

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

Damiano Carra¹, Konstantin Avrachenkov², Sara Alouf²,
Alberto Blanc², Philippe Nain², and Georg Post³

¹ University of Verona, Verona, Italy,
`damiano.carra@univr.it`

² INRIA Sophia Antipolis Méditerranée, 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 the new generation high speed routers, it becomes possible to improve the Quality of Service, the Quality of Experience for users and the network efficiency for ISPs with the help of “flow-aware” traffic management. An example of the “flow-aware” traffic management is the Alcatel-Lucent framework “Semantic Networking.” In the framework “Semantic Networking”, short-lived and long-lived TCP flows are treated differently. Short-lived flows are processed with high priority and long-lived flows are controlled in a “flow-aware” fashion. To control efficiently the long-lived flows, one needs to know an estimation of the Round Trip Time (RTT). In the present work, we provide an online RTT estimation algorithm which is passive and can deal with a one-way traffic. 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 the Internet demonstrate high accuracy of the proposed estimator. The results show that, 75% (resp. 99%) of the time, the RTT estimation is within 10% (resp. 20%) of the RTT at the source.

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 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” [2].

Let us outline several building blocks of “Semantic Networking”. An interested reader can find a more detailed description of the framework in [2]. First,

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

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, in the “Semantic Networking” framework, the short-lived flows are served with high priority. The efficiency of such approach has been validated 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 performance and its efficiency and providing customers with high QoS and QoE [2].

One of the most important notions for a TCP flow is its aggressiveness or how fast a TCP connection increases its sending rate. Most deployed TCP versions are based on congestion window which limits a number of packets that can be sent during one Round Trip Time (RTT). This implies that the RTT is one of the principal parameters which 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 present an algorithm 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 growth rate of a flow 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. One cannot assume that the monitoring point sees packets in both directions as forward and reverse paths may be different. Furthermore, 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 work makes the following important contributions. We design a method for RTT estimation based on the traffic observed only in one direction. The method relies on spectral analysis of the signal built considering the inter-packet

times. The self-clocking mechanism of TCP introduces *periodic* components into the arrival times of packets. We use spectral analysis to extract such periodic components. There are different tools available for spectrum estimation, mainly for regularly sampled data. The samples collected at the router are irregularly spaced, thus the choice is essentially limited, for either technical or computational reasons, to the *Lomb periodogram*. The traditional implementation of any spectral analysis tool considers an offline approach. Our main contribution is the development of an online version of the Lomb periodogram. At each packet arrival, the estimation of the spectrum is updated with $O(N)$ operations, where N is the length of the initial sequence collected (the number of frequencies in the spectrum is $2N$).

Another contribution lies in the fundamental frequency extraction algorithm: once the estimation of the spectrum is done, we need to extract the fundamental frequency that corresponds to the inverse of the RTT. This is done using a pattern matching technique that looks for the greatest common divisor of a subset of frequencies.

We tested our estimation algorithm both on a controlled testbed and on the Internet. The 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, i.e., packets of only one direction instead of both directions.

2 Related Work

The RTT estimation has been the subject of many studies. The aim of such 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 in [7] is based on the reconstruction of the TCP congestion window values, which requires to maintain a “replica” of the TCP sender’s state. The paper [8] extends the work in [7], thus it maintains similar computational complexity.

Another example is the work in [9], which proposes 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.

The paper [10] makes 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 [11], a method based on the TCP timestamp option is proposed. 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 [12], the RTT is estimated using the first packets of a connection, thus it is not generally applicable to the continuous monitoring of a connection.

In [13], the authors 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 from a computational point of view: 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 Sect. 5), thus, if we consider N consecutive samples, the complexity becomes $O(N^2)$.

In summary, all the previous works have some limitations that make them not applicable to the context we are considering: Since we are looking for real-time (online) estimation of the RTT inside a router (see Sect. 3 for details), the proposed methods cannot be used for their computational complexity or the low accuracy. Note also that some methodologies (works in [7, 8, 11]) use traffic captured in both directions, and correlate packets with their acknowledgments. Our methodology uses packets of only one direction (thus we use less information) yet obtaining a more accurate estimation of the RTT.

