufficiently narrow to provide a correct estimation.

According to [JD02], `pathload` gives good results, with a relatively strong correlation with passive measurements done with MRTG. The authors have shown that the intrusivity of `pathload` is low: a measure doesn't affect the link characteristics (delay and TCP accessible rate). Although it sends traffic at a greater rate than available bandwidth, it seems that sending a train of 100 packets (about 20 ko of data) suffices in order not to increase too much the queue length along the path.

The first criticism about `pathload` is that it needs a cooperation from both the emitter and the receiver (software installation). On the other hand, even if it is said to be non intrusive, this affirmation is based on the observation of TCP behaviour (obtained rate) and delay at a coarse grain [JD02]. At the IP level, a measurement really load the network to its congestion point during a short period of time, by principle and so increase loss and delay. This is partly the reason why `pathload` is limited to low rate measurements (around 120 Mbits/s) and cannot be used in a high performance network context.

2.2 Capacity measurements

The link capacity is the maximum rate provided by the underlying technology of the link (link layer). We define the capacity of a path as the minimum capacity of the edges. Unlike the available rate evaluation, capacity measurements use more complex mechanisms. You need indeed, in order to have an access to this link characteristic and evaluate it, to have a model of the network behaviour and to make a subtle observation of the measures you can do [CM01]. The methods are so relatively non-intrusive intrinsically.

In the following, we need to develop a model of the studied path at the network level (IP layer). This model is described on figure 1. From now we assume the following notations:

- $(\mathscr{H}_i)_{0\leq i\leq N}$ the set of network equipments of the studied path. $\mathscr{H}_0$ and $\mathscr{H}_N$ are the ends of the path ;
- d_i is the processing time of a packet at node i on the way (i increasing). We consider d_i can be function of time. d'_i is the processing time of a packet at node i on the way back (i decreasing) ;
- q_i is the queuing delay of the packet in the output queue of node i on the way. q'_i represents the same delay but on the way back. We choose $q'_0 = 0$;
- l_i and c_i are respectively the latency and capacity of link i ;

[2] http://ftp.arl.mil/~mike/ttcp.html
[3] http://www.netperf.org/

- τ_i is the time an host take to emit physically a packet on link i, aka *transmission time*. We have the relation: $\tau_i = {}^{s}/_{c_i}$, for a packet of size s.

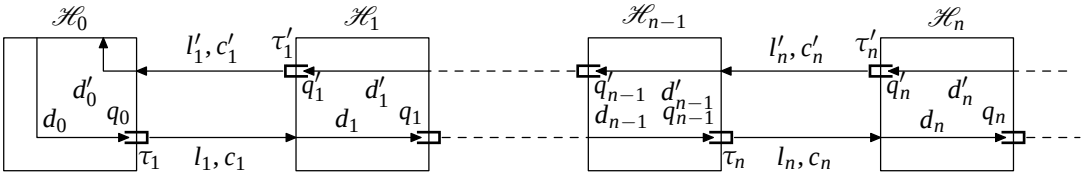

Figure 1: Model of a multi-hop IP path

We highlight here three wide classes of capacity evaluation method : the *Variable Packet Size* method, the *Packet Tailgating* method and the *Packet Pair* method. They are illustrated by three tools, significant representants of each class : `pathchar`, `tailgater` and `pathrate`. Other tools are implementation variations or data analysis improvements.

2.2.1 *Variable Packet Size* method

The `pathchar`[4] tool was developed by Van Jacobson in 1997 [Jac97]. It reuses the technical principle of `traceroute` from the same author, *i.e.* the utilization of ICMP messages for probing the path and getting some cooperation of the network equipments. It allows to evaluate the delay, queue length and capacity of each link of the path and so to identify the bottleneck. Since 1997, the method keeps evolving with other implementations such as `clink` or `pchar`, improving the duration, the efficiency or the intrusivity of a measurement.

Principle Like `traceroute`, `pathchar` use the TTL field of the IP header. This field is set by the emitter and represent the maximal number of hops a packet can go through. If an equipment receives a packet with null TTL value, it sends a *Time Exceeded* ICMP message, containing the header of the rejected packet, back to the source. Otherwise, it decrements the TTL field before sending the packet. For the last hop (the receiver), the return is obtained using *Port Unreachable* ICMP message, since the host rejects the emitted packets. With this functionality, `pathchar` is able to emit packets to an host which sends them back. Using increasing values of TTL, you can do measurements on each loop between the emitter and any equipment of the path (cf. figure 2).

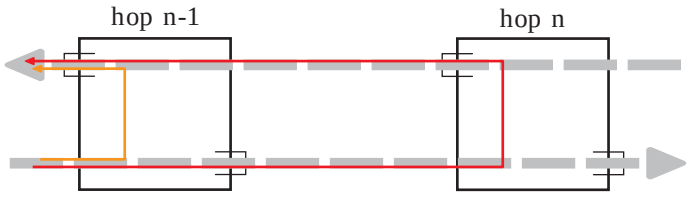

Figure 2: Illustration of the utilization of the TTL field. With increasing values of TTL, it is possible to probe incrementally a path between two hosts.

Let the TTL be equal to n and let us express the round-trip time $\mathscr{R}_n$ between an emitter $\mathscr{H}_0$ and an intermediate host $\mathscr{H}_n$. We note σ_n the extra processing time of the packet to emit the ICMP message. The others notations have been previously introduced. It comes:

$$\mathscr{R}_n = \sigma_n + \sum_{k=1}^{n} (d_{k-1} + q_{k-1}) + (\tau_k + l_k) + (d'_{k-1} + q'_{k-1}) + (\tau'_k + l'_k) + q'_n \qquad (1)$$

We assume from now on that :

[4] `pathchar` is the acronym of *path characteristics*.

1. if the size of ICMP messages is small compared with the size of the sent packets, τ'_k can be neglected ;
2. the packets don't wait in router queues ($q_k \approx 0$ and $q'_k \approx 0$). This is the case in a low load network during the measurement ;
3. the creation of the ICMP message in $\mathscr{H}_n$ is very quick. σ_n is therefore negligible.

The equation (1) becomes:

$$\mathscr{R}_n = \sum_{k=1}^{n} d_{k-1} + l_k + d'_{k-1} + l'_k + \tau_k \tag{2}$$

On reminding that, if the packet size is s, we have $\tau_k = {}^{s}/_{c_k}$ and noting $\mathscr{L}_k = d_{k-1} + l_k + d'_{k-1} + l'_k$ the total delay of $\mathscr{H}_{k-1}$ and the link k, it comes so:

$$\mathscr{R}_n - \mathscr{R}_{n-1} = \mathscr{L}_n + \frac{s}{c_n} \tag{3}$$

Incrementally, after having measured the RTT at step $n-1$ and on doing many measurements with various packet sizes[5], we can deduce the values of $\mathscr{L}_n$ et c_n with a simple linear regression.

Analysis One difficulty is to verify the validity of the second hypothesis (empty queues) during the measurement. For this, `pathrate` does many measurements for a given packet size and keep only the minimum observed delay: the packet with this delay has probably suffer from no queuing delay. On figure 3, we observe the round-trip time versus packet size during a measurement, on the left without filtering and on the right with minimum filtering. Keeping only the minimum RTT value is an error source because it doesn't eliminate measurement noise.

