Estimation of the Parameters of Token-buckets in Multi-hop Environments

Javier Ramos[a], David Muelas[a], Jorge E. López de Vergara[a], Javier Aracil[a]

[a] *High Performance Computing and Networking Research Group, Dpto. Tecnología Electrónica y de las Comunicaciones, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Fco. Tomás y Valiente, 11, 28049 Madrid, Spain*

Abstract

Bandwidth verification in shaping scenarios receives much attention of both operators and clients because of its impact on Quality of Service (QoS). As a result, measuring shapers' parameters, namely the Committed Information Rate (CIR), Peak Information Rate (PIR) and Maximum Burst Size (MBS), is a relevant issue when it comes to assess QoS. In this paper, we present a novel algorithm, *TBCheck*, which serves to accurately measure such parameters with minimal intrusiveness. These measurements are the cornerstone for the validation of Service Level Agreements (SLA) with multiple shaping elements along an end-to-end path. As a further outcome of this measurement method, we define a formal taxonomy of multi-hop shaping scenarios. A thorough performance evaluation covering the latter taxonomy shows the advantages of *TBCheck* compared to other tools in the state of the art, yielding more accurate results even in the presence of cross-traffic. Additionally, our findings show that MBS estimation is unfeasible when the link load is high, regardless the measurement technique, because the token-bucket will always be empty. Consequently, we propose an estimation policy which

☆Manuscript received 19 June 2017. Revised 10 November 2017. Revised 7 March 2018. Accepted 3 April 2018

☆☆The final publication is available at Elsevier via `https://doi.org/10.1016/j.comnet.2018.04.003`, to be published in *Computer Networks*.

Email addresses: `javier.ramos@uam.es` (Javier Ramos), `dav.muelas@uam.es` (David Muelas), `jorge.lopez_vergara@uam.es` (Jorge E. López de Vergara), `javier.aracil@uam.es` (Javier Aracil)

maximizes the accuracy by measuring CIR and PIR during busy hours and MBS during off-peak hours.

Keywords: Service-level agreements; Quality-of-service management; Network and systems monitoring and measurements; Token-Bucket

1. Introduction

Nowadays, core and access networks provide higher bandwidth, together with smaller delays and jitters. As a result, many real-time services (such as Voice over Internet Protocol (VoIP), video on demand, teleconferencing, HD/3D television, remote desktop —e.g., *Citrix*—, etc.), which have strong requirements in terms of Quality of Service (QoS), are widespread to a huge variety of end users, both residential and corporate. From the network monitoring standpoint, the massive usage of these interactive services is posing significant challenges. In the past, measuring the average utilization of links was deemed sufficient, while current trends require further consideration of QoS parameters.

Additionally, large corporations such as banks are increasingly trusting Multi-Protocol Label Switching (MPLS) or Virtual Private LAN Service (VPLS) networks, instead of dedicated links, to provide connectivity between branch offices and data centers. Such networks typically provide a *virtual circuit or tunnel* whereby traffic from each branch office is shaped in the access network, most likely by means of a token-bucket shaper. In fact, such traffic is often hierarchically aggregated in backbone links also by means of a token-bucket mechanism. Moreover, the advent of virtualized infrastructures requires the application of different policies among tenants which share a common physical network architecture. Remarkably, traffic shaping algorithms have been explored as an alternative for such resource sharing, with the consequent introduction of token-buckets in these emergent deployments [1].

In this context, the verification of Service Level Agreements (SLA) is an issue of paramount importance for both operators and clients. Hence, and given their influence on the services' performance, the parameters of shaping mechanisms in operational networks must be correctly measured. For example, larger Peak Information Rates (PIR) and Maximum Burst Sizes (MBS) allow users to transfer larger traffic bursts at higher speeds, which greatly influences the perceived QoS in video broadcasting [2]. Furthermore,

there are several QoS classes offered in the network (gold, silver, real-time, etc.) each one with its own shaper parameters. Hence, the estimation of the shaper parameters is relevant in order to choose which service to transport over which QoS class. To complicate matters, the lack of standardized methodologies to measure network performance produces a dispersion on the achieved results depending on the specific configuration of the tests —e.g., transport protocol, packet size, amount of transferred data, among others. Actually, recent works such as [3] have addressed the problems related to TCP-based measurements, which in fact is broadly relied by end users to test the performance of their connections.

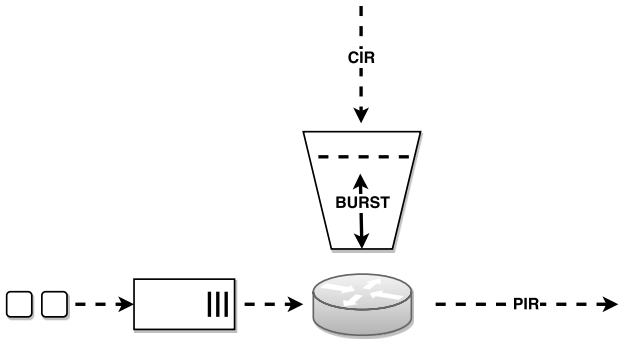
Those facts expose the broad adoption of token-bucket in multi-hop environments, how their configuration exerts a direct effect in the QoS for a diversity of end-users, and the necessity of well-defined methodologies to obtain accurate results. Paradoxically, the problem of detecting chained token-bucket-based shapers and estimating their parameters has not received attention from the research community —even in recent works related to this matter, only the narrow link is well-characterized [4].

