Passive Online RTT Estimation for Flow-Aware Routers using One-Way Traffic*

Damiano Carra,[1] Konstantin Avrachenkov,[2] Sara Alouf,[2]
Alberto Blanc,[2] Philippe Nain,[2] and Georg Post[3]

[1] University of Verona, Verona, Italy, damiano.carra@univr.it
[2] INRIA, Sophia Antipolis, France, first.last@sophia.inria.fr
[3] Alcatel-Lucent Bell Labs, Nozay, France, georg.post@alcatel-lucent.fr

Abstract. With the introduction of new generation high speed routers, and with the help of "flow-aware" traffic management, it becomes possible to improve the Quality of Service for users as well as the network efficiency for ISPs. An example of the "flow-aware" traffic management is the Alcatel-Lucent "Semantic Networking" framework where short-lived flows are processed with high priority and long-lived flows are controlled on a per flow basis. In order to control efficiently the flows, it is useful to know an estimate of the Round Trip Time (RTT). In the present work, we provide an online RTT estimation algorithm which is passive and needs one-way traffic only. The one-way traffic requirement is essential for the application of the algorithm for "flow-aware" traffic management inside the network. To the best of our knowledge, there was no online one-way traffic RTT estimators. Tests on a controlled testbed and on the Internet demonstrate high accuracy of the proposed estimator.

1 Introduction

Customers' increasing demand for better quality of service (QoS) and quality of experience (QoE) [1] have increased the interest in techniques for actively managing and controlling the traffic inside the network. With the introduction of the new generation of high speed routers, it becomes possible to treat individually a significant number of flows. An example of "flow-aware" traffic management is Alcatel-Lucent's framework of "Semantic Networking;" cf. [2].

The framework "Semantic Networking" takes advantage of the *mice-elephants* phenomenon. Many measurement studies have observed that the traffic is approximately composed of two types of connections: short-lived and long-lived flows, also known as *mice* and *elephants*. It has been observed that the number of flows of each type and the actual traffic they generate can be summarized by the 80-20 rule: mice account for the majority of the flows (80%), but the volume of the traffic associated to them represents 20% of the total traffic, while elephants (20% of the flows) convey 80% of the total traffic. Recent measurements [1] show that this proportion is shifting to a 90-10 rule. Therefore, by serving the short-lived flows with high priority one can significantly decrease the number of active flows. The efficiency of such approach has been validated

* This work was done in the framework of the INRIA and Alcatel-Lucent Bell Labs Joint Research Lab on Self Organized Networks.

by [3, 4]. For the long-lived flows, it has been observed that the instantaneous number of elephants present in the router is small, actually only few hundreds [5]. This means that it is possible to manage long-lived flows on *per flow* basis. Such an approach can give to an ISP a greater flexibility in managing its network, potentially increasing its efficiency and providing customers with high QoE [2].

One of the most important notions for a TCP flow is its aggressiveness that is how fast a TCP connection increases its sending rate. Most deployed TCP versions are based on a congestion window whose value is changed every Round Trip Time (RTT). This implies that the RTT is one of the principal parameters that determine the aggressiveness of a TCP flow, and it needs to be taken into account for the design of new "flow-aware" traffic management schemes.

In this paper, we design a method based on spectral analysis for the *online* estimation of the RTT by passively monitoring, in real-time, only one direction of a TCP flow. Specifically, our algorithm satisfies different constraints. The estimation is passive, as the measuring point (e.g., the router) does not inject packets into an existing flow, nor does it alter their flow. The estimation is done in *real-time*, since the flow rate and its rate growth should be instantaneously available at the measuring point. This constraint implies that the used algorithms must be both computationally efficient and working incrementally as new packets arrive. In other words, we need to find efficient *online* algorithms that provide sufficiently accurate results. The last constraint imposes the use of information on only *one direction* of a TCP flow. In fact, even when forward and reverse traffic do flow through the monitoring point, collecting and real-time processing of two-way traffic may impose excessive load and complexity on the network cards. As we will discuss in detail in the ensuing section on related work, none of the existing methods for RTT estimation satisfies simultaneously the above mentioned criteria.

Our contribution is threefold. First, since typical implementations of spectral analysis tools are for offline use, we reformulate one implementation of the *periodogram* to make it suitable for online use. Second, using a pattern matching technique, we develop an algorithm for extracting the fundamental frequency—whose inverse is the estimated RTT—once the spectrum has been estimated. Third, we validate our RTT estimation algorithm by conducting experiments on a controlled testbed and on the Internet during both peak and off-peak hours.