3 Motivations

The methods proposed so far for estimating the RTT have focused on the evaluation of the characteristics of the TCP connections for monitoring purposes: the interior monitoring point collects packets from a router interface (e.g. from both directions of a network card) and the traces are analyzed offline. In this paper, we have a completely different point of view. We consider the estimation of the RTT as a building block for actively controlling the traffic inside the network. We start considering a flow-aware networking approach, where routers are able to manage flows rather than simply packets. This in turn opens the possibility of exploring novel Active Queue Management (AQM) techniques in order to efficiently manage the router resources.

3.1 Flow-Aware Networking

Customers' increasing demand for better quality of service (QoS) and quality of experience (QoE) have increased the interest in techniques for actively managing and controlling the traffic inside the network. One such approach calls for treating each flow individually, where by flow we mean a connection identified by the classical five-tuple of protocol ID, source and destination addresses and ports. Considering a flow as the basic entity to deal with, rather than single

packets, has a beneficial impact on scalability, flexibility and operational complexity. For a detailed view of the flow-aware networking approach the interested reader is referred to [2]. Here we consider the practical implementation of such an approach, and in particular the structure of a *flow-aware* router.

The basic architecture can be summarized as follows. There is a single physical buffer that is divided into a number of virtual queues, one of which has high priority and the others have low priority. Flows are sorted into two classes, whether they are short-lived or long-lived flows. The basic idea is to put all the short-lived flows in the high priority queue, and to assign a low priority queue to each of the long-lived flows. When a new flow arrives, it is initially treated as short-lived. If the cumulative number of packets of a given flow reaches a threshold, this flow is then considered as an elephant, i.e., its packets are enqueued in a low priority queue associated to this flow.⁴ Low priority queues are served in a round robin fashion or a similar scheduling policy.

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. Moreover, since they carry 90% of the total traffic, by controlling elephants, it is possible to manage accurately the available resources.

3.2 Novel AQM Policies

Flows for large data transfers use 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 [14, 15].

With flow-aware routers, where each elephant uses a different queue, there is the possibility of new flow-aware queue management 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

⁴ There exist a number of different mechanisms to “promote” a flow to elephant, mainly probabilistic methods. For instance, each arriving packet is compared with one or more packets randomly picked from the buffer: if the packets belong to the same flow, then this flow can be considered with high probability an elephant.

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 (e.g. due to dropped packets).

If the information about the rate and its growth is made available at the router, it is then possible to design novel AQM policies that take into account this information. For instance, consider the scenario with two competing flows that are saturating the available bandwidth, i.e., the buffer occupation starts increasing. Assume that both flows have, at a given point in time, the same instantaneous rate, but different RTTs (which means the instantaneous flow rates have different *growth* rates). The router may decide to mark/drop in advance packets belonging to one or both flows, before the buffer becomes full. By knowing the RTT, it is possible to choose, for instance, the more aggressive flow, i.e., the flow with the smaller RTT.

While the study of novel 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. Note that previously proposed schemes (e.g. [7, 8]) cannot be used in such a context, since they assume to have access to the traffic in both directions. Such solutions would be technologically and computationally infeasible inside a router, since they require the router itself not only to correlate packets with their acknowledgments, but also to do so in real-time, in order to make the information available, in real-time, for correctly applying the AQM policy.

In summary, while the RTT estimation is a well covered area of research, no techniques are available for the continuous, real-time and passive estimation of the RTT, using the traffic of one direction.

4 A methodology Based on Spectral Analysis

The self-clocking mechanism of TCP introduces *periodic* components into the arrival times of packets. 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 [16]. 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 k th 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 k th 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 solutions for irregularly sampled data have been found to be impractical [17]:

The choice of spectral analysis for irregularly sampled data is essentially limited, for either technical or computational reasons, to the *periodogram*.

