PingPair: a Lightweight Tool for Measurement Noise Free Path Capacity Estimation

Andrea Di Pietro, Domenico Ficara, Stefano Giordano, Francesco Oppedisano, Gregorio Procissi

Dept. of Information Engineering, University of Pisa
Via Caruso 16, 56122 Pisa, Italy
Email: {a.dipietro@netserv., d.ficara@, s.giordano@, f.oppedisano@, g.procissi@}@iet.unipi.it

Abstract—The paper presents PingPair, a novel tool for end-to-end path capacity estimation. The tool is based on the classical packet dispersion technique, enhanced by a novel algorithm for the selection of the best measurement samples based on queueing delay estimation. In addition, PingPair takes into account the measurement noise that afflicts the interarrival times registered by a user level application; we experimentally observe the Gaussian nature of such a noise. Since PingPair relies on one-point measurements only, it can be deployed in almost all network scenarios, thus providing maximum flexibility. The performance of the tool has been assessed through both NS2 simulations and extensive experimental campaigns, including Internet as well as field trial measurements. The results are compared to those achieved by Capprobe, which is one of the most effective out of the many available one-point measurement–based capacity estimation tools. Despite the very low amount of probing traffic generated, PingPair outperforms Capprobe in most scenarios, yielding more precise capacity estimates; therefore, it proves to be a very fast and unintrusive way to measure the capacity of a network path.

I. INTRODUCTION

The knowledge of the capacity of a network path can be useful for many purposes: multimedia applications, service level agreement verification, network monitoring and management. Several solutions have been proposed for this purpose and many tools are available. Out of the many proposals, the most effective techniques are those which infer the path capacity from the packet pair dispersion, i.e. the interarrival time of two packets of the same length sent back to back by the source. The concept of packet pair dispersion was first introduced in [1] but its application for bottleneck link speed estimation by a user level application; we experimentally observe the Gaussian nature of such a noise. Since PingPair relies on one-point measurements only, it can be deployed in almost all network scenarios, thus providing maximum flexibility. The performance of the tool has been assessed through both NS2 simulations and extensive experimental campaigns, including Internet as well as field trial measurements. The results are compared to those achieved by Capprobe, which is one of the most effective out of the many available one-point measurement–based capacity estimation tools. Despite the very low amount of probing traffic generated, PingPair outperforms Capprobe in most scenarios, yielding more precise capacity estimates; therefore, it proves to be a very fast and unintrusive way to measure the capacity of a network path.

Unfortunately, this kind of measurements can be seriously distorted by the queueing delay due to cross traffic: its effect on the distribution of the packet pair dispersion measurement has been studied in several papers [4]–[6]. It is generally established that, due to the influence of interfering packets, some evident modes can emerge in the sample distribution, corresponding to dispersion values which can be significantly different from the actual one. The occurrence of such values is determined by many causes, including the length of the probe packets as well as that of the cross traffic packets. In order to discard the distorted measurements, several techniques have been proposed and many tools have been implemented: among the most well-known are Pathrate [4], Capprobe [7], Nettimer [8]. However, the queueing delay is not the only cause of capacity measurement errors: the interarrival time measurements obtained by a user level application are affected by a measurement noise that is mainly due to the variable latencies related to the transfer of a packet between the wire and the application socket. In this paper we assume a Gaussian model for such noise and verify this hypothesis through experimental analysis; also, we develop a novel selective filtering algorithm in order to deal with this kind of disturb as well. Unlike most of the available tools, PingPair is based on one-point measurements: the packet pair sent by the probing host is composed by two ICMP echo request packets and the dispersion of the couple of the corresponding ICMP echo reply packets is measured by the same host. For this reason, our tool can be deployed in almost all network scenarios, since no special cooperation from the destination host is required (many other tools, such as Pathrate [4], must run on both ends of the network path). The solution proposed in this paper is based on the queueing delay estimation method described in [9] and selects, in the set of packet pair dispersion measurements, the ones which have experienced the minimum queueing delay throughout the network: the values of their dispersion is then used to derive a reliable estimate of the whole path capacity. The performance of the tool is compared to that of Capprobe [7], since both techniques are based on one point measurements (even if a client-server implementation of Capprobe is also available). The rest of the paper is organized as follows: section II illustrates the queueing time estimation algorithm which is the basis of the tool, while section IV describes the novel sample selection algorithm implemented by PingPair. Section III shows an experimental analysis of the noise affecting the measurements of interarrival and interdeparture times. Section V assesses the performance of the proposed solution by showing the results of several NS2 simulations, while section VI reports the results of several experiments performed over the Internet. Finally, section VII shows the results of experimental tests performed over a controlled field trial, under different flavors of interfering traffic; conclusions and final remarks follow thereafter.