The experimental results show that our solution is able to accurately estimate the RTT, the estimation error being within $\pm 10\%$ (resp. $\pm 20\%$) of the true value with probability equal to 75% (resp. 99%). The results given by our methodology show a higher accuracy with respect to the results obtained in previous studies (see Sect. 2 for a review of the literature), using less information, namely packets flowing in only one direction.

2 Related Work

The RTT estimation has been the subject of many studies. The aim of these studies is to understand the characteristics of the TCP connections in the Internet, in order to study different aspects, such as the rate-limiting factors or

the non-conforming TCP senders. These works consider methods that can be applied offline, since they are generally computationally intensive. For instance, the work of S. Jaiswal et al. in [6] is based on the reconstruction of the TCP congestion window values, which requires to maintain a "replica" of the TCP sender's state. This work was later extended by J. But et al. in [7], maintaining however the computational complexity.

Another example is the work of Y. Zhang et al. who propose in [8] a method based on time correlation of samples: while the computational complexity is similar to our approach, the main problem is related to robustness, since the results are strongly affected by noise, which impacts the accuracy of the RTT estimation.

In [9], R. Lance et al. make use of spectral analysis—the basic mechanism used by our solution—as one of the possible steps for off-line estimation of the RTT. In particular, the authors apply the spectral analysis to a set of samples, but they do not consider the continuous, real-time update of the spectrum as new samples arrive. Moreover, the post processing of the spectrum for the extraction of the RTT is based on a simple evaluation of the frequency with the maximum power, which not always corresponds to the fundamental frequency.

In [10], B. Veal et al. propose a method based on the TCP timestamp option. Our method does not rely on information provided by the protocols, but it is based only on the inter-arrival times of the packets. In [11], Jiang and Dovrolis estimate the RTT using the first packets of a connection, this method is therefore not generally applicable to the continuous monitoring of a connection.

In [12], Y. Qi et al. propose a method for Bayesian spectrum estimation based on Kalman filtering. Nevertheless, this method is not applied to RTT estimation. The main issue is its complexity: the filter gain computation requires a matrix multiplication, whose complexity is approximately $O(N^{2.8})$ (N is the number of samples used); there exist algorithms with $O(N^{2.376})$, but with much higher constants, which makes them practically unusable. Our methodology has a complexity of $O(N)$ at each sample arrival, since $2N$ is the number of frequencies in the spectrum (see Section 5), thus, if we consider N consecutive samples, the complexity becomes $O(N^2)$.

3 Motivations

The estimation of the RTT is a building block for *actively controlling the traffic* inside the network. We focus on a flow-aware networking approach, where routers are able to manage flows –a connection identified by the classical five-tuple of protocol ID, source and destination addresses and ports– rather than simply packets. For a detailed view of the flow-aware networking approach the interested reader is referred to [2].

It has been observed in [5] that the number of concurrent long-lived flows at a router is of the order of hundreds. This means that it is possible, with the current technology, to individually manage these flows.

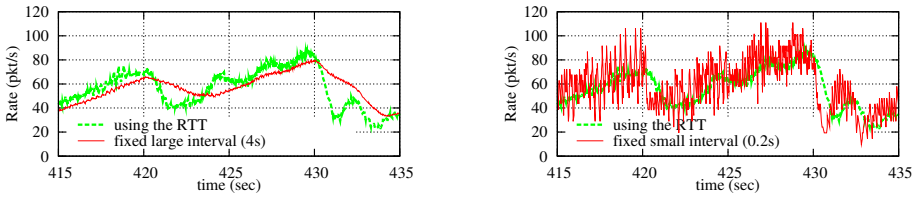

Fig. 1. Rate estimate using the estimated RTT, compared with the same using two fixed averaging intervals: large interval (left) and small interval (right).

Flows for large data transfers, which carry 90% of the total traffic [1], use mainly TCP to compete for bandwidth. More specifically, they use window based congestion control, where each flow increases its sending rate until a packet loss is detected (or an explicit congestion notification is received). Conventional routers that are not flow-aware cannot insure fairness under high traffic loads, especially when the competing flows have different RTTs [13, 14].