The periodogram is simply the Discrete Fourier Transform (DFT) of the finite sequence. In case of irregularly sampled data, the periodogram is computed using the Lomb-Scargle method [18] and it is generally referred to as the Lomb periodogram. The periodogram has well-known limitations [16]: (i) it is not possible to distinguish two sinusoidal components which are too close (low resolution), and (ii) the variance of the estimation error at a given frequency does not decrease as the number of samples used in the computation increases (non-consistent estimator). Despite these limitations, the periodogram remains an efficient solution in case of well-separated sinusoids in white noise and medium to large dynamic range [17]. The dynamic range is defined as the ratio between the maximum and the minimum power values, usually measured in decibel. A dynamic range between 20 dB and 30 dB (i.e., between two to three orders of magnitude between the maximum and the minimum power) is said to be “medium”; if it is greater than 30 dB, then it is said to be “large.”

In our case, the signal h_k , the inter-arrival times of the packets, either contains a single strong sinusoidal component, with large dynamic range, due to the RTT, or it does not contain any periodicity (consider the case of a flow that has reached the bandwidth-delay product, where packets form a continuous stream). In the latter case, no method based on spectral analysis is able to extract the RTT: only methods based on the analysis of the traffic in both directions would work—a case that we cannot consider since a flow-aware router has access in real-time only to one direction. Thanks to the characteristics of the signal h_k , the periodogram does represent a good estimator.

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. While this approach can be suitable for traditional analysis of signals, it cannot be applied in case of RTT estimation by a flow-aware router. It may take several RTTs before being able to collect N samples (typical values of N are 256, 512 or 1024). If the RTT varies, the router is not able to correctly follow these variations, leading to a wrong estimation of the flow rates, with negative consequences on the efficiency of the AQM policies.

It is extremely important to have a spectrum estimation methodology that works in real time, i.e., that is able to update the estimations 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 [17]) are based on complex optimized algorithms, and cannot be translated into corresponding online versions.

In summary, among the different methods for spectral analysis, the periodogram represents the only choice for two reasons:

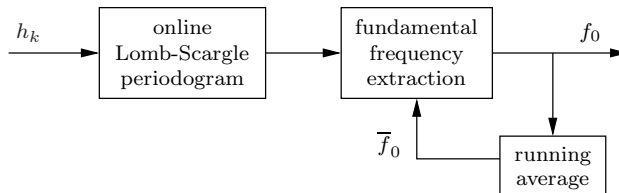


Fig. 1. The RTT estimation process: after the update of the spectrum estimation, the fundamental frequency is extracted. The estimation of the RTT is the inverse of the fundamental frequency f_0 .

- the structure of the signal h_k (with a single strong sinusoidal component) makes the periodogram a sufficiently good estimator, overcoming its limitations;
- the periodogram allows for the creation of an efficient online version of the algorithm, something not possible in the case of irregularly sampled data with other proposed spectral analysis approaches.

The spectral analysis represents the basic building block of our RTT estimation method: the whole estimation process is composed of different steps that we describe in detail in the following section.

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. Figure 1 shows the functional blocks of the estimation process.

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 $f_0, 2f_0, 3f_0, \dots$, 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, 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 for other frequencies in the list.

5.1 Online Lomb Periodogram

The Lomb-Scargle periodogram [18] 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 k th 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 :

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

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

and where τ_k is the solution of

$$\tan(2\omega\tau_k) = \frac{\sum_{i=0}^{N-1} \sin 2\omega t_{k-i}}{\sum_{i=0}^{N-1} \cos 2\omega t_{k-i}}. \quad (4)$$

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., $f_k^{\min} = 1/(t_k - t_{k-N+1})$. 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 $\omega_i = 2\pi f_k^{\min} + i\Delta_\omega$, $i = 0, \dots, 2N-1$.