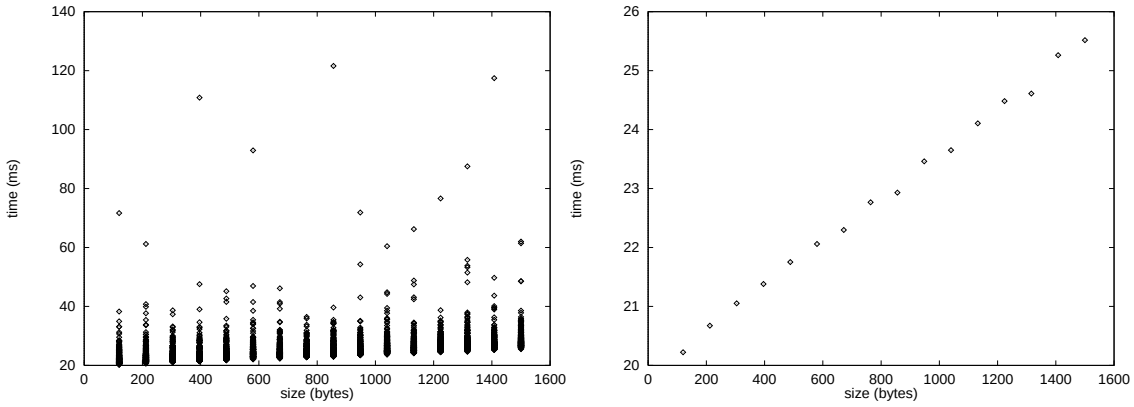

Figure 3: Example of `pathchar` measurements. This graphs present the round-trip time $\mathscr{R}_n$ versus packet size. On the left, without filtering and on the right with minimum filtering. Figures extracted from [Jac97].

Beside being sensible to noise, some non-linearities can appear in the right curve of figure 3, because for example of specific policies applied to small packets or complex queuing disciplines. On the other hand, it is impossible for this method to detect an IP link build on top of many agregated physical channels. In that situation, `pathchar` is able to measure, in the best case, only the capacity of one of the channels if all packets go through the same channel or if all channels have the same capacity (*e.g.* ATM links). If packets go randomly through from a channel to an other of a different capacity, a non-linearity will appear. It is impossible too to handle correctly invisible hops [PDM03], for example level 2 or lower devices (hubs, switches, *etc.*).

[5] Here appears the explanation for the name of the method: *Variable Packet Size*.

2.2.2 *Packet Tailgating* method

Principle The `tailgater` tool use an original method, based on hypothesis slightly different from other methods. With a study of the relations between the network delay of a packet pair and the path characteristics (cf. section 2.2), the authors of [LB00] show that the capacity c_g of the path bottleneck follows the equation (4), on noting:

s^k the size of the packet k in the pair ($k \in \{1, 2\}$) ;
t_n^k the arrival time of packet k in the node n. By convention, t_0^k is the departure time of the packet k from the first node ;
L^k the cumulative latency of the links up to k: $L^k = \sum_{i=1}^{k} l_i$;
C^k defined by $1/C^k = \sum_{i=1}^{k} 1/c_i$.

$$c_g = \frac{s^1}{t_N^2 + \frac{(s^2 - s^1)}{C^{g-1}} - \frac{s^2}{C^{N-1}} - t_0^1 - L^{N-1}} \tag{4}$$

This formula supposes that the packets don't suffer from any queue waiting time, except for the second packet at the bottleneck.

Analysis To insure the validity of this last hypothesis, `tailgater` implements a method which consists in identifying first the value of the quantities depending of the link (C^{N-1} and L^{N-1}) by using a method similar to `pathchar` (see section 2.2.1) with an optimisation proposed by [Dow99].

Once this values and the location of the bottleneck are known (hop g) and to validate the hypothesis of no queue waiting time except at the bottleneck, `tailgater` use the following method[6]: to probe the link g on the path, it sends a packet of maximal size with a TTL equal to g immediately followed by a minimal size packet with a maximal value of TTL (255). Like this, the small packet will be always just behind the big one, because of its smaller transmission time. At node g, the first packet will be dropped because of the TTL and the second one will continue on its way without waiting in a following queue. On the other hand, it's difficult to measure a one-way delay for the second packet (problem of synchronization of the two hosts). To overcome this, `tailgater` measures a RTT and considers that the receiver will send the packet back immediately and that the links are symmetric. So, there is only extra terms in L^k and C^k because of the way back.

The major interest of this method is that it sends less packets to do a measurement. It is so potentially less intrusive and faster than other methods. It allows to detect a link constituted of many physical links too. One of the limitations of this tool is the incremental discovery of the path characteristics and therefore the error accumulation while the TTL increases. It is more or less clear that measuring long paths is problematic for this tool.

2.2.3 *Packet Pair* method

An other solution of capacity evaluation is to send a packet pair with an known delay between the two probes and then analyze the distribution of these delays at the receiver. This notion is quite old and has been first introduced by Van Jacobson in [Jac88]. It has been used after in [Bol93] to study the structure and evolution of the network load in the Internet. It has been also studied in details in Vern Paxson's thesis [Pax97]. As a capacity evaluation method, the first works are more recent and have been implemented particularly in the `pathrate` tool [DRM01].

Principle The principle is quite simple (see figure 4): if two back-to-back packets are sent in the network, the inter-packet delay (or *dispersion*) at the receiver is the result of the going through of the path bottleneck.

Without any concurrent traffic on the whole path, we can model the method, with Δ_i the dispersion up to the node i and $\Delta_0 = 0$. It comes, with the notations of § 2.2, the following recursive definition:

$$\Delta_i = \max(\Delta_{i-1}, \tau_i) \tag{5}$$

[6] This particularity gives the tool its name.

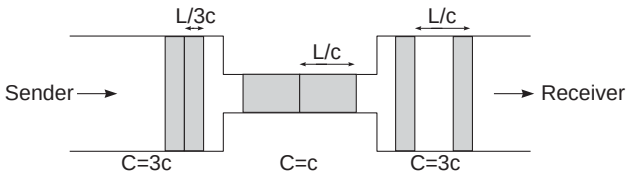

Figure 4: Illustration of the effect of a bottleneck on a packet pair. L is the packet size and C is the link capacity. The effect of the bottleneck is theoretically preserved along the path.

At the receiver, the measure of Δ_N provide us with the maximal value of τ along the path: $\Delta_N = \max(\tau_1, \ldots, \tau_N)$, *i.e.* for the link with the smallest capacity. Unfortunately, this is only valid if there is no cross-traffic. To obtain a model for the situation with cross-traffic (cf. figure 5 et [DRM01]), we need to take into account the waiting-time in the equipments. It comes then the equation (6) where the power index (1 or 2) indicates the first or the second packet of the pair.

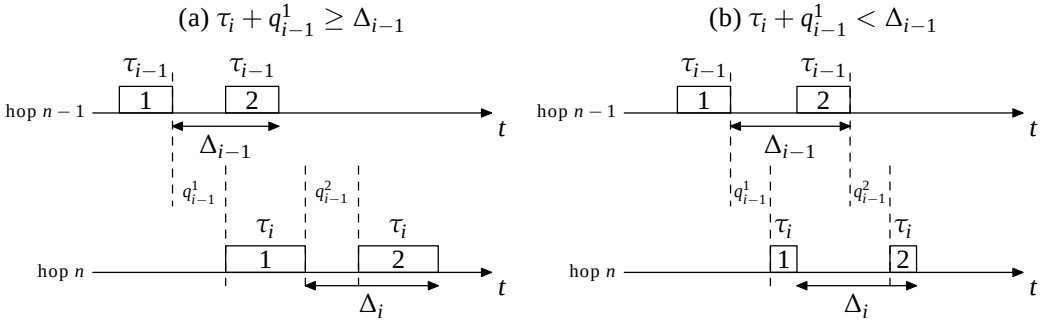

Figure 5: Effect of cross-traffic on a dispersion measurement according to the two cases of equation (6). From a figure taken from [DRM01].

$$\Delta_i = \begin{cases} \tau_i + q^2_{i-1} & \text{(a) if } \tau_i + q^1_{i-1} \geq \Delta_{i-1} \\ \Delta_{i-1} + q^2_{i-1} - q^1_{i-1} & \text{(b) otherwise} \end{cases} \tag{6}$$

It can be interpreted this way: the equation (6a) corresponds to the case when the packet pair manages to overload the capacity of the link i (decrease of the capacity). The measure gives also the value of τ_i tainted with an error due to cross-traffic: if a packet inserts itself between the two probes, the value of the bottleneck capacity may be underestimated. The alternativ (6b) is the opposite case, when the capacity increases (normally after a local minimum of the capacity). The dispersion is in that case propagated along the path. Unfortnately, if the sum of the two last terms is negative, the dispersion decreases and the information about the path capacity is lost: the capacity may be overestimated. It is the case when the first packet waits in a queue more than the second one due to cross-traffic (non empty queue at arrival time of the first packet). The extreme case is when the two probes are again back-to-back at the output of the equipment. In the same way, if the sum of the two last terms is positive (cross-traffic packet inserted between the two probes), the dispersion increases and the capacity may be underestimated. Going through all the path links and the presence of cross-traffic is hence critical since the information can be lost and the capacity of the bottleneck can be under- or overestimated.