With flow-aware routers, there is the possibility of new flow-aware active queue management (AQM) algorithms capable of proactively controlling each flow. Being able to predict the behavior of a flow can be very useful in order to achieve this goal. In the case of a TCP connection, its future behavior (in the absence of a packet drop or congestion signal) is dictated by the increase of the congestion window. The rate of the increase depends on the RTT, as the congestion window is incremented by one or more packets each RTT. Therefore, tracking in real time this parameter is a fundamental step in predicting the evolution of a flow. While the RTT is available at the sender, current TCP versions do not propagate this information to the routers; only experimental protocols like XCP convey this information explicitly.

Furthermore, if the RTT is known, it is possible to better estimate the flow rates: by averaging the number of packets received within an interval equal to RTT it is possible to correctly estimate the TCP throughput. A smaller value of the averaging interval results in a fluctuation of the estimate, while a bigger value does not allow for a prompt detection of rate changes. This is illustrated in Fig. 1, which shows an experiment's rate estimation made with our estimation tool (cf. Section 5), compared with the same using fixed averaging intervals.

If the information about the rate and its growth is made available at the router, it is possible to design novel AQM policies that take into account this information. While the study of such AQM policies is not the focus of this paper, we highlight that the estimation of the RTT represents an important building block for such policies.

4 A methodology Based on Spectral Analysis

The self-clocking mechanism of TCP introduces *periodic* components into the arrival times of packets, the main one being the RTT. The basic idea of the RTT estimation is to use spectral analysis to extract such periodic components. With spectral analysis, it is possible to build an estimation of the spectrum of

a signal starting from a finite sequence of samples. There is a large set of useful methods suitable for different scenarios [15]. In order to choose the best method, the first step in spectral analysis is to identify the characteristics of the *signal* whose spectrum is to be estimated.

In our case, the signal is the packet inter-arrival time of the flow at hand. This signal is sampled at each packet arrival. More precisely, at the arrival of the kth packet of the flow at hand, a new sample of the signal is computed, namely $h_k := t_k - t_{k-1}$, with $k \geq 1$ where t_k is the arrival instant of the kth packet and $t_0 := 0$. Since samples are taken at packet arrivals, the sequence h_k is *unevenly* spaced. The signal is said to be *irregularly* sampled (also unevenly sampled or nonuniformly sampled). While the literature on regularly sampled data proposes many efficient methods to estimate the signal's spectrum, the proposed solutions for irregularly sampled data have been found to be impractical [16]: The choice of spectral analysis for irregularly sampled data is essentially limited, for either technical or computational reasons, to the *periodogram*.

In case of irregularly sampled data, the periodogram is computed using the Lomb-Scargle method [17] and it is generally referred to as the Lomb periodogram. Even if the periodogram has some limitations, it remains an efficient solution in many cases [15]. Considering our case, thanks to the characteristics of the signal h_k, the periodogram *does* represent a good estimator (the interested reader is referred to the companion technical report [18] for details).

A problem common to all the methods for spectral analysis (for either regularly or irregularly sampled data, including the periodogram) is that they are based on offline algorithms: they consider a sequence of N samples and they build an estimation of the spectrum. If the signal is composed of more than N samples, it is divided into subsequences of N samples and the methodology is applied to each subsequence separately. This approach is not suitable for the RTT estimation by a flow-aware router, which has to closely follow the variation of the RTT in real-time. It is extremely important to have a spectrum estimation methodology that is able to update the estimate at each packet arrival. The solution should also be computationally efficient. The simplicity of the periodogram helps in designing an *online* version of the method. All the other methods (even the more advanced ones recently proposed for irregular sampled data, see [16]) are based on complex optimized algorithms, and cannot be translated into corresponding online versions.

5 Estimation Process

At each packet arrival a new sample is added to the sequence h_k. The estimation process takes as input the new packet arrival time and updates the current estimation of the RTT. The functional blocks are shown in Fig. 2.

Given a periodic discrete signal, with period T_0 and frequency $f_0 = 1/T_0$, the periodogram of the signal presents peaks at frequencies $f_0, 2f_0, 3f_0, \ldots$, where f_0 is called the *fundamental frequency*. In general, the highest peak in the periodogram is not necessarily at f_0, but it can be at any multiple of f_0; therefore,

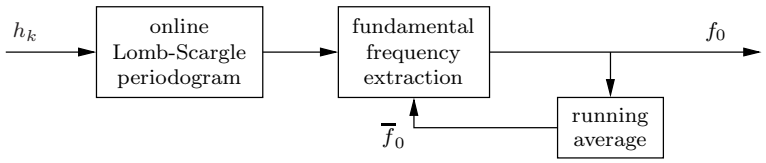

Fig. 2. The RTT estimation process: after the update of the spectrum estimation, the fundamental frequency f_0 is extracted; the RTT estimation is the inverse of f_0.