II. QUEUEING DELAY ESTIMATION

While Capprobe selects the most reliable dispersion samples based on the probe packets’ round-trip times (or one-way
delays) our algorithm selects the best measurements on the basis of an estimate of the probe packets’ queueing delay. The queueing delay estimation technique described in this section has been originally proposed in the field of tomographic network topology discovery, but it can be applied without significant modifications to packet pairs, since it is based on the knowledge of the packets’ interdeparture and interarrival times only.

Let us therefore indicate with \( p_k^{(n)} \) the \( k \)-th (with \( k \in \{ 1, 2 \} \)) packet of the \( n \)-th packet pair and let us assume that both its departure time \( \pi_k^{(n)} \) and its arrival time \( a_k^{(n)} \) are deterministically known; this hypothesis is easily satisfied since both the arrival times and the departure times are measured by the probing host. The end-to-end delay that the probing host experiences is given by the sum of a deterministic time \( R_k \) (the one experienced by packets that do not interact with cross traffic) and a random, non-negative queueing delay \( Q_k^{(n)} \), induced by the interaction of cross traffic. The packet arrival time \( a_k^{(n)} \) is therefore:

\[
a_k^{(n)} = \pi_k^{(n)} + R_k + Q_k^{(n)}  \tag{1}
\]

The interarrival time \( X_k^{(n)} \) of packets \( p_k^{(n)} \) and \( p_k^{(n+1)} \) can therefore be expressed as:

\[
X_k^{(n)} = a_k^{(n+1)} - a_k^{(n)} = Q_k^{(n+1)} - Q_k^{(n)} + (\pi_k^{(n+1)} - \pi_k^{(n)}) \tag{2}
\]

where \( \tau_{n+1} \) is the interdeparture time of packets \( p_k^{(n)} \) and \( p_k^{(n+1)} \). Equation (2) can be seen as a recursive relation between the queueing delay of the corresponding packets belonging to consecutive packet pair probes:

\[
Q_k^{(n+1)} = Q_k^{(n)} + X_k^{(n)} - \tau_{n+1}  \tag{3}
\]

Unfortunately, the initial condition \( Q_k^{(0)} \) is unknown; nevertheless, by choosing any arbitrary initial condition, the queueing delay estimates given by (2) are biased by the same, constant, initial error \( \epsilon_k \). By indicating with \( \hat{Q}_k^{(n)} \) the sequence of queueing delay estimates, the following relation holds:

\[
\hat{Q}_k^{(n)} = Q_k^{(n)} + \epsilon_k  \tag{4}
\]

Therefore, it is possible to reorder the set of packet pairs in terms of their estimated queueing delay \( \hat{Q}_k^{(n)} \) in that, as previously mentioned, the constant estimation bias does not modify the sequence order.

The packet probes experiencing the smallest queueing delay are those which most likely give the best (i.e. less corrupted by cross traffic interference) interarrival time measurements. Equation (3) holds for both packets \( p_1 \) and \( p_2 \) of a packet pair: the sample selection is therefore based on the two estimates sequences \( \hat{Q}_1^{(n)} \) and \( \hat{Q}_2^{(n)} \).