Finally, if we repeat the experiment several times, we obtain a *multimodal* distribution of values (see figure 6): the capacity mode is not the only maximum and may only be a local maximum. On one hand, the cross-traffic creates modes lower than the capacity mode (noted *Sub-Capacity Dispersion Range* in [DRM01]), typically when packets insert themselves between the two probes and increase artificially the dispersion. We found also modes with greater values than the capacity mode (noted *Post-Narrow Capacity Mode* in [DRM01]) if the probes have been in the equation (6b) case with a negative error term: the first packet has wait in an equipment after the bottleneck and

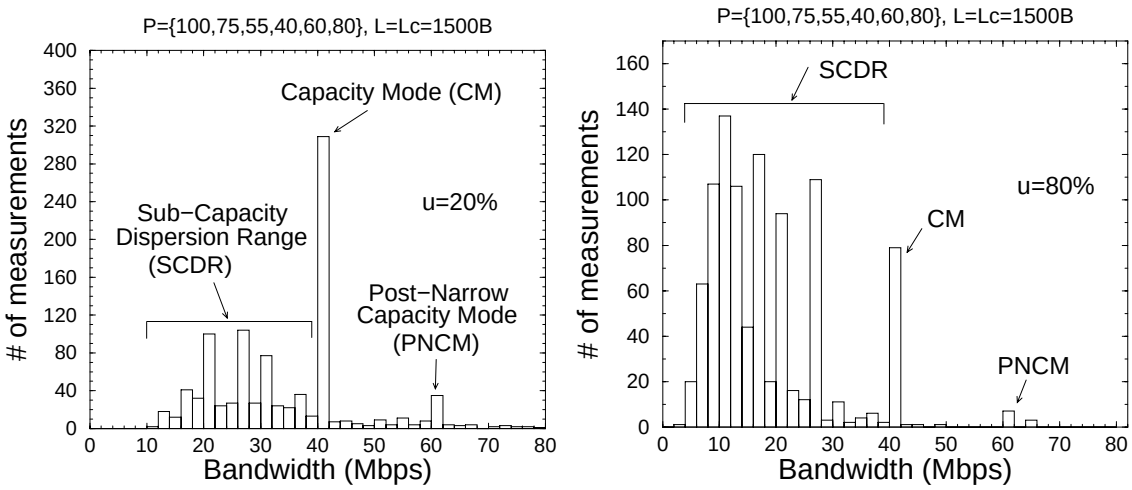

Figure 6: Examples of capacity distribution obtained with simulations. The path is constituted of six links (100, 75, 55, 40, 60 and 80 Mbit/s) and the utilization rate changes (20 % on the left, 80 on the right). Graphs extracted from [DRM01].

the packet pair is again back-to-back at the output of this equipment. The dispersion measured corresponds to the capacity of the links downstream the bottleneck.

Analysis Because of the form of the results distribution obtained during the measurements collection, `pathrate` should implement a complex technique to determine the path capacity. It consists principally in a first coarse evaluation of the capacity with averaging truncated of the extreme values. Then, as the length of the packet trains grows, the dispersion allows to evaluate another parameter, the *Asymptotic Dispersion Rate* [DRM01], always lower than the link capacity. `pathrate` chooses finally the capacity mode as the mode which value is just greater than this parameter.

As we can see, this methodology is relatively complex, needs to send many packets in the network and is hence slow for producing a result which is finally only the bottleneck capacity.

The *Packet Pair* method is used in the `nettimer` tool [LB01] too. With a more efficient results filtering method, this tool provides good results while decreasing the amount of packets sent and the duration of a measurement.

2.3 Conclusion

Existing methods to measure available bandwidth are very intrusive or too limited in rate. They don't allow either to discover the path topology and to *localize* the bottleneck. To know only the available bandwidth gives no detail about the quality of the interconnexion in terms of load, *i.e.* the ratio between the utilization and the total capacity.

The three presented classes of capacity measurement methods have their pros and cons (cf. table 1 and [PDM03]). The first common remark is that all these methods are obviously limited by the capacity (sending rate) of the sender : the provided informations are very sparse if the bottleneck is the output link of the sender, even if the measure is still valid (the bottleneck is correctly identified). Another problem caused by the capabilities of the sender is the *accurate* time measurement. To measure the dispersion of a 1 Gbit/s bottleneck with 1.5 kB packets, the sender must be able to detect a delay of about 12 µs.

There is also side effects of operating systems and hardware optimizations, such as interruption coalescing and layer 2 queues, invisible at network level. These methods have often difficulties to measure very high capacity and high contrast of both rate and load along the path. They are

nevertheless all relatively low intrusive to do a measurement: they send about ten megabytes in the network.

	Class	Type of measure	Measure	Protocol	Receiver
`pathchar`	*Variable Packet Size*	hop-by-hop	slow	UDP, ICMP	no
`tailgater`	*Packet Tailgating*	end-to-end	fast	TCP, ICMP	no
`pathrate`	*Packet Pair*	end-to-end	slow	UDP	yes

Table 1: Recapitulative table for comparizon of the three chosen tools

`pathchar` has the great advantage of providing a view of the path topology and of the characteristics (delay, capacity) of the constituting edges. The drawbacks are essentially in the implementation or in the methodology (*e.g.* slow measurement) and has been fixed or improved in other tools. The *Variable Packet Size* method is particularly sensible to the presence of layer 2 store-and-forward equipments [PDM03], invisible at layer 3. These equipments are the causes of significant errors.

`tailgater` proposes an original and clever method, but with no mean to get informations about the topology. The implementation provides the choice to do active measurements (by sending packets) or passive measurement (by using packets of others connections). By the way, it suffers from errors accumulations as soon as the path grows.

Finally, `pathrate` gives a way to identify simply the path bottleneck. Unfortunately, the used method is complex in order to solve inherent difficulties and is therefore pretty slow (around 5 minutes). As said before, it depends on the capacity of the sender to send back-to-back packets and to measure accurate spacing of the packets at the receiver. It needs by the way the collaboration of the receiver (software installation).

As a conclusion, these three tools offer good means to know a path capacity, information often unknown in an open environment like the Internet and very useful to tune transport protocols parameters or to estimate a transfer time. It turns out of this comparison that it is interesting to propose a *Packet Pair* based method, more robust considering the presence of invisible nodes than the *Variable Packet Size* method [PDM03]. It could provide an accurate and fast capacity measurement method, hop-by-hop to allow the localization of the bottleneck, without the collaboration of the receiver. We could also derive from these measurements an estimation of the available bandwidth of the path.

3 Proposition

Considering the previous methods, we propose a new measurement methodology for discovering the topology characteristics using a packet pair dispersion analysis. We will see that it is able to evaluate the utilization rate and so the available bandwidth of the path. This methods is split into a measurement gathering, a bins detection and finally an extraction of the capacity informations.

3.1 Distribution modes detection

Let us take a set of capacity measurements obtained by a *Packet Pair* technique. We want to extract from the measure distribution the set of modes. A mode is defined as a measure interval where the population increases up to a maximum and then decreases down to zero or up to increasing again (see algorithm 1, p. 10). A mode is characterized by its *location*, *i.e.* the value the extreme measures (**ltmin** and **rtmin**) and the maximum one (**max**). From the measure distribution (**distrib[]**), the function computes the set of modes (**modes[]**) and returns the number of detected modes. The differences which are below the threshold (**thresh**) are ignored to decrease the noise influence.