it is not sufficient to look for the largest peak in the spectrum to find f_0. For this reason, we need another module to extract the fundamental frequency. This module takes as input a list of the spectrum peaks and the average of the previous estimations. The fundamental frequency is extracted with a pattern-matching technique. Basically, after smoothing the periodogram through a low-pass filter, we take the largest W peaks and iteratively search for the frequency in the list that is the least common divisor of the other frequencies in the list.

5.1 Online Lomb Periodogram

The Lomb-Scargle periodogram [17] is built using N samples. Once the initial N samples are collected and the Lomb periodogram is built, every time a new packet arrives, the oldest sample is removed and the new sample is used to update the Lomb periodogram.

The offline version of the Lomb periodogram takes $O(N \log N)$ operations. To the best of our knowledge, no online implementation (i.e., update of the periodogram as the samples arrive) exists, so we had to devise one. For the online computation, we start from the definition of the Lomb periodogram. The power spectrum (Lomb periodogram), at angular frequency $\omega := 2\pi f$, and at the kth sample with $k \geq N$, is

$$P_k^N(\omega) := \frac{1}{2\sigma_k^2} \left\{ \frac{\left[\sum_{j=0}^{N-1} (h_{k-j} - \bar{h}_k) \cos \omega(t_{k-j} - \tau_k)\right]^2}{\sum_{j=0}^{N-1} \cos^2 \omega(t_{k-j} - \tau_k)} + \frac{\left[\sum_{j=0}^{N-1} (h_{k-j} - \bar{h}_k) \sin \omega(t_{k-j} - \tau_k)\right]^2}{\sum_{j=0}^{N-1} \sin^2 \omega(t_{k-j} - \tau_k)} \right\} \quad (1)$$

where $\bar{h}_k$ and σ_k^2 are the mean and variance of the N last samples of h_k:

$$\bar{h}_k := \frac{1}{N} \sum_{i=0}^{N-1} h_{k-i} \;=\; \bar{h}_{k-1} + \frac{h_k - h_{k-N}}{N}; \quad (2)$$

$$\sigma_k^2 := \frac{1}{N-1} \sum_{i=0}^{N-1} h_{k-i}^2 - \frac{N}{N-1} \bar{h}_k^2 \;=\; \sigma_{k-1}^2 + \frac{h_k^2 - h_{k-N}^2}{N-1} + \frac{N}{N-1} \left(\bar{h}_{k-1}^2 - \bar{h}_k^2\right), \quad (3)$$

and where τ_k is the solution of: $\tan(2\omega\tau_k) = \dfrac{\sum_{i=0}^{N-1} \sin 2\omega t_{k-i}}{\sum_{i=0}^{N-1} \cos 2\omega t_{k-i}}$. The summations in (2) and (3) can be efficiently updated at each packet arrival.

The periodogram (1) is evaluated for a number of frequencies equal to $2N$. In particular, we compute the minimum frequency at arrival of packet k, $f_k^{\min}$, considering the interval of time of the N current samples, i.e., $f_k^{\min} = 1/\left(t_k - t_{k-N+1}\right)$. The maximum frequency, $f_k^{\max}$, is derived from the Nyquist theorem: since we have N samples, we obtain $f_k^{\max} = \frac{N}{2} f_k^{\min}$. The interval $[f_k^{\min}, f_k^{\max}]$ is divided into $2N$ values, obtaining $\Delta_\omega = 2\pi \frac{f_k^{\max} - f_k^{\min}}{2N}$. Thus, we compute $P_k^N(\omega_i)$ for every $\omega_i = 2\pi f_k^{\min} + i\Delta_\omega$, $i = 0, \ldots, 2N-1$.