To fill this gap, in this paper we present *TBCheck*, a novel algorithm to estimate the parameters of token-bucket-based shapers (namely, CIR, PIR and MBS). As a distinguishing feature, *TBCheck* is able to detect the effect of chained token-buckets in multi-hop environments. At the same time, it also surpasses other state-of-the-art tools by keeping a minimal intrusiveness on the network, which makes it suitable for measurements during both busy and off-peak hours. We present an extensive empirical performance evaluation that assesses the suitability of the methodology to characterize the behavior of a broad variety of scenarios.

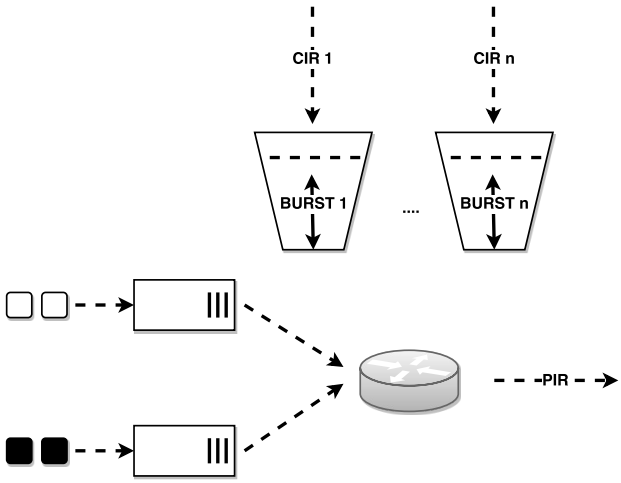
The rest of this paper is organized as follows: Section 2 presents the fundamentals of token-bucket algorithms and related measurement techniques, followed by the state of the art in Section 3. Then, we proceed with both single-hop and multi-hop token-bucket parameter estimation in Section 4. Next, Section 5 presents a performance evaluation of the proposed technique. Finally, Section 6 highlights the main conclusions of this work.

2. Problem description

Prior to the state of the art, we provide a brief description of the token-bucket algorithm, and the problem that motivates the development of *TBCheck*.



(a) Single queue



(b) Differentiated services

Figure 1: Token-bucket Algorithm Architecture: 1a depicts a single queue architecture, while 1b represents a multi-queue deployment, each queue with its own corresponding bucket.

Token-bucket is a network admission control algorithm which is aimed at limiting the services' (or users') transmission rates. Figure 1a shows the typical setup of a token-bucket-based shaper where packets are queued until enough tokens are available in the bucket. As it turns out, tokens are generated at a constant rate which is typically called Committed Information Rate (CIR). Each token represents a predefined amount of traffic and they are accumulated in a token-bucket with a given maximum capacity, which is typically expressed in bits. If enough tokens are available, the incoming packet is transmitted at the maximum rate (called PIR) that may be equal

to or lower than the link speed. The maximum amount of bits that can be transmitted at PIR (namely, the token-bucket capacity) is called MBS. Once a packet is transmitted, the corresponding tokens are removed from the bucket. If there are no available tokens, the packet is queued and the token-bucket transmission rate becomes limited by the CIR.

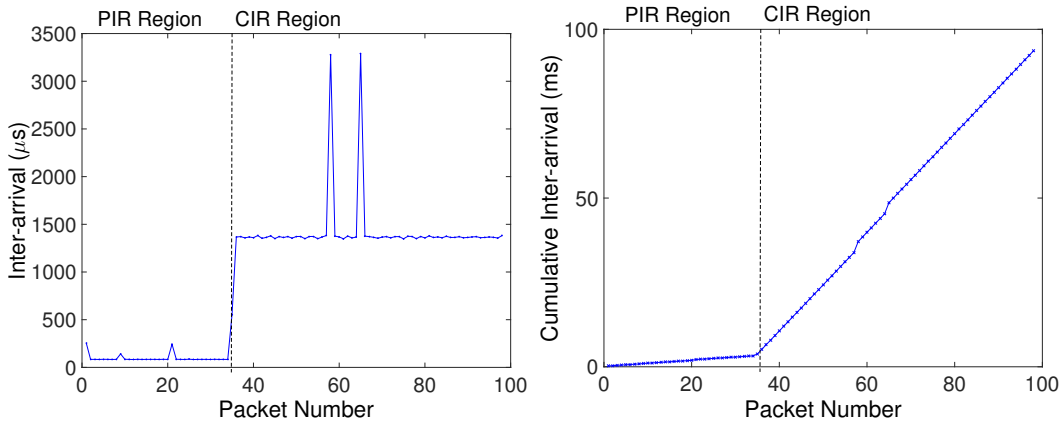
Additionally, token-bucket algorithm can be used to shape differentiated services according to diverse policies. For instance, a higher speed may be imposed for real time applications (teleconferencing, video on demand, etc) than for file transfers. To this end, several queues and token-buckets are incorporated for each specific service, as illustrated in Figure 1b.

To sum up, a token-bucket limits the rate to the token generation rate (CIR), unless the bucket has available tokens. In such a case, if the bucket is full, up to MBS bits will be transmitted at the maximum speed (PIR), thereby shortening the inter-arrival times of packets that find the bucket full upon arrival. To illustrate these issues, we have conducted a simple experiment where a token-bucket is deployed between two physical machines directly connected using a 100 Mbps Ethernet link. The rate limitation is performed by means of Linux `tc tbf` utility. To generate traffic, a packet-train is sent using Linux `pktgen`[5] kernel module and, on the receiver side, packets are captured using `tcpdump`. We have added background concurrent traffic to provide a more realistic scenario. Such background traffic has been generated using *hping3* [1] tool.

Figure 2a shows the inter-arrival times of a 100-packet long train (1 KB sized packets) originally sent back-to-back, with a token-bucket configuration of $CIR = 6$ Mbps, $PIR = 100$ Mbps , $MBS = 40$ KB and concurrent traffic generated at 5 Mbps rate. Two regions can be clearly observed: packets from 1 to 35 are sent at PIR while packets from 36 to 100 are sent at CIR. *TBCheck* exploits such behavior to estimate the PIR using the packets present in the first region and the CIR using the remaining packets. This is accomplished by detecting change-points in the slope of the accumulated inter-arrival times, as shown in Figure 2b.