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

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

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:

$$P_k^N(\omega) = \frac{1}{\sigma_k^2} \left\{ \frac{[\Phi_k \cos(\omega\tau_k) + \Gamma_k \sin(\omega\tau_k)]^2}{N + \Phi_k^* \cos(2\omega\tau_k) + \Gamma_k^* \sin(2\omega\tau_k)} + \frac{[\Gamma_k \cos(\omega\tau_k) - \Phi_k \sin(\omega\tau_k)]^2}{N - \Phi_k^* \cos(2\omega\tau_k) - \Gamma_k^* \sin(2\omega\tau_k)} \right\}. \quad (5)$$

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. The operations performed *at each packet arrival* by the Fundamental Frequency Extraction module on the updated periodogram are summarized in the following.

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 was sufficiently strong (one order of magnitude greater than the white noise). By considering the average characteristics of the TCP flows observed in [7], 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 [7], 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, \bar{f}_0 . If the current estimation is not *NULL*, we consider the ratio between

the current estimation and the average of the previous estimations, $r = f_0/\bar{f}_0$. In [7], 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 \bar{f}_0 .

The output of this module is then the estimated fundamental frequency from which it is possible to derive 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 *tcpdumb* 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 (SRTT) using a moving average, without considering retransmitted packets. The parameter α for the exponential weighted moving average (EWMA) algorithm (see RFC 2988) is set to 1/8. Note that the RTT is updated for every packet.

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 RTT through an EWMA algorithm with parameter $\alpha = 1/8$. Hereinafter, we will take as reference the values of the SRTTs (at the source and estimated), using the general term “RTT.”

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 distribution we can 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 estimation and the real RTT. To overcome the issues highlighted in Sect. ??, we consider the means of the RTTs (estimated and at the source) taken over short, non overlapped, intervals. In practice, we compute the mean RTTs every 5 seconds and we compute the error of the estimate defined as

$$\text{error} = \frac{\text{mean estimated RTT} - \text{mean RTT at the source}}{\text{mean RTT at the source}}.$$

From the error, we build the corresponding empirical CDF.

6.1 Controlled Environment

We first considered a testbed set up at INRIA using Dell Precision 380 workstations running Fedora Linux version 10 (kernel 2.6.27). The topology is shown in Fig. 2: 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.

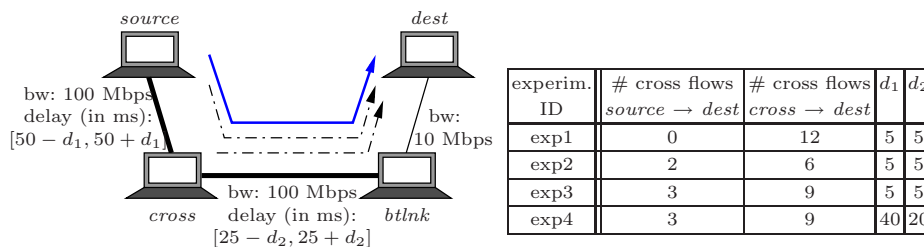


Fig. 2. Scheme of the testbed: the continuous line represents the observed flow, and dashed lines represent the cross traffic. **Fig. 3.** Configuration of the testbed experiments, with the different cross flows and delays.

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*. Table 3 summarizes the experiments done.

Figure 4 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. As explained in Sect. ??, 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

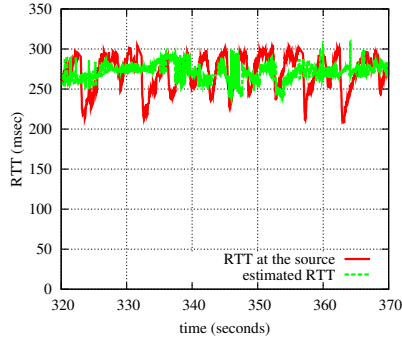


Fig. 4. Example of the RTT measured at the source and the estimated RTT (exp2).