Unfortunately, given the particular structure of (3), the sequences of estimates can be seriously affected by the propagation of measurement errors. Let us therefore consider the measured interarrival time of packets \( p_k^{(n)} \) and \( p_k^{(n+1)} \); the measured time interval \( X_k^{(n)} \) can be expressed as:

\[
X_k^{(n)} = Q_k^{(n+1)} - Q_k^{(n)} + \tau_{n+1} + \sigma_k^{(n)}  \tag{5}
\]

where \( \sigma_k^{(n)} \) is a random measurement noise term. In section III we will further investigate the causes and the characteristics of the measurement noise, showing that each sample \( \sigma_k^{(n)} \) can be modeled as a zero mean Gaussian random variable. If compared to the other time intervals involved in the estimation process, each sample of such a noise can be considered negligible; nevertheless, as it is easy to infer from (3), all the noisy terms are summed up together by the recursive estimator. As a consequence, after some dozens of iterations, the estimated queueing delay \( \hat{Q}_k^{(n)} \) can be affected by a considerable amount of measurement noise. Even though the measurement noise will be shown to have zero mean, its variance grows at each iteration of the recursive estimation. The selection of the best packet pair dispersion sample would then be based on an unreliable information.

It is then necessary to devise an effective sample selection algorithm to take advantage of the information provided by (3) and to cope with the effects of measurement noise.

III. ANALYSIS OF THE MEASUREMENT NOISE

The noisy term \( \sigma_k^{(n)} \) takes into account all the phenomena that influence the interarrival time measurements and that cannot be included in the queueing delay term; a wide variety of elements can contribute to the measurement error but the most relevant source of noise is introduced by the measurement process at the user level application. Such uncertainty is mainly due to the variable latencies associated with the interrupt based mechanism which is in charge of transferring a packet between the Network Interface Card and the application socket. The extent of such a delay is related to the features of the computer and the operating system on which the tool runs; moreover, it heavily depends on the workload of the whole system. With reference to (3), each \( \sigma_k^{(n)} \) term can be expressed as the difference of the noise affecting the interarrival time measurement and the one affecting the interdeparture time measurement. Since both the noisy terms are due to the same causes and are related to a wide variety of independent phenomena, we assume that the overall measurement noise \( \sigma_k^{(n)} \) may be safely modeled as a zero mean Gaussian random variable. We will assess the validity of such a hypothesis by analyzing the results of the experiments we performed on an ad-hoc experimental testbed.

A. Experimental testbed layout

Figure 1 depicts the testbed used to characterize the measurement noise. The probing PC, which runs the PingPair application and probes a remote server (probed server in the picture), is connected to a 1Gbps Ethernet switch featured with the port mirroring functionality; this switch is in turn connected to the Internet via a 10Mbps Ethernet switch (this is the bottleneck link). The two ingress interfaces involved in the
terms interdeparture times, we extracted the values of the noisy
B. Analysis of the experimental data
PC are compared to characterize the measurement noise.
by the probing PC and those captured by the DAG-equipped
with a DAG card in order to take on-wire hardware packet
timestamps. The interarrival and interdeparture times captured
by the probing PC and those captured by the DAG-equipped
PC are compared to characterize the measurement noise.

B. Analysis of the experimental data

After measuring the actual values of the interarrival and
interdeparture times, we extracted the values of the noisy
terms $\sigma_k^{(n)} \forall n$ and performed some statistical analysis on the
experimental data, in order to verify our hypotheses.

By averaging the whole set of noise samples, we found
that the zero mean hypothesis is well grounded: the sample
mean was always several orders of magnitude smaller than the
samples themselves.

In order to verify the Gaussian distribution hypothesis, we
further calculated the normal probability plot of each set of
data. In all cases, the normal probability plots associated with
the experimental data approximated quite well the linear plot,
which is typical of a Gaussian distribution. An example of such
a plot is shown in figure 2. As a more formal verification of
our hypothesis, the Smirnoff-Kolmogorov test (with $\alpha = 0.05$)
on the same set of experimental data yielded a positive result,
thus confirming our qualitative conclusion.

The standard deviation of the measurement noise we ob-
served in our experiments turned out to be of the order of about
$10^{-7}\text{sec}$. Even if the noisy terms themselves are negligible (as
compared to the measured dispersion values with a bottleneck
of $10\text{Mbps}$), after a few dozens of iteration of (3) the overall
noise can determine significantly wrong estimations of the
queueing delay.