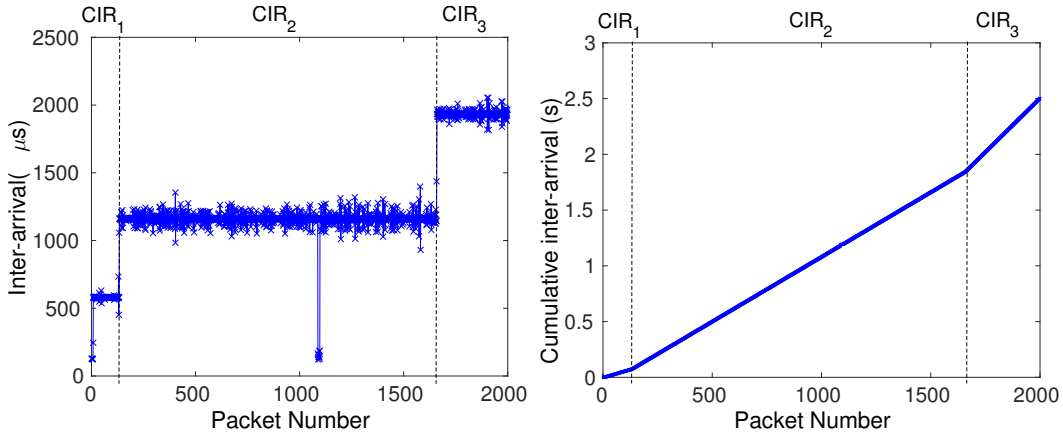
Following with the effect on packet inter-arrival times, we now address how chained token-buckets affect traffic that exceeds the shaping configuration. To this end, we have performed a simple experiment using the same setup previously commented with the difference that now we use three differ-

[1] `http://www.hping.org/hping3.html`



(a) Packet inter-arrival times. (b) Cumulative packet inter-arrival times.

Figure 2: Packet inter-arrival time distribution of a packet-train through token-bucket ($N = 100$ packets, $B = 1$ KB, $\widehat{r} = 38$, $CIR = 6$ Mbps, $PIR = 100$ Mbps and $MBS = 40$ KB) in the presence of cross-traffic (5 Mbps).



(a) Packet inter-arrival times. (b) Cumulative packet inter-arrival times.

Figure 3: Packet inter-arrival time distribution of a packet-train through multiple token-bucket —$N = 2000$ packets, $CIR_1 = 20$ Mbps, $CIR_2 = 10$ Mbps, $CIR_3 = 6$ Mbps, $MBS_1 = 10$ KB, $MBS_2 = 100$ KB, $MBS_3 = 1000$ KB and $PIR = 100$ Mbps.

ent physical machines each one with its own `tc tbf` configuration. Figure 3a and Figure 3b show the inter-arrival time and the cumulative inter-arrival for a train of 2000 packets originally sent back-to-back in a multi-hop scenario where $CIR_1 = 20$ Mbps, $CIR_2 = 10$ Mbps and $CIR_3 = 6$ Mbps. Besides, MBS values are 10 KB, 100 KB and 1000 KB respectively. As it can be observed, three different change-points can be identified. The first one (close to the 10^{th} packet) represents the change-point from the packets transmitted at PIR (100 Mbps) to the packets transmitted at CIR_1 (20 Mbps). The second one (around the 150-th packet) represents the change point from the packets transmitted at CIR_1 (20 Mbps) to the packets transmitted at CIR_2 (10 Mbps). Finally, the last change-point (close to the 1600^{th} packet) represents the change of the packets transmitted at CIR_2 (10 Mbps) to the packets transmitted at CIR_3 (6 Mbps).

This behavior, which exhibits slope changes in the accumulated inter-arrival times series, is exploited in *TBCheck* to detect several token-bucket shapers deployed along a multi-hop path. Consequently, *TBCheck* is able to detect shapers that constrain the traffic —that is, only if they are an actual bottleneck for the incoming traffic. We provide further details about these aspects in Section 4.3.

3. Related work

3.1. Packet-pair and packet-train techniques

In this section we briefly review packet-pair and packet-train techniques, since they constitute the basis for bandwidth estimation algorithms. Several tools such as *CapProbe*, *Iperf* (UDP) and also our *TBCheck* are based on the packet-pair or packet-train techniques. Packet-pair [6, 7] is a commonly used technique for estimating bottleneck link bandwidth. This method is based on the analysis of the inter-arrival time between two packets sent back-to-back. According to the bottleneck bandwidth and network conditions the two packets will be spaced with a given inter-arrival time. Then, the bottleneck bandwidth is estimated as the ratio between packet size and inter-arrival time. The main advantage of this method is that only a small number of packets is required, thus it presents minimal intrusion. However, this technique is very sensitive to cross traffic. Such interfering traffic influences the inter-arrival measured by the packet-pair methods in two different ways [8]:

- Expansion of inter-arrival time: when one or more interfering packets slip in between the two probe packets, the measured inter-arrival time increases and the estimated bandwidth decreases.

- Compression of inter-arrival time: when two probe packets with a particular inter-arrival time are queued together and interfering traffic fills the router queue, the two probe packets are dequeued as fast as the router link allows. Thus, the probe packets present a smaller inter-arrival time, which hides the bottleneck link effect on the bandwidth.