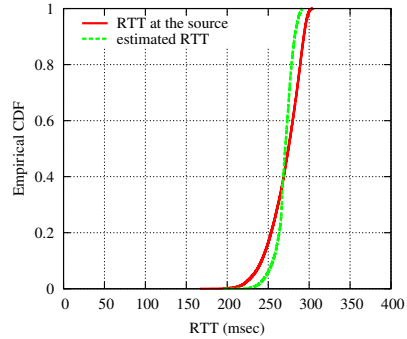


Fig. 5. Empirical CDF of the RTT measured at the source and the estimated RTT (exp2).

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

exp. ID	RTT at the source (s)		estimated RTT (s)	
	mean	variance	mean	variance
exp1	0.2795	$1.8476 \cdot 10^{-3}$	0.2814	$9.4612 \cdot 10^{-5}$
exp2	0.2760	$1.8431 \cdot 10^{-3}$	0.2706	$1.4061 \cdot 10^{-4}$
exp3	0.2782	$1.6236 \cdot 10^{-3}$	0.2792	$3.5682 \cdot 10^{-6}$
exp4	0.2752	$2.6997 \cdot 10^{-3}$	0.2875	$5.4265 \cdot 10^{-4}$

cumulative distribution function (CDF) of the RTTs built from the samples. Figure 5 compares the two empirical CDFs (exp2): both CDFs share approximately the same support, i.e., the estimation lies in the same interval of the real RTT.

In Table 1 we report the mean and the variance obtained from the empirical CDFs of the RTTs (real and estimated). 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 estimation lies within 10% of the estimated value with probability 95%. In the worst case, our solution is able to accurately estimate the RTT with an error within $[-10\%, 10\%]$ with probability equal to 75%, and with an error within $[-20\%, 20\%]$ 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. [7]): our results have been obtained observing the traffic of one direction, unlike the results of [7] which require to collect traffic in both directions. In summary, we are able to find an accurate estimation of the RTT using less information w.r.t. previous works, and most noticeably, our method works online.

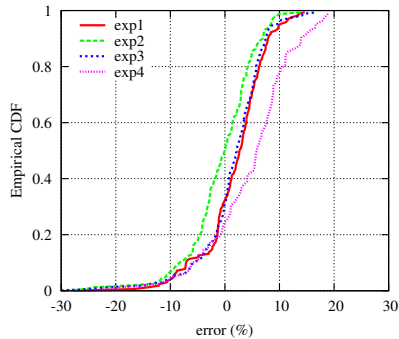


Fig. 6. Empirical CDF of the error between the RTT measured at the source and the estimated RTT (testbed).

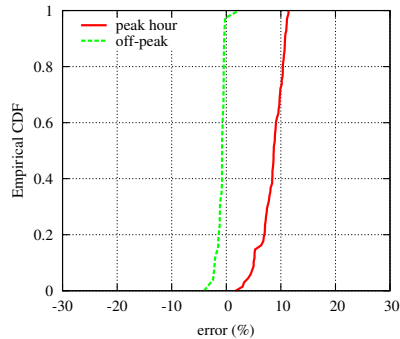


Fig. 7. Empirical CDF of the error between the RTT measured at the source and estimated (Internet experiments).

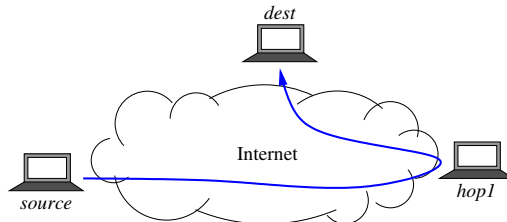


Fig. 8. Scheme of the Internet tests.

6.2 Internet Experiments

In this section, we show the results of experiments using three machines connected through Internet (Fig. 8). 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.

Table 2 shows 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. 4: the estimation algorithm often does not detect sudden decreases of the RTT. When the drops in the RTT occur, we obtain large positive error values, while there are not large negative error values to compensate for the large positive ones.