We will illustrate the modes detection on the example of figure 7. On this one, three modes are detected. The first one has for position $(3, 5, 7)$. The second is detected at $(8, 8, 10)$, the left

algorithm 1 Function **detect_modes** to detect modes distribution

```
   Procedure detect_modes(IN distrib[]: integer, OUT modes[]: mode)
 2: {distrib[] is the measures distribution. modes[] will store the detected modes.}
   ltmin ← 0; rtmin ← 0; max ← 0; nb_modes ← 0;
 4: for all i in distrib[] do {all bins in the distribution}
     if no current mode then
 6:    if distrib[i] ≥ thresh then {First value greater than the threshold}
         ltmin ← i; max ← i;
 8:      left minimum detected;
       end if
10:   else if the left minimum has already been detected then
       if (distrib[i] ≥ distrib[max]−thresh and distrib[i] ≥ thresh) then {Look up the maxi-
       mum}
12:      max ← i; rtmin ← i;
       else if distrib[i] ≥ thresh then {The previous bin was the maximum}
14:      rtmin ← i;
         Maximum detected;
16:    else
         End of mode;
18:    end if
     else if The maximum has been detected then
20:    if (distrib[i]−thresh ≤ distrib[rtmin] and distrib[i] ≥ thresh) then {Right minimum}
         rtmin ← i;
22:    else
         End of mode;
24:    end if
     else if End of current mode then
26:    modes[nb_modes].ltmin ← ltmin;
       modes[nb_modes].max ← max;
28:    modes[nb_modes].rtmin ← rtmin;
       nb_modes ← nb_modes +1;
30:    if distrib[i] ≥ thresh then {Two consecutive modes}
         ltmin ← i; max ← i;
32:      Left minimum detected;
       else
34:      No current mode;
       end if
36:  end if
   end for
38: {Detection termination}
   if A mode is on the way then
40:  modes[nb_modes].ltmin ← ltmin;
     modes[nb_modes].max ← max;
42:  modes[nb_modes].rtmin ← rtmin;
     nb_modes ← nb_modes +1;
44: end if
```

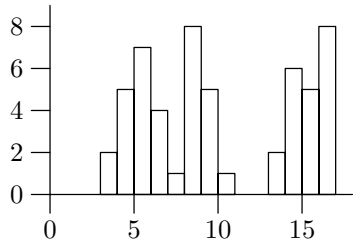

Figure 7: Modes detection example. Three modes are here detected: $(3, 5, 7)$, $(8, 8, 10)$ and $(13, 16, 16)$.

minimum and the maximum are merged. It is difficult to get a global view of the distribution: the second mode could have begin at 7 but this intuitive view is as valid as the others. Finally the third mode is located at $(13, 16, 16)$, the maximum and the left minimum are this once merged. The little variations (increase or decrease) due to the noise are ignored in the algorithm, for example between 14 and 15. This is fixed with the addition of a threshold in all comparisons.

3.2 Capacity mode extraction

The method consists in using the dispersion of a packet pair, because it has many advantages compared with the *Variable Packet Size* method [PDM03]. We have seen (cf. section 2.2.3) however that because of cross-traffic, the dispersion measurements are tainted with noise, which forces to elaborate complex analysis methodologies. There is two kinds of error: the first one is typically due to cross-traffic when packets are inserted between the two probes and hence the capacity is underestimated (cf. figure 6, SCDR). This is not too much trouble since this noise is random and so relatively wide on small values.

The second type of problems is the modes called PNCM on figure 6, since they can be on the contrary relatively acute and with a value greater than the capacity mode. These modes are obtained when the cross-traffic causes the first probe to wait for the second one before being served by the node. The probes are again back-to-back at the node output and their final dispersion is the image of the downstream capacities, after the loaded node. If we are able to gather some informations on the path topology (especially the link capacities), it is possible to eliminate this difficulty.

To know the link capacities of the path, we can use one of the principles of `pathchar`. To be more precise, if we send a back-to-back packet pair with a TTL value equal to n, we can evaluate the capacity of the hop n, between the sender and the n-th equipment of the path. Then, by doing the same measurement with a TTL value equal to $n + 1$ and by assuming the links are symmetric, the only unknown capacity is the one between nodes n and $n + 1$. If this link is not a bottleneck, it can only result in a parasitic mode greater than the capacity already estimated at hop n. It this link is the bottleneck, the previous capacity value will become a parasitic mode. It is fairly easy to identify these two situations:

- If the distribution doesn't have a relatively acute mode below the already estimated capacity, we are in the first situation: the bottleneck (up to hop $n + 1$) has already been passed and is in the previous hop ;
- Otherwise, a mode lower than the previous capacity value is detected and the links between nodes n and $n + 1$ is the new bottleneck.

Actually, the addition of a link between each measurement can only, in the worst case (loaded links), create an extra mode in the distribution.

The method lies on an incremental discovery of the path characteristics. For this, we are going to evaluate three parameters for each measurements distribution (cf. figure 8):

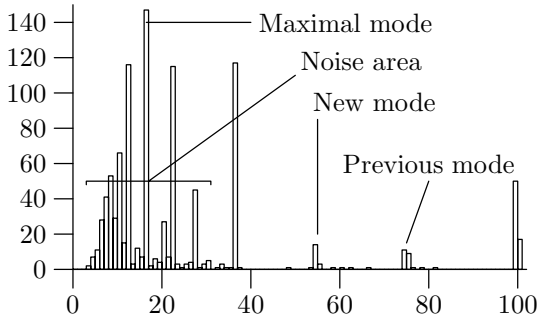

Figure 8: Example of the necessary parameters of a measurements distribution. On this example, the previous evaluated capacity was 75.

The maximal mode is the easiest to determine. It corresponds to the interval with the maximum numbers of samples ;

The previous mode is the mode of the current distribution which has the same capacity value as the one estimated for the previous hop ;

The new mode is the mode with a capacity value strictly lower[7] than the previous mode and which includes a sufficient number of samples (here 1% of the total number of measurements).

For the first hop, the previous and new mode are the same. We evaluate the **noise area** too as the little capacity values area which contains three or more side-by-side modes, *i.e.* not separated by an interval of at least a distribution step.

The rules for choosing the capacity mode are the followings. These rules are sorted by decreasing order of importance: as soon as a rule is verified, the followings don't apply. On the contrary, if a rule doesn't apply, the following ones are to be tested.

- If the maximal mode is the previous or the new mode (if the new mode is not in the noise area), the maximal mode is the capacity mode. It is the best situation (little cross-traffic, short path) ;
- If the number of samples in the new mode is greater (or at least about the same) than the previous mode, the new mode is the capacity mode if it is out of the noise area ;
- If the maximal mode is out of the noise area or contains more than 60% of the total number of measurements, the maximal mode is the capacity mode if the previous mode isn't of the same order ;
- By default, the previous mode is returned if no previous rule is valid. We avoid this way to taint the further steps, which can eventually detect the correct capacity of the path.

4 Validations

4.1 Implementation

The implementation of the previously described method is split into two modules. The first one is the measurement modules, which will send many times a back-to-back packet pair and gather the dispersion measurements. The second module do the distribution analysis according to the previous rules. The measurement tool, which name is `tracerate`, use this two modules successively to produce the result.

[7] A new mode implies that the capacity decreases.

4.1.1 Measurement gathering

The measures are done for each value of TTL between the source and the destination in order to investigate the whole path. By default, 500 packet pairs are sent for each loop with 1,400 bytes packets. We measure for each of them the differences between the RTT of the two probes. Some network configurations create extra difficulties ; some of them have been solved this way:

Rate limitation Sometimes, the rate of ICMP and UDP packets are limited in the network (in routers or firewalls) to avoid Deny of Service attacks. So we use TCP packets (`SYN` and `RST`) ;

Presence of a firewall TCP packets may be filtered by firewalls. The utilization of port 80 (HTTP protocol) or port 22 (SSH protocol) can circumvent this difficulty. Since we use the "open connection" and "end connection" messages in the packet pair, a measure doesn't saturate the connection tables of possible firewalls.