The power spectrum is to be updated at each packet arrival. Since the values of τ_k and $\bar{h}_k$ change at each packet arrival, the summations in (1) must be always recomputed. This can be avoided by decomposing (1) using basic trigonometric properties. We define

$$\Phi_k := \sum_{j=0}^{N-1} h_{k-j} \cos(\omega t_{k-j}) - \bar{h}_k \sum_{j=0}^{N-1} \cos(\omega t_{k-j}); \qquad \Phi_k^* := \sum_{j=0}^{N-1} \cos(2\omega t_{k-j});$$

$$\Gamma_k := \sum_{j=0}^{N-1} h_{k-j} \sin(\omega t_{k-j}) - \bar{h}_k \sum_{j=0}^{N-1} \sin(\omega t_{k-j}); \qquad \Gamma_k^* := \sum_{j=0}^{N-1} \sin(2\omega t_{k-j}).$$

Note that Φ_k, Γ_k, Φ_k^* and Γ_k^* are simple summations that can be updated at each packet arrival in a similar way as done for (2)-(3). We can then rewrite the Lomb periodogram as follows:

$$P_k^N(\omega) = \frac{1}{\sigma_k^2} \left\{ \frac{\left[\Phi_k \cos(\omega\tau_k) + \Gamma_k \sin(\omega\tau_k)\right]^2}{N + \Phi_k^* \cos(2\omega\tau_k) + \Gamma_k^* \sin(2\omega\tau_k)} + \frac{\left[\Gamma_k \cos(\omega\tau_k) - \Phi_k \sin(\omega\tau_k)\right]^2}{N - \Phi_k^* \cos(2\omega\tau_k) - \Gamma_k^* \sin(2\omega\tau_k)} \right\}. \quad (4)$$

When a new packet k arrives, the values of $\bar{h}_k$, τ_k, σ_k^2, Φ_k, Γ_k, Φ_k^* and Γ_k^* are immediately updated, and the periodogram is recomputed accordingly. The number of operations done at each packet arrival is $O(N)$, since updating $\bar{h}_k$, τ_k, σ_k^2, Φ_k, Γ_k, Φ_k^* and Γ_k^* takes $O(1)$ and we have $2N$ angular frequencies ω.

5.2 Fundamental Frequency Extraction

The outcome of the computation of the Lomb periodogram is usually *noisy*, with many local maxima and a global maximum that not always corresponds to the fundamental frequency f_0: sometimes the global maximum corresponds to a frequency multiple of the fundamental one (see for instance Fig. 5 in [18]). The operations performed *at each packet arrival* by the Fundamental Frequency Extraction module on the updated periodogram can be summarized as follows:

Periodogram smoothing. A basic low-pass FIR filter is applied to the sequence that composes the Lomb Periodogram: in particular, the filter is a moving average filter of order three. We have tested different orders for the filter obtaining similar results, as long as the order is not too high (e.g., greater than 8), since the smoothing effect decreases the dynamic range.

Peak detection. We consider the value at frequency f_k to be a peak if it is greater than the values at frequencies f_{k+1} and f_{k-1}; the detected peaks are put in a "peak list."

Peak pruning and ordering. We remove from the list the entries whose values are not the top W largest peaks, with $W = 10$. The parameter W is configurable: in all our experiments, the number of peaks in the spectrum was between 15 and 25, but only a subset of them were sufficiently strong (one order of magnitude greater than the white noise). By considering the average characteristics of the TCP flows observed in [6], we can conclude that this value can be applied in general. From the peak list, we remove also some frequency values considering the following boundary conditions: the RTT should be greater than 2 ms and less than 500 ms. In [6], it has been observed that 70% of the flows have RTT smaller than 500 ms. The list is then ordered by frequency, smallest first.

Multiple frequency search. Starting from the smallest frequency in the list, we evaluate if the other frequencies in the list can be a multiple of the considered frequency. If we find at least two multiples, this step ends and returns the estimated fundamental frequency f_0. Otherwise it goes on with the following value in the list. If no fundamental frequency is found, this step returns a *NULL* value.

Comparison. We compare the output with the average of the previous estimations, $\overline{f}_0$. If the current estimation is not *NULL*, we consider the ratio between the current estimation and the average of the previous estimations, $r = f_0/\overline{f}_0$. In [6], the authors show that, in 95% of the flows, the ratio between the maximum and the minimum RTT is less than 3/2. Thus, if $2/3 < r < 3/2$, the algorithm returns f_0 and terminates. Otherwise, it returns $\overline{f}_0$.

The output of this module is then the estimated fundamental frequency whose inverse is the estimated RTT.

6 Numerical Results

We validated our methodology with experiments both in a controlled environment and over the Internet. We focus on a single long-lived *scp* transfer between a source and a destination and we collect traces with *tcpdump* at two points: (i) at the source, where we capture the traffic in both directions, in order to have the packets and their acknowledgments; (ii) at a measuring point between the source and the destination, where we record the packets in one direction. In addition to the observed long-lived *scp* transfer, there are different flows on the path between the source and the destination: In the controlled testbed, the cross-traffic flows are generated by *scp* transfers.