Obviously, estimating shaper parameters using a single pair is not feasible as, at least, two different rates have to be measured. In this line, packet trains have been proposed as an alternative to mitigate this limitation and the previously described negative effects. In this case, a packet-train is sent and the link bandwidth is estimated at the receiver using the minimum inter-arrival gap, the more packets the more chances for the inter-arrival times not to be affected by interfering traffic [8]. However, packet-train techniques have drawbacks in terms of intrusiveness and packet loss at high link utilization.

For token-bucket parameter estimation, packet-train techniques lack accuracy due to the two different shaping levels —CIR and PIR. If we choose the minimum inter-arrival between two packets of a train we are prone to measure the PIR but not the CIR. Conversely, if we try to remove the PIR influence, possibly by using the average packet inter-arrival times then we get closer to the CIR estimation, which is possibly biased because several packets in the train may be transmitted at the PIR instead of at the CIR. Let us consider a link with PIR and CIR equal to 100 Mbps and 6 Mbps respectively. Then, the inter-arrival times between typical Ethernet MTU-sized packets at the CIR are approximately equal to 2 ms whereas the PIR inter-arrival times between the same packets are approximately equal to 121 μs, namely a very significant difference. Furthermore, to complicate matters, we note that the value of MBS/PIR, i. e. the maximum transmission time at the PIR, typically ranges from 1 ms to 2000 ms. In this range, the packet-train receiver is usually affected by Interrupt Coalescence [9] which further pollutes the inter-arrival times estimation. More specifically, the network interface card issues an interrupt per group of packets, which have the same timestamp. Then, the packet inter-arrival time within the burst is negligible. Clearly, such packets cannot be used for PIR nor CIR estimation whatsoever.

3.2. Related measurement tools

Several studies have focused either on the impact of token-bucket parameters over different types of traffic [2, 10] or on the efficient implementation of token-bucket-based shaping schemes [11, 12, 13]. Additionally, authors in [14] analyzed the impact of shaping the traffic in terms of delay and peak rates using different flavors of token-bucket-based algorithms. As the authors point out, *ad hoc* shaping policies are usually implemented by ISPs and the application of multiple shapers may produce large performance losses for network flows. Therefore, measuring the token-bucket parameters is very relevant, because of their impact on the QoS.

However, measuring the parameters of shapers on real networks has not received much attention by the research community. Actually, the most recent techniques in the literature, such as *Iperf*[2], *Speedtest* [3] and *CapProbe*[4], deal with bandwidth estimation and not token-bucket characterization.

As for token-bucket specific measurement techniques, authors in [15] present an analytical method to estimate the Committed Information Rate (CIR) and MBS of a token-bucket. The main drawback is that the proposed technique requires prior knowledge of the input traffic (in terms of the related stochastic model that describes the traffic). This is not feasible in a real scenario because the traffic distribution is unknown and it is very dependent on the users and network. Furthermore, the traffic mix in the current Internet is very diverse and difficult to model stochastically. In this light, in [16], authors conducted a performance analysis of a token-bucket shaper for MPEG4 video and a real audio signal. Both approaches lack generality as they consider very specific conditions, which are known beforehand.

Reference [17] presents *BonaFide*, a tool for detecting shaping in mobile environments. Such tool is able to generate traffic from six different profiles in order to detect whether a shaper is present or not. However, the tool is not designed to characterize the shaping parameters and it is only suitable for mobile environments. Similarly, in [18], the authors present a tool to detect traffic shaping differentiation based on a set of protocols. Such approach only detects if a shaper is present or not and lacks generality as it only considers a given group of protocols.

[2] `http://sourceforge.net/projects/iperf/`

[3] `http://www.speedtest.net`

[4] `http://www.cs.ucla.edu/NRL/CapProbe/`

On the other hand, *ShaperProbe* [19] is a tool which aims to detect whether operators perform traffic shaping on residential access links or not. A synthetic traffic stream is sent to the link and changes in the received rate are analyzed to detect the different token-bucket states (high -PIR- or low -CIR-). Despite the good results achieved by the tool, we note that a huge amount of traffic (close to 480 MB) must be generated to estimate the token-bucket parameters. As it turns out, MPLS networks have statistical multiplexing gain and their performance depends on the overall network utilization [20]. Therefore, the measurement technique should not overload the network during peak hours, in order not to degrade the service.

4. Estimation algorithm

Once we have stressed the problems which motivate the development of new methods for token-bucket parameters estimation, we first describe our algorithm for single-hop scenarios and, then, we proceed with the multi-hop counterpart.

4.1. Algorithm description

To start with, we note that in absence of cross traffic and by design of the token-bucket algorithm, if a train of N constant-sized packets is sent back-to-back through a token-bucket shaped link, the first $\widehat{r}$ packets are forwarded at the PIR as enough tokens are available. After that, the remaining packets $(N - \widehat{r})$ are forwarded at the CIR, as this is the generation rate for new tokens and when the bucket is empty packets may not be sent. Based on this observation, we distinguish two cases, namely $\widehat{r} < N$ and $\widehat{r} = N$. Basically, the estimation of the CIR and PIR is based on the change point detection $(\widehat{r})$ between packets at PIR and CIR —if $\widehat{r} < N$ — and, then, on the measurement of the bandwidth values before and after such change-point.