The implementation is partly based on the sources of the `tcptraceroute`[8] tool. This tool is an adaptation of the well-known `traceroute` which sends TCP packets instead of ICMP packets.

The measure gathering raises many implementation difficulties. We have first to face the problem of the accuracy of timing measure. The detection of the arrival time of a packet is disturb by the going through of software layers and is not as accurate as wanted. With the libpcap[9] library, we can capture packets at the kernel level and not at the application level. We minimize this way the crossed layers. On the other side, packets arrival are detected through interruptions raised by the interface card. But network cards[10] and/or recent operating systems[11] provide interrupt coalescing mechanisms to lower per packet communication costs. These mechanisms can disturb measurements since packets seems to arrive at the same time in the kernel. Only the first packet (or a packet every ten) triggers an interruption and initiates the processing of all the following incoming packets in the card buffer. This difficulty is hard to overcome without disabling this functionality in the card or in the OS, which can lead to others problems in a high-rate environment.

4.1.2 Measurements analysis

From the measurements, the analysis module evaluate the capacity mode. The first action is to compute the measures distribution. An important parameter of this stage is the distribution step. It is not automatically determined because a too big value harm the accuracy of the result whereas a too small value increase the noise influence. This parameter is set by the user.

Once the set of modes is extracted from the distribution using algorithm 1 (p. 10), we have to point out the four parameters of the distribution (the previous, new and maximal modes and the noise area). From this four parameters, the capacity mode is chosen according to the rules given at section 3.2.

4.2 Simulation and tests

The purpose of simulations is first to validate the proposition by verifying if the obtained behaviour is the one expected. Then, the simulations allow us to test the implementation of the two modules against various situations and so its robustness. Finally, the simulations are important to evaluate the influence of some external characteristics (utilization rate, path length), often not easily controllable in real life but completely controllable in simulations.

Simulations are conducted with the network simulator NS. The implemented model is described in the figure 9 and corresponds to the same model as the one used in [DRM01] for comparison purpose. The topology is a string, allowing to simulate every path between two nodes and is composed of 7 nodes, with symmetric 10 ms latency[12] links in between. A source (node zero)

[8] Michael C. Toren, `http://michael.toren.net/code/tcptraceroute/`

[9] `http://www.tcpdump.org/`

[10] Gigabit cards provide often this functionality, particularly useful at this kind of rate.

[11] The last stable version of Linux kernel (2.4.20) implements such a functionality with NAPI [HSOK01].

[12] The latency has little influence on the dispersion measure.

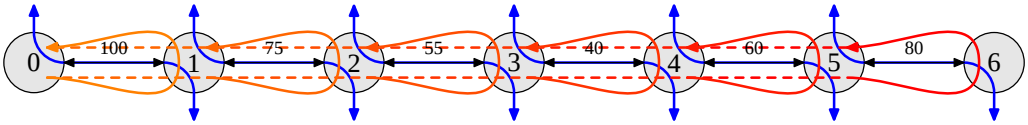

Figure 9: Topology and traffic model used in simulations. Each link (its capacity is given in Mbits/s) is crossed by a traffic flow in both directions independtly.

sends a back-to-back packet pair a thousand times for each value of TTL (from 1 to 6). Cross-traffic is generated on each link in both directions: traffic sources and sinks are set up on each extremity of the links and sends traffic with a controllable rate. The evaluated metric is the difference between the RTT of the two probes.

To validate the analysis method, we first do an experiment to insure the measurements are the one expected whatever the network conditions (load, path length) are. Then, we test with two others experiments, the accuracy and the robustness of the method versus the utilization rate and the number of hops.

4.2.1 Behaviour study

Figures 10 and 11 present the capacity measurements distribution obtained for each hop (from 1 to 6) and for an utilization rate of 20 and 80%. We observe on this graphs that the capacity of each loop can be relatively easily detected, except on long loaded loops (hop #6 and $u = 0.8$): the noise is in that case sufficiently high to disturb the detection. Otherwise, the incremental detection can be achieved. In the first hop, a capacity mode is extracted. Since the second link is the bottleneck for the second loop, it creates an other mode, lower than the previous one. We can hence safely eliminate the maximal mode and we identify the capacity mode as the second mode immediately lower than the maximal mode. We proceed like this and store in each step the detected capacity mode to eliminate it in the next hop.

If you observe the graphs of figure 10, you will see that the maximal mode is always the capacity mode: the situation is more or less ideal (small load, sufficient gap between two consecutive link capacities). If the load increases (figure 11), the maximal mode is always below the capacity mode due to cross-traffic. The incremental detection works nevertheless. For example, at hop #3, the modes for 100 and 75 Mbits/s have already been detected. The new mode is the one immediately lower, *i.e.* 55 Mbits/s, corresponding to the capacity mode.

4.2.2 Accuracy tests of the method

To validate the method for determining the capacity mode, we use the previous simulation to generate a measure batch with a varying utilization rate from 0 to 100% by step of 1%. The analysis module described in the section 4.1 is used to analyze the produced data. We want to prove that our method is reliable and accurate whatever the network conditions (load, path length) may be.

	Hop #1	Hop #2	Hop #3	Hop #4	Hop #5	Hop #6
$u \leq 0.5$	0.1%	0.1%	1.1%	2.5%	4.8%	6.9%
$u \leq 0.75$	0.1%	1.4%	4.6%	7.1%	5.9%	8.3%
$u \leq 1$	0.1%	12.4%	14.9%	15.3%	11.5%	13.7%

Table 2: Accuracy tests of the capacity evaluation. This table gives the average relative error on various utilization rate intervals and for each hop of the path.

From the table 2, we can conclude two things. First, the results remain correct as long as the path grows. Actually, we can even observe that the result is often better for high utilization rate when we are one or two hops further than the bottleneck (*e.g.* hop #4). In these high load