The traces collected at the source are post-processed computing the instantaneous RTT and the smoothed RTT using an exponentially weighted moving average (EWMA) algorithm whose parameter α is set to 1/8 (see RFC 2988). Note that the RTT is updated for every packet without considering retransmitted packets.

The traces collected at the measuring point are analyzed using our solution. For the spectral analysis, the length of the sequence considered is $N = 256$. The spectral analysis, along with the fundamental frequency extraction, gives the estimation of the instantaneous RTT. Similarly to TCP, we compute a smoothed

estimated RTT through an EWMA algorithm with parameter $\alpha = 1/8$. Hereinafter, we will use the general term "RTT" to refer to the smoothed value of the RTT (at the source and estimated).

The file size used for the testbed and for the Internet experiments is 120 MBytes. From the samples of the RTTs we create the empirical Cumulative Distribution Function (CDF). From this empirical CDF we derive any performance index of interest, such as the mean RTT and the variance of the RTT.

In order to evaluate the accuracy of the estimation process over time, we consider the error between the estimate and the real RTT, these being the average values observed over small, non overlapping, intervals (of 5 seconds, in practice). We compute: error $= \frac{\text{mean estimated RTT} - \text{mean RTT at the source}}{\text{mean RTT at the source}}$ and build the corresponding empirical CDF.

6.1 Controlled Environment

We first considered a testbed using Dell Precision 380 workstations running Fedora Linux version 10 (kernel 2.6.27). The topology is shown in Fig. 3: all machines are directly connected as shown (using Ethernet cables) and there is no outside traffic. Given that the propagation and processing delay are very small, we introduce random delays between the machines. The link between the source of the traffic and the machine *cross* has a delay uniformly distributed on $[50 - d_1, 50 + d_1]$ ms, where d_1 is set to either 5 or 40 ms. The link between the machines *cross* and *btlnk* has a delay uniformly distributed on $[25 - d_2, 25 + d_2]$ ms, where d_2 is set to either 5 or 20 ms.

All the links have a capacity of 100 Mbps (fast Ethernet) except the link between the machines *btlnk* and *dest* which is capped to 10 Mbps in order to act as the bottleneck link. The random delays and the capacity limitations are implemented using the Netem kernel module. Throughout the experiments we explicitly configured the sender to use Reno TCP and not Cubic (the default option for kernel 2.6.27).

The main flow we consider is between machines *source* and *dest*. We add additional cross traffic directed to *dest*, starting from *source* and from *cross*. We have a measuring point of the packets' arrivals at the machine *btlnk*. Figure 4 summarizes the experiments done.

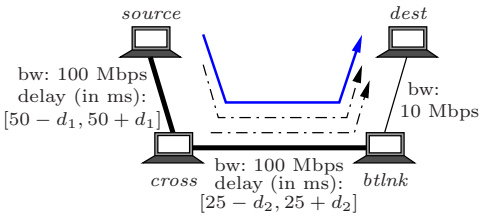

Fig. 3. Scheme of the testbed: the continuous line represents the observed flow, dashed lines represent the cross traffic.

experim. ID	# cross flows *source* → *dest*	# cross flows *cross* → *dest*	d_1	d_2
exp1	0	12	5	5
exp2	2	6	5	5
exp3	3	9	5	5
exp4	3	9	40	20

Fig. 4. Configuration of the testbed experiments, with the different cross flows and delays.

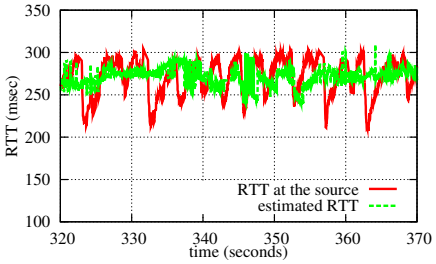

Fig. 5. Example of the RTT measured at the source and the estimated RTT (exp2).

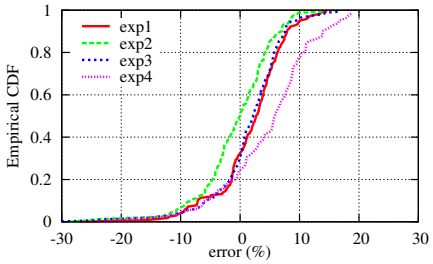

Fig. 6. Empirical CDF of the error between true and estimated RTTs (testbed).