IV. MEASUREMENTS SELECTIVE FILTERING

The simplest approach to cope with the accumulation of
the measurement noise is to split the whole sequence $Q_k^{(n)}$
(which includes all the queueing delay estimates) into several
subsequences, each of them evaluated from null initial condi-
tions. The queueing delay estimates can then be computed
over disjoint sets of consecutive samples and the measurement
error propagation is limited to each set.

The experimental results that have been previously shown
suggest that about a dozen of samples for each subsequence
turns out to be a reasonable choice in order to keep the overall
measurement error at negligible values.

Once the subsequences have been calculated, the best (i.e.
less influenced by the queueing delay) sample out of each
of them is selected. Notice that, if the subsequences are
composed by a few packet pairs, it is possible that none of
the corresponding packets has crossed the network without
being significantly affected by queueing delay; a sample is
accepted only if both the first and the second packets of
the corresponding packet pair are the best (i.e. they have
the smallest estimated queueing time) of their subsequences.
After selecting the best packet pair dispersion measurements,
the dispersion is calculated as the statistical mode of their
distribution. More formally, the algorithm steps are described
in Algorithm 1.

Algorithm 1 Algorithm for the selection of the most reliable
dispersion measurements.

- Let $N_{sub}$ be the number of samples composing a subset
  and $N$ be the total number of available samples.
- Let $B$ be the (initially empty) set of the reliable dispersion
  measurements.
- Partition the whole set of packet pair samples in a $N_{sub}$
  non-overlapping subsets composed by $N_{sub}$ consecutive
  probes. Let $S_l$ be the $l^{th}$ subset: it will be then composed
  by packet pairs $\{(l-1) \times N_{sub} \ldots l \times N_{sub} - 1\}$.
- For each subset $S_l$ and for $k \in \{1, 2\}$:
  - $Q_{k}^{(l-1) \times N_{sub}} = 0$;
  - compute $Q_{k}^{(l-1) \times N_{sub}} \forall m \in \{l-1 \times N_{sub} + 1 \ldots l \times N_{sub} - 1\}$ using equation (3);
  - if $\exists h|Q_{h}^{(l-1) \times N_{sub}} = \min_{j \in S_l} Q_{j}^{(l-1) \times N_{sub}}\forall k \in \{1, 2\}$ add the
dispersion measured by packet pair $h$ to the set $B$.
- Compute the dispersion as the mode of the distribution
  of the dispersion values in $B$.

The rationale of such an approach is intuitive: despite the
acceptance criterion previously described, a “bad” sample
can be selected out of a subsequence if no “good” (i.e. not
significantly afflicted by queueing delay) samples belong to
that subsequence. Although the selected “good” samples will
be concentrated in the neighborhood of a mode centered in the nominal dispersion value, “bad” samples will only result as outliers. The effects of the application of this selection criterion can be easily noticed by comparing figure 3 and figure 4: the first figure shows the distribution of the dispersion of 500 packet pairs sent to Google (www.google.it) over the Internet, while the second figure shows the distribution of the “good” samples selected by algorithm 1 out of all the measurements. It is evident that, while in figure 3 the main mode corresponds to a dispersion value which is significantly lower than the actual one (which is indicated in both figures by the empty sample and corresponds to the actual 10 Mbps bottleneck link capacity), the dispersion of the selected samples is concentrated nearby the theoretical dispersion value.

V. PERFORMANCE EVALUATION THROUGH NS2 SIMULATIONS

In order to assess the performance of our tool in a totally controlled network scenario, we ran several simulations by using NS2 [10]. We compared the performance to that of Capprobe, which can be simulated by using the NS2 module available at [11]. The simulation scenario consists of a 6-links path with capacities of 51.84 Mbps, 155.52 Mbps (typical of the OC links) and 10 Mbps (the bottleneck link). The cross traffic is generated by one–hop persistent TCP connections and the traffic load is the same on each link of the path.

Both capacity estimation tools have been tested in a wide range of path load conditions; in each scenario 30 independent simulations have been performed to properly estimate a confidence interval for the performance of both tools. In order to perform a fair comparison, both estimates were based on a set of 200 probes; such an amount of probes is fairly small if compared to the that needed by other tools (e.g.: Pathrate sends at least 1500 probes to estimate the capacity of the path). The results of these tests are summarized in fig. 5.