The change-point $(\widehat{r})$ detection method is presented in Algorithm 1. It estimates the change-point $\widehat{r}$ by comparing the slope of the cumulative inter-arrival times at sample i with the slope of the cumulative inter-arrival times calculated in limited data windows. If a significant difference between such two slopes is observed, a rate shift has occurred due to the modification of inter-arrival times. Figure 4 shows the comparison process between the windowed and the cumulative slope values for a window value of 3. Each time a packet arrives, we add its inter-arrival to the cumulative inter-arrival packet list and we calculate the slope for all i packets. Similarly for the

Algorithm 1 Change-point detection

Input: k, w, inter-arrivals
total_samples = *length*(inter-arrivals)
i=w
window_slope=0
cumulative_slope=0
while (cumulative_slope $\leq$ window_slope $\times$ k) AND (i $<$ total_samples) **do**
  Calculate slope of cumulative inter-arrivals samples in current window (window_slope)
  Calculate slope of all cumulative inter-arrivals samples (cumulative_slope)
  i=i+1
end while
if i+w== total_samples **then**
  {No change is detected}
  return w+1
else
  {Change detected}
  return i+w
end if

windowed value we calculate the slope but only taking into account the last three packets. As new packets arrive, the window slides.

In the algorithm the w (window size) parameter represents the number of samples used to calculate the windowed inter-arrival time curve slope, the higher the value the more accurate the slope estimation. As it turns out, larger window size values produce deviations of the change-point estimated value, $\widehat{r}$, from the real change-point value. To detect the difference between slopes in the change-point a threshold must be used. In this light, the parameter k represents the hysteresis factor. Such factor is used to determine whether there is a significant difference between the slope of the cumulative inter-arrival time curve and the windowed counterpart. As the difference between the CIR and PIR value decreases, the hysteresis factor must be reduced in order to detect the change-point correctly. Based on our initial experimental analysis, the hysteresis values (k) for $PIR = 1000$ Mbps and CIR ranging between 6 and 800 Mbps must be located between 3 and 1.1 —in the following sections, we formally describe a methodology to refine this coarse bounding. Finally, we use the cumulative inter-arrival time slope and estimate the change in the slope using the least square error method, given

by the general expression in Equation 1 —y_i are the cumulative inter-arrival times and x_i the packet numbers.

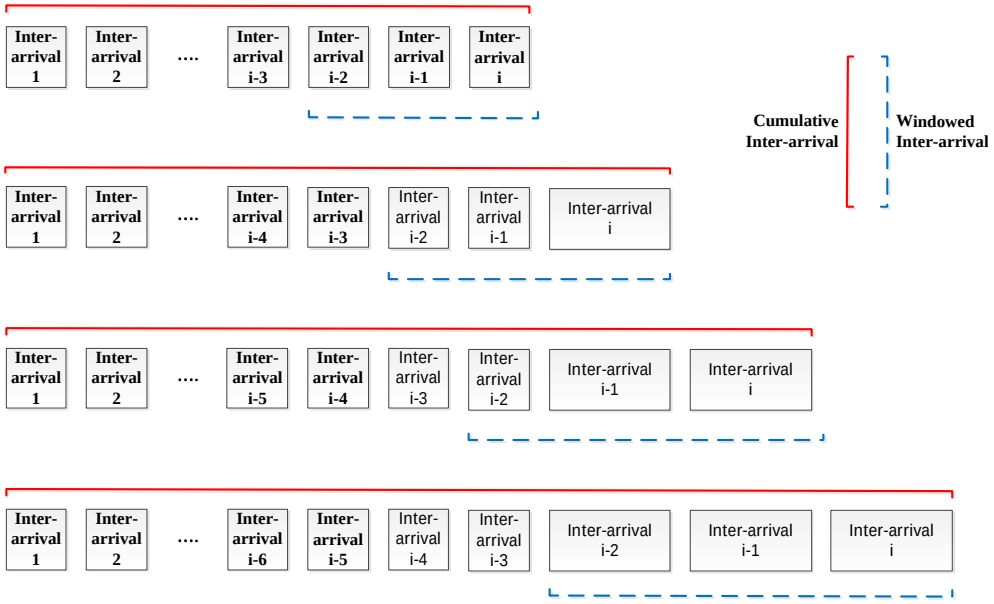


Figure 4: Windowed and cumulative slope comparison process

$$\begin{cases} y_i = f(x_i) = slope \cdot x_i + \beta_0, \text{where } slope,\ \beta_0 \in \mathbb{R} \\ \widehat{slope} = \dfrac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \end{cases} \tag{1}$$

Furthermore, if $\widehat{r} < N$, we can estimate the CIR and PIR at the receiver as follows:

$$\widehat{CIR} = \frac{B}{\underset{i=\hat{r},\ldots,N-1}{\text{median}}\ (t_{i+1} - t_i)} \tag{2}$$

$$\widehat{PIR} = \frac{B}{\underset{i=1,\ldots,\hat{r}-1}{\text{median}}\ (t_{i+1} - t_i)} \tag{3}$$

with t_i denoting the arrival time of the $i-th$ packet, $i = 1 \ldots N$.

Then, the MBS value can be estimated as follows:

$$\widehat{MBS} \;=\; \widehat{r} \times B \left(1 - \frac{\widehat{CIR}}{\widehat{PIR}}\right) \tag{4}$$

where B is the packet length expressed in bits.

On the other hand, if $\widehat{r} = N$, the link is not filled by the train (i.e. $MBS > N \cdot B$), hence every packet is sent at PIR. In this case, the packet-train length must be increased in order to detect the two working regions. Importantly, if no change point is detected after the increment, then we can assume that there is no shaping.

Once the change-point is detected, the CIR and PIR values must be calculated as shown in Equations 2 and 3. Packets usually present some dispersion because real shaper implementations are not totally deterministic. To circumvent this issue the median value is calculated over the inter-arrival times of both PIR and CIR regions as representative elements of each region. By doing so, the effect of outliers is minimized and consequently the bandwidth calculation is robust.