Table 2. Mean and variance of the RTT (real and estimated): Internet experiments.

exp. ID	RTT at the source (s)		estimated RTT (s)	
	mean	variance	mean	variance
peak	0.0389	$1.7194 \cdot 10^{-5}$	0.0422	$2.4034 \cdot 10^{-6}$
off	0.0448	$2.4014 \cdot 10^{-6}$	0.0446	$2.9448 \cdot 10^{-6}$

Figure 7 shows the whole empirical CDFs of the error. As for the testbed experiments, the estimation lies within 10% of the estimated 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. Since we considered only a limited set of scenarios, we plan to extend the evaluation on different scenarios: for instance, it is interesting to understand the variation in estimation accuracy as the RTT changes due to changes in the number of connections sharing a link or in the routing.

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. D. Collange and J.L. Costeux, “Passive Estimation of Quality of Experience,” *Journal of Universal Computer Science*, vol. 14, no. 5, pp. 625–641, 2008.
2. L. Noirie, E. Dotaro, G. Carofiglio, A. Dupas, P. Pecci, D. Popa, and G. Post, “Self-* Features for Semantic Networking,” in *Proc. of FITraMEN*, Porto, Portugal, Dec. 2008.
3. K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg, “Differentiation between short and long TCP flows: predictability of the response time,” in *Proc. of IEEE INFOCOM*, Hong Kong, Mar. 2004.

4. I.A. Rai, E.W. Biersack, and G. Urvoy-Keller, "Size-based scheduling to improve the performance of short TCP flows," *IEEE Network*, vol. 19, no. 1, pp. 12–17, 2005.
5. A. Kortebe, L. Muscariello, S. Oueslati, and J. Roberts, "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. D. Carra, K. Avrachenkov, S. Alouf, P. Nain, and G. Post, "Method for Estimating a Round Trip Time of a Packet Flow," patent no. 2416, filing no. 09305207.4, filing date Mar. 3, 2009.
7. S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP connection characteristics through passive measurements," in *Proc. of IEEE INFOCOM*, Hong Kong, Mar. 2004.
8. J. But, U. Keller, D. Kennedy, and G. Armitage, "Passive TCP Stream Estimation of RTT and Jitter Parameters," in *Proc. of IEEE 30th Conference on Local Computer Networks*, Sydney, Australia, Nov. 2005.
9. Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates," in *Proc. of ACM SIGCOMM*, Pittsburgh, PA, USA, Aug. 2002.
10. R. Lance, I. Frommer, B. Hunt, E. Ott, J.A. Yorke, and E. Harder, "Round-trip time inference via passive monitoring," *ACM Sigmetrics Performance Evaluation Review*, vol. 33, no. 3, pp. 32–38, Dec. 2005.
11. B. Veal, K. Li, and D. Lowenthal, "New Methods for Passive Estimation of TCP Round-Trip Times," in *Proc. of PAM*, Boston, MA, USA, Apr. 2005.
12. H. Jiang and C. Dovrolis, "Passive estimation of TCP round-trip times," *Computer Communication Review*, vol. 32, no. 3, pp. 75–88, 2002.
13. Y. Qi, T.P. Minka, and R.W. Picard, "Bayesian spectrum estimation of unevenly sampled nonstationary data," in *Proc. of ICASSP*, Orlando, FL, USA, May 2002.
14. P. Brown, "Resource sharing of TCP connections with different round trip times," in *Proc. of IEEE INFOCOM*, Tel Aviv, Israel, Mar. 2000.
15. E. Altman, T. Jimenez, and R. Nunez Queija, "Analysis of two competing TCP/IP connections," *Performance Evaluation*, vol. 49, no. 1-4, pp. 43–55, 2002.
16. P. Stoica and R.L. Moses, "Introduction to spectral analysis," Prentice-Hall, 1997.
17. P. Stoica and N. Sandgren, "Spectral Analysis of Irregularly-Sampled Data: Paralleling the Regularly-Sampled Data Approaches," *Digital Signal Processing*, vol. 16, no. 6, pp. 712–734, 2006.
18. J.D. Scargle, "Statistical aspects of spectral analysis of unevenly spaced data," *Journal of Astrophysics*, vol. 263, pp. 835–853, 1982.