The results show that the estimates provided by PingPair are generally more accurate than those provided by Capprobe, since the central values of the corresponding confidence intervals is, in most of the cases, closer to the actual capacity value of 10Mbps. In addition, the estimates provided by Capprobe generally exhibit larger variability, as it can be inferred from the wider extension of the corresponding confidence intervals.

VI. INTERNET MEASUREMENTS

In order to test PingPair in a real network scenario, we implemented it by writing a user level application which, at present, is still at its beta version, but will be available as soon as possible at the website of our research goup (http://netgroup.iet.unipi.it).

From a host located in the Network Laboratory of the Dept. of Information engineering in Pisa, we sent packet pair probes to several hosts over the global Internet and estimated the capacity of the corresponding paths; the bottleneck was always the 10Mbps Ethernet link connecting the sending host to the network. The location of the bottleneck link in the first hop of a path is quite common in real scenarios, since the links at the edges of a network are often slower than those located in the core. Again, we compared the performance of our tool to that achieved by the Linux implementation of Capprobe [11]. We always ran Capprobe immediately after PingPair, so as to test both tools in the same network conditions. The two tools used the same number of packet pairs.
The results of several experiments are reported in table I and confirm the capability of PingPair of providing fairly good estimates by using an extremely limited number of samples.

<table>
<thead>
<tr>
<th>path dest.</th>
<th>dest. addr.</th>
<th>probes</th>
<th>PingPair</th>
<th>Capprobe</th>
</tr>
</thead>
<tbody>
<tr>
<td>google.it</td>
<td>66.249.93.147</td>
<td>101</td>
<td>9.630</td>
<td>15.059</td>
</tr>
<tr>
<td>youtube.com</td>
<td>208.65.153.251</td>
<td>151</td>
<td>8.680</td>
<td>11.907</td>
</tr>
<tr>
<td>myspace.com</td>
<td>216.178.39.13</td>
<td>101</td>
<td>9.867</td>
<td>3.413</td>
</tr>
<tr>
<td>ucla.edu</td>
<td>169.232.33.135</td>
<td>151</td>
<td>8.204</td>
<td>3.436</td>
</tr>
<tr>
<td>ecole-francaise.it</td>
<td>193.204.146.3</td>
<td>101</td>
<td>1.986</td>
<td>1.390</td>
</tr>
<tr>
<td>gmail.com</td>
<td>66.249.91.18</td>
<td>101</td>
<td>9.414</td>
<td>10.449</td>
</tr>
<tr>
<td>hotmail.com</td>
<td>213.19.160.118</td>
<td>101</td>
<td>8.467</td>
<td>15.515</td>
</tr>
<tr>
<td>berkeley.edu</td>
<td>169.229.131.92</td>
<td>101</td>
<td>8.641</td>
<td>16.516</td>
</tr>
<tr>
<td>mit.edu</td>
<td>18.7.22.83</td>
<td>101</td>
<td>9.491</td>
<td>3.507</td>
</tr>
<tr>
<td>Tin DNS</td>
<td>212.216.112.112</td>
<td>101</td>
<td>9.830</td>
<td>0.342</td>
</tr>
</tbody>
</table>

VIII. CONCLUSIONS AND FINAL REMARKS

In this paper we have presented PingPair, a novel capacity estimation tool which is based on the well known packet dispersion technique. This technique is enhanced by a novel selective filtering algorithm which can deal with both the sample distortion caused by the queuing delay experienced by the probe packets and with the measurement errors that are mainly due to the variable latencies between the wire and the user level applications. We performed an experimental analysis of such a measurement error and verified the hypothesis of its Gaussian nature. Then we assessed the effectiveness of our tool through NS2 simulations, Internet experiments and field trial measurements and by comparing its performance with that achieved by Capprobe in the same network scenarios. PingPair proves to be generally more accurate than Capprobe and it generally yields fairly good estimates even with a very small number (always less than 200) of packet pair probes.

REFERENCES