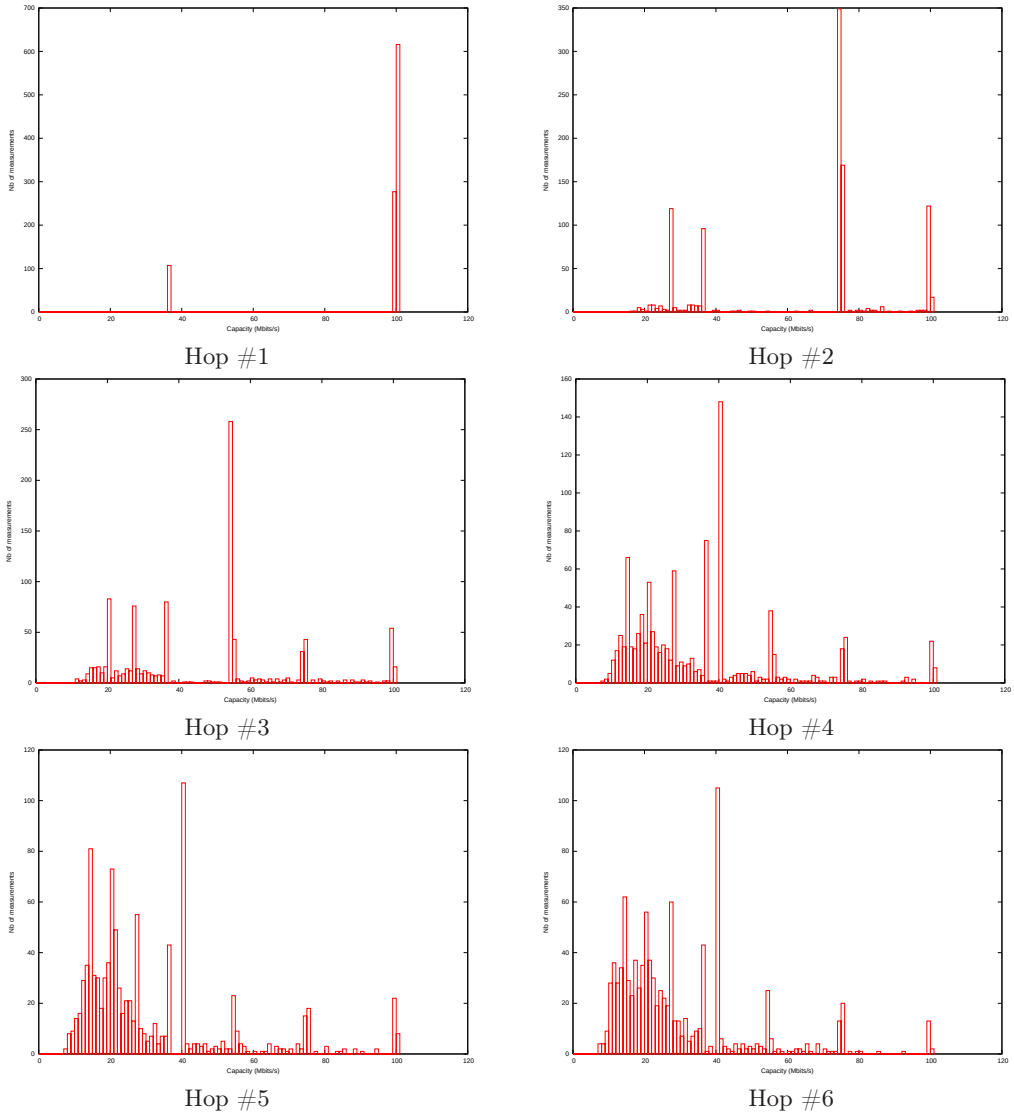

Figure 10: Distribution of capacity measurements obtained by simulation with an utilization rate $u = 0.2$. The various graphs consists in the measurements for increasing TTL loops.

situations, the method tends to be conservative and returns the previous result. The next steps are yet able to give the expected capacity value once the bottleneck is passed and the last given value is often correct, even for high utilization rate. On the other side, the measure can be qualified as reliable for low or middle range load whatever the length of the path is : the given result is always correct for an utilization rate lower than 50%.

4.2.3 Robustness tests of the method

This time, we will generate a topology with random characteristics in terms of links capacity and utilization rate. The number of nodes and links remains the same. We have done hundred simulations and extracted two informations: first, the correlation between the measure and the capacity of the bottleneck and second, the relative error between these two quantities versus the network load. The results are presented on figure 12.

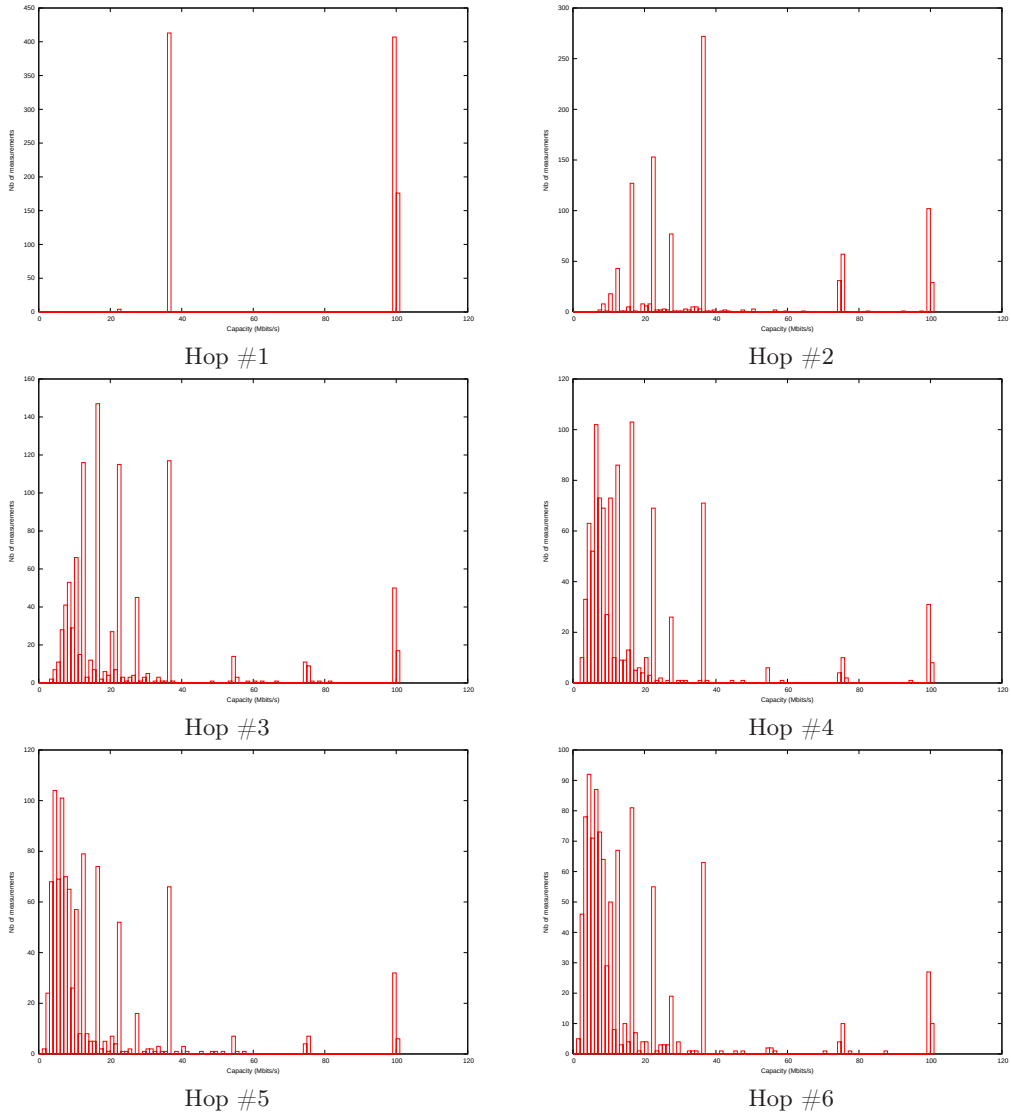

Figure 11: Distribution of capacity measurements obtained by simulation with an utilization rate $u = 0.8$. The various graphs consists in the measurements for increasing TTL loops.

First, we can see that the correlation is strong between the measure and the real value (the squared correlation factor is equal to $R^2 = 0.58$). This correlation is much stronger if we restrict the measurements to the one with an utilization rate lower than 50%: the correlation factor is then equal to $R^2 = 0.82$. The right graph show that the relative error grows logically with the utilization rate. The average relative error is equal to 0.28. But, it remains low for an utilization rate lower than 50%: the average relative error decreases then to 0.14.

The influence of the path length is important to study: the previous simulations keeps the 7 nodes topology (figure 9). To study the influence of the number of hops, we have done the previous simulation again, but with a 10 links path and random capacities and utilization rate. The R^2 factor remains of the same order as for the shorter path ($R^2 = 0.62$). If the utilization rate remains below 50% (cf. figure 13), the correlation factor is even better ($R^2 = 0.88$) and the relative error is most of the time lower than 0.1. The average relative error is equal to 0.16.