Note that sampled inter-arrival times may oscillate around the real value due to the effect of cross traffic and interrupt coalescence [9] in the receiver side. In order to avoid the influence of such effects, the inter-arrival times used for CIR and PIR calculation are pre-processed using a smoothing algorithm (similar as the one in [21]) based on exponential weights. Even though this technique minimizes the effects of the outliers, the measurement may still be affected by values that significantly diverge from the average value. Consequently, extreme outliers must be removed before the smoothing is applied, using the difference between an inter-arrival time sample and the previous one as a filtering parameter. To filter out extreme outliers, the typical threshold of three times the inter-quantile range is adopted.

Now, let us focus on multi-hop scenarios where two or more token-bucket-based shapers are present along an end-to-end path. In this case, depending on the configuration parameters and order of application of the shapers, several measurement scenarios are possible.

To detect and estimate the parameters of each token-bucket along an end-to-end path, a similar approach to the one presented in Section 4.1 may be used. In this case, Algorithm 1 is slightly modified to detect all change-points (not only the first one) in a packet sequence. Once all the change-points are detected, the CIR and MBS values are calculated over each packet group

delimited by the intervals defined by a pair of consecutive change-points ($\widehat{r_n}$ and $\widehat{r_{n-1}}$). The CIR, PIR and MBS values associated to the the n^{th} change point are now calculated following Equations 5 to 7.

$$\widehat{CIR_n} = \frac{B}{\underset{i=\hat{r_n},\ldots,\hat{r_{n+1}}}{\text{median}} (t_{i+1} - t_i)} \quad (5)$$

$$\widehat{PIR_n} = \frac{B}{\underset{i=\hat{r_{n-1}},\ldots,\hat{r_n}}{\text{median}} (t_{i+1} - t_i)} \quad (6)$$

$$\widehat{MBS_n} = \widehat{r_n} \times B \left(1 - \frac{\widehat{CIR_n}}{\widehat{CIR_{n-1}}}\right) \quad (7)$$

Specifically, in one-hop scenarios only two rates appear (namely, $\widehat{PIR}$ and $\widehat{CIR}$), which define two differentiated behavioral regions separated by a unique change point $\widehat{r}$. However, multi-hop scenarios exhibit more than one change points ($\widehat{r_n}$) (i.e., more than two regions), which requires adapting the definitions of $\widehat{CIR}$ and $\widehat{PIR}$ as presented in Equations 5 and 6. Furthermore, we need the value of $\widehat{CIR_{n-1}}$ (as this rate takes the place of the $\widehat{PIR}$ in the previous scenario) and $\widehat{CIR_n}$ to adapt the definition of $\widehat{MBS_n}$, as presented in Equation 7.

4.2. Selection of the algorithm parameters

As stated above, *TBCheck* depends on the values of a set of parameters (namely, the hysteresis factor k and window size w) to properly detect shaper behavior. To select the optimal values of k and w we performed a parameter optimization following a grid search strategy. Equation 8 presents the definition of the aggregated error metric to guide this grid search in terms of the disparity among actual and estimated values for the 3-tuple (CIR, PIR, MBS):

$$\epsilon(k,w) = \beta_1 \delta_{\widehat{CIR}(k,w)} + \beta_2 \delta_{\widehat{PIR}(k,w)} + \beta_3 \delta_{\widehat{MBS}(k,w)},\ \beta_1 + \beta_2 + \beta_3 = 1 \quad (8)$$

where $\delta_{\widehat{CIR}(k,w)}$, $\delta_{\widehat{PIR}(k,w)}$ and $\delta_{\widehat{MBS}(k,w)}$ represent the expected relative error of each parameter for each (k, w), and β_1, β_2, and β_3 are the weight that adjust the importance of each parameter.

First of all, note that we can link each (k, w) to the expected value in the space $(\widehat{CIR}, \widehat{PIR}, \widehat{MBS})$. Therefore, the optimization formulation aims to find the nearest (k, w) to the one that corresponds to the actual values of (CIR, PIR, MBS) —i.e., minimizing a distance. Additionally, we include the following restrictions:

1. **No dependency with respect to the units of the parameters:** required given the usual disparity among the three parameters, and to prevent aberrations as a result of rescaling.
2. **Invariant for translations:** that is, it does not depend on the actual values of the shapers' parameters, but only on the unsigned differences between them and the estimations.
3. Given that several values of (k, w) may lead to equivalently acceptable results, we impose that two elements of the parametric space, (k_1, w_1) and (k_2, w_2) produce equivalent results if the improvement in any of the dimensions (CIR, PIR or MBS) equals the loss of accuracy in the rest.
4. **Flexible adaptation of the relevance of individual error metrics:** depending on the use case, not all the dimensions may have the same importance.

The first property can be obtained if estimated values are divided by the actual values. Second and third properties link the error metric to the L^1 distance in $\mathbb{R}^3$. Finally, the last property requires the introduction of a rescaling value for each component —which can be constrained to the convex hull of the components without loss of generality. Therefore, putting all together we get the derivation for $\epsilon(k, w)$ presented in Equation 9.

$$
\begin{aligned}
&||(\tfrac{\beta_1}{CIR}CIR, \tfrac{\beta_2}{PIR}PIR, \tfrac{\beta_3}{MBS}MBS)- \\
&(\tfrac{\beta_1}{CIR}\widehat{CIR}(k,w), \tfrac{\beta_2}{PIR}\widehat{PIR}(k,w), \tfrac{\beta_3}{MBS}\widehat{MBS}(k,w))||_1 = \\
&\\
&\tfrac{\beta_1}{CIR}|CIR - \widehat{CIR}(k,w)| + \tfrac{\beta_2}{PIR}|PIR - \widehat{PIR}(k,w)| \\
&+\tfrac{\beta_3}{MBS}|MBS - \widehat{MBS}(k,w)| = \\
&\\
&\beta_1\delta_{\widehat{CIR}(k,w)} + \beta_2\delta_{\widehat{PIR}(k,w)} + \beta_3\delta_{\widehat{MBS}(k,w)} = \epsilon(k,w)
\end{aligned}
\tag{9}
$$

We remark that in this case the grid search explores the parameter space $k \times w$. In practice, such space can be bounded by the product $[1.1, \ldots, 3] \times$

$[2, \ldots, (N-1)]$, where N is the train length. Specifically, the values of w cannot be out of the compact $[2, \ldots, (N-1)]$, and the bounds for k suffice to detect all the measurable changes in the slopes. Consequently, the accuracy can be explicitly evaluated for all the points in the parameter space, selecting the optimal combination for each scenario.