Figure 5 shows an example of the output of one experiment (exp2). For each packet, we compute the RTT at the source and its estimation at the measuring point. We have found that the estimated RTT is of the same order of magnitude as the real RTT, even though it seems to be less variable than the RTT at the source. This is due to the averaging effect of the spectral analysis that uses the last observed 256 packets (which include 5-8 flights) for the spectrum estimation.

While this representation shows that the estimation and the real RTT are close, we need to characterize in detail the performance of the algorithm. In order to minimize the effects of the lag time between the RTT samples collected at the source and at the measuring point, we consider the empirical cumulative distribution function (CDF) of the RTTs built from the samples and report the corresponding mean and variance in Table 1 (cf. columns 3–6). While the real RTT has more variability, the estimated RTT is more stable due to the averaging effect of the spectral analysis.

Another performance index we are interested in is the error in the estimation: as Fig. 6 shows, in experiments exp1–exp3, the estimate lies within 10% of the true value with probability 95%. In the worst case, our solution is able to accurately estimate the RTT with an error in $[-10\%, 10\%]$ with probability equal to 75%, and with an error in $[-20\%, 20\%]$ with probability equal to 99%.

The results given by our methodology show a higher accuracy with respect to the results obtained in previous studies (e.g. [6]): our results have been obtained online observing the traffic in one direction, unlike the results of [6] which require to collect traffic in both directions and analyze it offline.

Table 1. Mean and variance of the RTT (real and estimated)

		Testbed experiments				Internet experiments	
		exp1	exp2	exp3	exp4	peak	off-peak
Real RTT	mean	0.2795	0.2760	0.2782	0.2752	0.0389	0.0448
(in s)	variance	$1.85\ 10^{-3}$	$1.84\ 10^{-3}$	$1.62\ 10^{-3}$	$2.70\ 10^{-3}$	$1.72\ 10^{-5}$	$2.40\ 10^{-6}$
Estimated	mean	0.2814	0.2706	0.2792	0.2875	0.0422	0.0446
RTT (in s)	variance	$9.46\ 10^{-5}$	$1.41\ 10^{-4}$	$3.57\ 10^{-6}$	$5.43\ 10^{-4}$	$2.40\ 10^{-6}$	$2.94\ 10^{-6}$

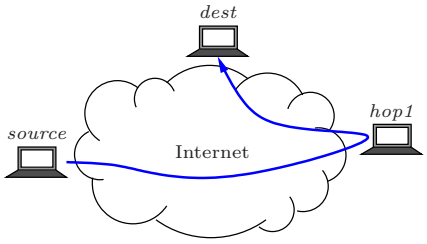

Fig. 7. Scheme of the Internet tests.

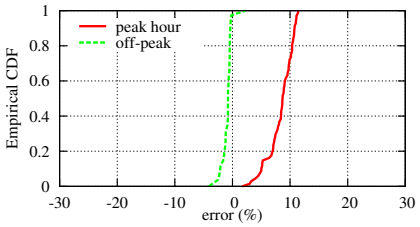

Fig. 8. Empirical CDF of the error between the RTT measured at the source and estimated (Internet experiments).

6.2 Internet Experiments

In this section, we show the results of experiments using three machines connected through Internet (Fig. 7). The source is at INRIA Sophia Antipolis (France), the intermediate machine (*hop1*) is at University of Trento (Italy) while the destination is at Eurecom (France). We created two SSH tunnels, one between *source* and *hop1*, and one between *hop1* and the destination *dest*. The traffic generated from the source passes through the tunnel at *hop1*, where packets interarrivals are measured, and reaches the destination.

Columns 7–8 of Table 1 show the mean and the variance of the RTTs (real and estimated) for two different experiments: the first, during a peak hour (Monday, May 4th, 2009, 10 AM, CEST) and the second, during an off-peak hour (Sunday, May 3rd, 2009, 12 PM, CEST). The high variability of the peak hour results in a higher error, that is not smoothed by considering the overall mean. The reason can be understood looking at Fig. 5: the estimation algorithm often does not detect sudden decreases of the RTT. When the RTT drops, we obtain large positive error values, while there are no large negative error values to compensate for the large positive ones.