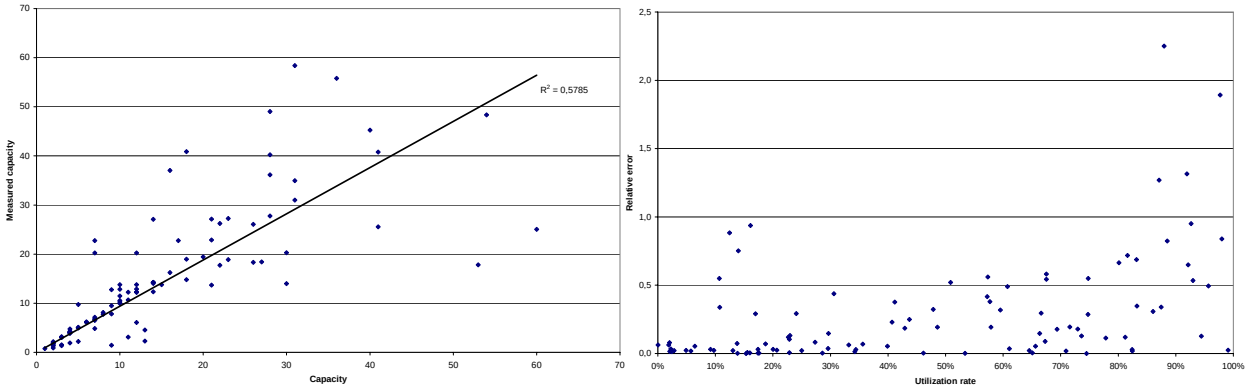

Figure 12: Robustness test of the capacity evaluation method. The left graph presents the correlation between the measured and real capacities of the bottleneck. The right graph presents the relative error between the measured and real capacities versus the utilization rate.

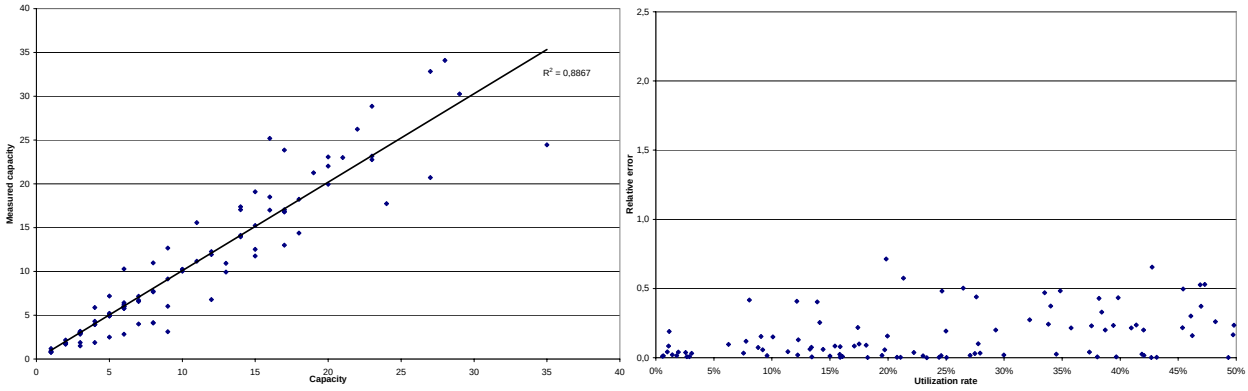

Figure 13: Robustness test of the capacity evaluation method on a 10 links path and for $u \leq 0.5$. The left graph presents the correlation between the measured and real capacities of the bottleneck. The right graph presents the relative error between the measured and real capacities versus the utilization rate.

4.2.4 Evaluation of the utilization rate

The previously proposed technique give an hop-by-hop evaluation of the capacity $\mathscr{C}$ of a path. It can also give a mean to evaluate the link utilization rate u and so the available bandwidth $\mathscr{A}$. For a sample to be in the capacity mode, the cross-traffic must have not spaced the probes more than the dispersion caused by the bottleneck.

Since the utilization rate is directly related to the probability for probes to be disturb by the cross-traffic, we can consider the measurements set as a random experiment and found a relation between the probablity p for a measure to be in the capacity mode and the utilization rate u of the link. If we extract from the previous simulations the number of samples in the capacity mode versus the utilization rate, we obtain the graph on figure 14. We observe that the relation between p and u depends on the considered hop: the dependency is linear for the first hop and less obvious for the next ones.

This graph tends to show that there is a relation between p and u. It is off course possible to derivate from simulations an empirical model, but the understanding of the mechanisms can give access to a more generic and robust method. However, we need to investigate further to obtain an analytic relation in the general case, *i.e.* for any number of hops. Such a study is future work and seems to be an important issue we really want to investigate deeply.

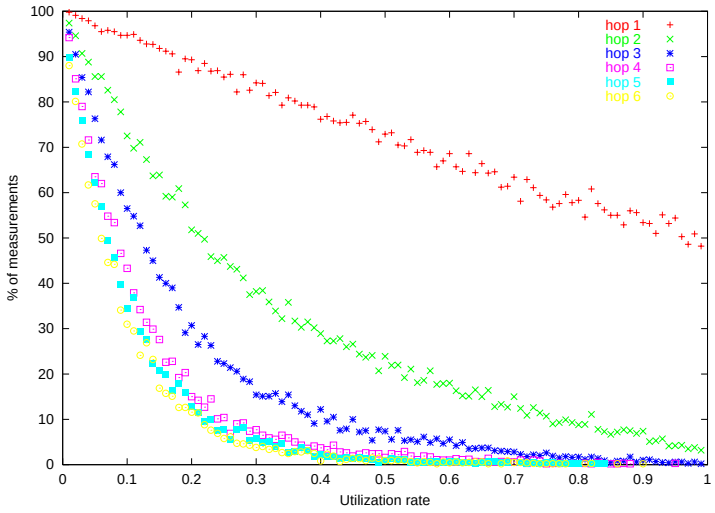

Figure 14: Evolution of the samples proportion in the capacity mode p versus the utilization rate u. The graphs represents the values for each of the six loops (cf. figure 9).

4.3 Experimental validation

Simulations give us a validation of the analysis method. The implementation in Linux of the measure module and of the tool `tracerate` dans Linux needs validations too, for example to study the influence of OS mechanisms (invisible queue [PDM03], interrupt coalescing, timing accuracy, *etc.*) which can disturb dispersion measurements. These tests raise an important difficulty: we needs to have root access on computers[13]. Moreover, we need to know completely the path (capacity, underlying link technology, routing) between the two extremities in order to validate the result. These knowledges are hard to collect on a sufficiently "complex" path (capacity changes, varying utilization rate with an accessible value, *etc.*).

We have done this experiment thanks to the European project DataTAG. The fundamental aim of this project is the creation of an intercontinental computer grid, between European and national grid projects (DataGrid) and the equivalent ones in North America. This project wants

[13]These capabilities are necessary to open raw sockets.

to explore the most advanced network technologies and the interoperability between computer grids.

It regroups teams from five europeans countries: the *Centre Européen pour la Recherche Nucléaire* (CERN, Switzerland), the *Istituto Nazionale di Fisica Nucleare* (INFN, Italy), the *Institut National de Recherche en Informatique et Automatique* (INRIA, France), the *Particle Physics and Astronomy Research Council* (PPARC, UK) and the *National Institute for Nuclear Physics and High Energy Physics* (NIKHEF, Netherland). There is eleven partners (universities et laboratories) from North America (US and Canada).

The DataTAG experimental platform offers an ideal environment for such experiments: it is made of computers on each side of the Atlantic, at CERN (Geneva, Switzerland) and Caltech (Chicago, Illinois). These two sets of machines are connected with Gigabit Ethernet throug an private optical link at 2.5 Gbit/s.

We have conduct the following tests to validate `tracerate` in a high-performance environment and to compare it with `pathchar`, the only other tool proposing hop-by-hop measurements, and `pathrate` because it uses the same packet pair technique. The test consists in doing a measurement between a machine at CERN and a machine in Chicago. The path is a 3 hop path with 1 Gbit/s links. `tracerate` gives also latency and loss figures during the measurements but they are not given here.

		`tracerate`			`pathchar`			`pathrate`		
Capacity		0	25%	50%	0	25%	50%	0	25%	50%
Hop #1	1000	165	170	165	92	92	93	N/A	N/A	N/A
Hop #2	1000	*162	*165	*162	996	977	832	N/A	N/A	N/A
Hop #3	1000	933	862	862	N/A	N/A	N/A	981-986	760-776	927-947
Duration		2'40	2'40	2'40	N/A	N/A	N/A	25"	5'30	5'40

Table 3: Capacity measurements on the DataTAG platform from CERN to Chicago. The first column contains the real capacity between the current and the previous nodes. The following columns give the measured capacity with `tracerate`, `pathchar` and `pathrate` and on different load conditions. All capacities are in Mbit/s. The N/A mention indicates that the measure didn't succeed or that this information isn't available with this particular tool. Values with an asterisk indicates that the tool has given an information message about the validity of this value. The last row gives an estimation of the measurement duration.