As a final consideration, we note that the adjustment of the optimal sensitivity for the change-point detection, i.e., which threshold is considered to indicate a change in the slope (k) and how many samples are considered to compute the slope (w), is harmed if cross-traffic is introduced during the optimization stage. Specifically, the adjustment of these parameters can be distorted as a result of the binary phenomenon of estimation / no estimation of MBS, without improving the overall algorithm accuracy —we comprehensively analyze this matter in the following sections. Therefore, the optimization of the parameters k and w to minimize $\epsilon(k, w)$ should be accomplished in absence of cross-traffic.

4.3. Discussion

In the following, we discuss some aspects that must be considered during the application of *TBCheck* in practice. First, we formally define the conditions in which it provides a complete estimation of the parameters of token-bucket shapers along multi-hop paths. Second, we show how precision loss when timestamping packets can affect the achievable performance of the algorithm.

With this, we state key issues for the deployment of *TBCheck* implementations; and motivate the methodological principles that guide the evaluation and comparison with other state-of-the-art tools in the following section.

4.3.1. Conditions for the detection of token-buckets

As explained before, *TBCheck* estimates the parameters of token-bucket shapers by analyzing their effect in CBR traffic —specifically, by considering the behavior of cumulative packet inter-arrival times. This leads to the following conditions for the parameters that can be detected with our algorithm:

- C1. *TBCheck* can detect a token-bucket shaper $\iff$ it modifies the behavior of cumulative packet inter-arrival times.

- C2. Let $\mathcal{P} = \{PIR_i\}_{i=1,\ldots,N}$ be the space of PIR values corresponding to N chained token-bucket shapers. Then, *TBCheck* can estimate $PIR_i \iff PIR_i = \min(\mathcal{P})$.

- C3. Let $\mathcal{C} = \{CIR_i\}_{i=1,\ldots,N}$ be the space of CIR values corresponding to N chained token-bucket shapers. Then, *TBCheck* can estimate $CIR_i \iff MBS_j > MBS_i \, \forall j$ such that $CIR_j < CIR_i$.

- C4. Let $\mathcal{M} = \{MBS_i\}_{i=1,\ldots,N}$ be the space of MBS values corresponding to N chained token-bucket shapers ordered by their corresponding change-point. Then, *TBCheck* can estimate $MBS_i \iff \{CIR_i, CIR_{i-1}\}$ can be estimated.

C1 is a consequence of the *TBCheck* operation. The detection of a token-bucket shaper depends on the alterations of inter-arrival times corresponding to CBR traffic. This causes that every token-bucket with actual shaping effects is detectable with our algorithm.

C2 follows from the maximum rate of traffic traversing the multi-hop scenario. If there exists a limiting PIR ($\min(\mathcal{P})$), it is necessarily an upper-bound for the maximum measurable rate for network traffic traversing that path. This upper-bound implies that other $PIRs$ cannot be estimated (i.e., higher traffic rates cannot be reached) if they are not equal to such limiting PIR.

C3 and C4 adapt C1 to the detection of each configured CIR and MBS. Specifically, CIR_i can be detected if and only if it generates a change-point in the cumulative inter-arrival times slope —i.e.,there are no more restrictive token-bucket shapers limiting the rate of the traffic. That is equivalent to the specified joint relation in C3. A similar reasoning with the definition of the estimations that *TBCheck* provides leads to C4.

4.3.2. Implementation constraints

Once the algorithm detection conditions have been explained, let us now focus on the inherent specific problems of the implementation of the method. As our method relies on packet inter-arrival times, we need to obtain accurate measurements of such a magnitude. Obtaining accurate timestamps for incoming packets is a challenging task especially when capturing packets in high speed scenarios. Typically, network measurement tools are implemented in software and are executed in general purpose operating systems such as

GNU/Linux. Performing the packet timestamping in software is not a deterministic task and variance may exist[22]. Such variance may affect negatively to the estimation algorithm. To illustrate such issue, we have transmitted back-to-back a packet train of 10,000 UDP packets of size 1450 Bytes between two Linux hosts directly connected with a 1 Gbps Ethernet link. The train has been sent using Linux `pktgen`[5] module and has been received using `tcpdump`. Figure 5b inter-arrival times of the packet train while Figure 5a shows the Empirical Cumulative Distribution Function (ECDF) for the inter-arrival of the 10,000 packets. The theoretical inter-arrival value for this scenario is 11.6 μs (1450 Bytes *8/1000 Mbps). Analyzing the inter-arrival times distribution we can observe that approximately 45% of the samples are located near the theoretical value while the rest are deviated. Such a deviation ranges between 1 and 25 μs. Note that even in the best-case deviation (1 μs) the capacity estimation is 1094 Mbps (1450 Bytes *8/ 10.6 μs) which represents a relative error of 9.4%. Similarly, when the inter-arrival deviation is 25 μs the capacity estimation is 436 Mbps (1450 Bytes * 8 / 26.6 μs) which represents a 56% of relative error. To reduce the estimation error, the timestamping can be performed by means of deterministic approaches such as Field Programmable Gate Array (FPGA) devices.