As seen in Fig. 8 which shows the empirical CDF of the error, the estimate lies within 10% of the true value with probability equal to 99%.

7 Open Issues and Conclusions

In this paper we have presented a methodology, based on spectrum analysis, for the passive online estimation of the RTT of a long-lived TCP connection, using one-way traffic. The estimation of the RTT represents a basic building block in the framework of "Semantic Networking," where flow-aware routers can implement novel AQM techniques in order to control the traffic.

We validate our solution through measurements in a controlled testbed and over the Internet showing a good accuracy. We plan to extend the evaluation using different scenarios: for instance, it is interesting to understand the variation in estimation accuracy as the RTT varies due to changes in the routing or in the number of flows sharing a link.

Since the method is based only on the arrival pattern, it is clear that if the arrival pattern is strongly modified inside the network, the method may

not be accurate. For instance, if we observe a flow after it has passed through a bottleneck, the arrival pattern may become a continuous stream of packets. Should this happen, the method may not be applicable, however one would still be able to estimate the rate of a connection through a moving average estimator for instance. In future evaluations, we plan to investigate in detail the scenarios where the methodology may not be accurate, in order to provide a comprehensive view of the benefits of our solution.

References

1. Collange, D., Costeux, J.: Passive estimation of quality of experience. Journal of Universal Computer Science **14**(5) (2008) 625–641
2. Noirie, L., Dotaro, E., Carofiglio, G., Dupas, A., Pecci, P., Popa, D., Post, G.: Self-* features for semantic networking. In: Proc. of FITraMEn. (Dec. 2008)
3. Avrachenkov, K., Ayesta, U., Brown, P., Nyberg, E.: Differentiation between short and long TCP flows: predictability of the response time. In: Proc. of IEEE INFOCOM, Hong Kong (Mar. 2004)
4. Rai, I., Biersack, E., Urvoy-Keller, G.: Size-based scheduling to improve the performance of short TCP flows. IEEE Network **19**(1) (2005) 12–17
5. Kortebi, A., Muscariello, L., Oueslati, S., Roberts, J.: Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In: Proc. of ACM SIGMETRICS, Banff, Alberta, Canada (June 2005)
6. Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., Towsley, D.: Inferring TCP connection characteristics through passive measurements. In: Proc. of IEEE INFOCOM, Hong Kong (Mar. 2004)
7. But, J., Keller, U., Kennedy, D., Armitage, G.: Passive TCP stream estimation of RTT and jitter parameters. In: Proc. of IEEE CLCN. (Nov. 2005)
8. Zhang, Y., Breslau, L., Paxson, V., Shenker, S.: On the characteristics and origins of Internet flow rates. In: Proc. of ACM SIGCOMM. (Aug. 2002)
9. Lance, R., Frommer, I., Hunt, B., Ott, E., Yorke, J., Harder, E.: Round-trip time inference via passive monitoring. ACM SIGMETRICS PER **33**(3) (2005) 32–38
10. Veal, B., Li, K., Lowenthal, D.: New methods for passive estimation of TCP round-trip times. In: Proc. of PAM, Boston, MA, USA (Apr. 2005)
11. Jiang, H., Dovrolis, C.: Passive estimation of TCP round-trip times. Computer Communication Review **32**(3) (2002) 75–88
12. Qi, Y., Minka, T., Picard, R.: Bayesian spectrum estimation of unevenly sampled nonstationary data. In: Proc. of ICASSP, Orlando, FL, USA (May 2002)
13. Brown, P.: Resource sharing of TCP connections with different round trip times. In: Proc. of IEEE INFOCOM, Tel Aviv, Israel (Mar. 2000)
14. Altman, E., Jimenez, T., Nunez Queija, R.: Analysis of two competing TCP/IP connections. Performance Evaluation **49**(1-4) (2002) 43–55
15. Stoica, P., Moses, R.: Introduction to spectral analysis. Prentice-Hall (1997)
16. Stoica, P., Sandgren, N.: Spectral analysis of irregularly-sampled data: Paralleling the regularly-sampled data approaches. Digital Signal Processing **16**(6) (2006) 712–734
17. Scargle, J.: Statistical aspects of spectral analysis of unevenly spaced data. Journal of Astrophysics **263** (1982) 835–853
18. Carra, D., Avrachenkov, K., Alouf, S., Blanc, A., Nain, P., Post, G.: Passive online RTT estimation for flow-aware routers using one-way traffic. Research Report RR-7124, INRIA (Nov. 2009)