The results are presented in the table 3 and show that the two first tools suffer from incoherent measures (capacity doesn't decrease) due probably to an ICMP rate limitation in the first hops. The fact that these two tools aren't affected exactly the same way comes probably from the filters configuration.

On the other way, this experiment shows `tracerate` ability to perform well in a high-performance environment. Nevertheless, the ICMP rate limitation on the first measurement disadvantage a little `tracerate` because, in a normal condition, the given result would be always equal to 933 Mbit/s. Besides, an information message warns the user about this rate limitation problem. Finally, we can remark that `pathchar` doesn't manage to give a result on the last hop, again due to ICMP *Port Unreachable* rate limitation on the Linux receiver.

`pathrate` doesn't give any topology information and seems to be more sensible to network load, concerning both reliability and accuracy of the result and duration of the measure. However, the ICMP rate limitation doesn't logically bother him since it uses UDP packets.

This methods are said to be non-intrusive but they send anyway a certain amount of packets. If we evaluate the number of packets sent by `tracerate` and compare them in the same conditions with the figures given in [LB00] (cf. table 4), we can see that the amount of data sent by `tracerate` is lower than any other except `nettimer`.

Tool	4 hops path	11 hops path
`pathchar`	11,562	31,782
`clink`	6,002	16,400
`pchar`	11,732	32,417
`nettimer`	982	6,663
`tracerate`	4,000	11,000

Table 4: Comparison of the number of packets sent by various tools. The two columns represent the influence of the path length.

5 Future work and conclusion

5.1 Future work

Network monitoring and quality and performance management of a computer grid are two basic use of a tool like `tracerate`. Instantaneous measurements can be done for taking decisions quickly as well as regular measurements to store a network conditions history in an information system. These kind of data are important for the implementation and the perpetuation of a computer grid to elaborate a management system of the quality of service and even a performance prediction sytem.

More than this basic uses of the tool, you can imagine other possible uses of the analysis method. If you consider the measurements gathering can be active by sending packets or passive by using other established connexions between the computer and any other nodes of the networks, the analysis module can be seen as a service of the OS. This service can be ask for informations by transport protocols to adapt their rate or their flow control.

Such approaches are ongoing work but they often need modifications in the network equipements [Wel02]. Two visions exist actually: the first one want to keep the network as simple as possible and we have to use the already proposed functionalities the best we can to provide informations. Our proposition is included in this framework. The second vision wants to add more functionalities in the network. To ask more informations from routers is indeed very interesting because it can minimize the intrusivity while increasing the accuracy of measurements. However, this way raises a lot of deployment issues and you can't insure a service continuity along a path, not needed by a service without network participation.

Moreover, a service like ours can greatly benefit from new router functionalities to enhance accuracy, but without relying completely on them. You can then overcome the problem of service continuity and the deployment difficulties of new services in the Internet (incomplete deployment, request filtered by a firewall, *etc.*) while improving the quality of the measure with these new informations.

Otherwise, the measurement exploitation to evaluate the path utilization and so the available bandwidth is still open and promising. If this study succeed, our tool will be able to give with an unique measurement the hop-by-hop capacity until the bottleneck and the available bandwidth of a path. This method can also benefit of explicit utilization rate measurement by traffic injection in order to be calibrated and improve accuracy while not increasing intrusivity too much [NC03].

5.2 Conclusion

We have presented in the previous sections an analysis of rate evaluation method, then a proposition of a new method of capacity evaluation and topology discovery between two nodes, using a packet pair technique. We have shown that this method is relatively non-intrusive, robust, relatively accurate and reliable and keep these qualities under bad network conditions (high load, long path, *etc.*). Some inherent difficulties remain however. The first one is the same for all packet pair method: it can't evaluate capacity of aggregated channels at link layer (typically ATM channels) : the evaluated capacity will only be those of one channel.

By extension, you can't see queues at a lower level than IP. The packet pair method deals better than the variable packet size method, but these queues can disturb timing measurement, especially in the OS. Using the libpcap library tries to minimize this effect. The effect of invisible queues is particularly high and become non-negligible in high-performance networks where dispersion values are very small comparing to the timer resolution. We have show however that our tool works up to 1 Gbit/s.

We have shown too that the Linux implementation works and provides usable results in real life, without the participation of the receiving computer. Many perspectives are still open for this kind of methods: performances evaluation of an end-to-end path or utilization as an OS service or directly in a transport protocol. The fact that our tool can give in a single measure the capacity and the available bandwidth is very promising too.

List of Figures

List of Tables

References

[Bol93] Jean-Chrysostome Bolot. End-to-end Packet Delay and Loss Behavior in the Internet. In *ACM Sigcomm '93*, pages 289–298, San Fransisco, CA, September 1993.

[CM01] James Curtis and Tony McGregor. Review of bandwidth estimation techniques. In *New Zealand Computer Science Research Students' Conference*, University of Canterbury, New Zealand, April 2001.

[Dow99] Allen B. Downey. Using pathchar to estimate internet link characteristics. In *Measurement and Modeling of Computer Systems*, pages 222–223, September 1999.

[DRM01] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure? In *Proceedings of INFOCOM'01*, pages 905–914, 2001.

[FJJ+01] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y.Shavitt, and L. Zhang. IDMaps: a global Internet host distance estimation service. In *Proceedings of IEEE/ACM Trans. on Networking*, October 2001.

[HSOK01] Jamal Hadi Salim, Robert Olsson, and Alexey Kuznetsov. Beyond softnet. In 5^{th} *Annual Linux Showcase & Conference*, pages 165–172, Oakland, California, USA, November 2001. Usenix.

[Jac88] Van Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM '88*, volume 18, pages 314–329, Stanford, CA, August 1988.

[Jac97] Van Jacobson. pathchar - a tool to infer characteristics of internet paths. MSRI talk, April 1997.

[JD02] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *Passive and Active Measurements (PAM) Workshop*, March 2002.

[LB00] Kevin Lai and Mary Baker. Measuring link bandwidths using a deterministic model of packet delay. In *SIGCOMM*, pages 283–294, August 2000.

[LB01] Kevin Lai and Mary Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, pages 123–134, March 2001.

[LTC+03] Bruce Lowekamp, Brian Tierney, Les Cottrell, Richard Hughes-Jones, Thilo Kielmann, and Martin Swany. A hierarchy of network measurements for grid applications and services. Draft, Global Grid Forum, February 2003. http://www-didc.lbl.gov/NMWG/docs/measurements.pdf.

[NC03] Jiri Navratil and Les Cottrell. ABwE: A Practical Approach to Available Bandwidth Estimation. In *PAM Workshop '03*, March 2003.

[NZ02] T.S. Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proceedings of IEEE INFOCOM'02*, pages 170–179, New York, NY, USA, June 2002.

[Pax97] Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, Lawrence Berkeley National Laboratory, University of California, Berkeley, CA, April 1997. ftp://ftp.ee.lbl.gov/papers/vp-thesis/dis.ps.gz.

[PDM03] Ravi S. Prasad, Constantinos Dovrolis, and Bruce A. Mah. The effect of layer-2 store-and-forward devices. In *Proceedings of INFOCOM '03*, San Fransisco, CA, April 2003.

[PHB02] Pascale Primet, Robert Harakaly, and Franck Bonnassieux. Experiment of the NWS (network weather service) network forecasting for grid networking. In *Conference on Cluster Computing and Grid (CCGRID'02)*, Berlin, Germany, May 2002. IEEE.

[Wel02] Michael Welzl. Traceable congestion control. In *International Workshop on Internet Charging and QoS Technologies*, Zürich, Switzerland, October 2002. Springer LNCS 2511.

Contents