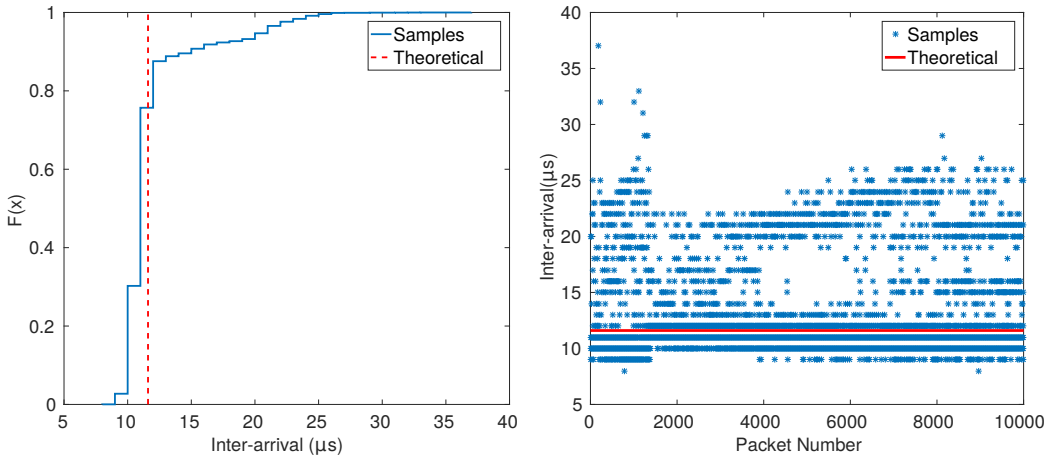
Moreover, when dealing with software token-bucket implementations such as the one present in `tc` tool, the shaping rate is not perfect (especially at high speeds)[23] and variance may exist due to the scheduling mechanisms of the operating systems and the time quanta used for such scheduling. Such behavior is also commented in the man page of the `tc tbf` command.[5]

5. Performance evaluation

5.1. Methodology and experimental conditions

In this section, we describe the experimental methodology that we follow to assess the performance of *TBCheck*. Our experimental design aims at a general evaluation of our method *(i)* in a variety of scenarios in terms of the situations described in Section 4.3, and *(ii)* with a minimal effect of the specific implementation of the algorithm. To this end, we evaluate the performance of the proposed solution in a testbed created using the *Mininet* [24] tool. *Mininet* allows the creation of custom SDN-enabled virtual networks

[5]https://linux.die.net/man/8/tc-tbf



(a) ECDF of inter-arrival times.

(b) Packet inter-arrival times.

Figure 5: Packet inter-arrival time distribution of a packet-train sent back-to-back trough a 1 Gbps link ($N = 10,000$ packets, $B = 1450Bytes$).

within a physical host. In our case, two different test scenarios have been created. The first, in what follows single-hop scenario, consists of two hosts (sender and receiver) connected through a switch that performs shaping. The second scenario, in what follows multi-hop scenario, also consists of two hosts, sender and receiver, connected through four switches in a linear topology as depicted in Figure 6.

To apply traffic shaping, the Linux shaper *tc* [6] with *tbf* (Token-bucket Filter) option has been used, which allows imposing different CIR, PIR and MBS values to the traffic. It is worth remarking that *tc* and similar tools are being used nowadays to perform traffic-shaping at the edges making use of software approaches[23].Furthermore, to generate cross-traffic, additional hosts have been attached to each switch of the topology that send packets by means of the *hping3* [7] tool.

We have implemented *TBCheck* in C language to test its performance. Our software codes the algorithms described in Section 4 to estimate the shaper parameters. Our implementation follows a client/server architecture,

[6] `http://linux.die.net/man/8/tc`

[7] `http://www.hping.org/hping3.html`

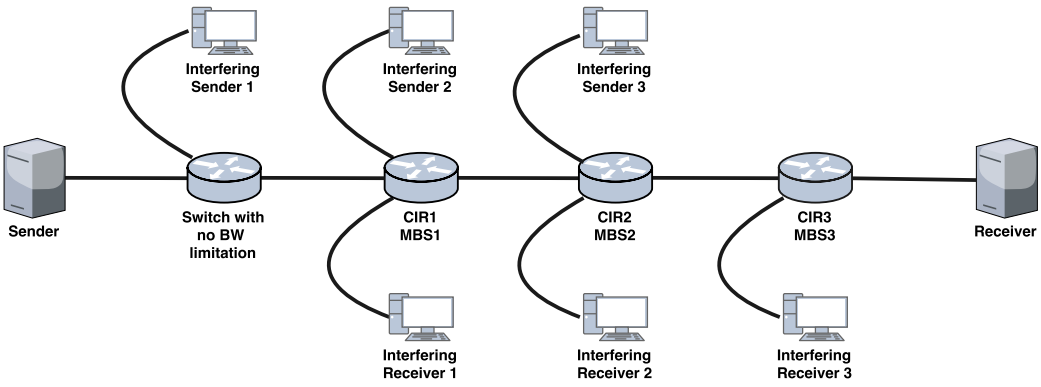


Figure 6: Topology used for the multi-hop tests.

where the server must be placed after the shaping systems whereas the client must be placed before the shaping systems.

5.2. Selection and definition of testing scenarios

With the two guiding principles stated above, we are going to analyze three-hop scenarios, as this is the minimal number of hops to represent all the combinations that may affect the accuracy of *TBCheck* —that is, situations in which all, some or only the most limiting token-bucket element can be detected. Besides, this deployment is suitable for the validation of the expected behavior of *TBCheck* in common operational scenarios, as previous work [25] suggests that the average number of hops in a MPLS tunnel ranges between two and five.

Table 1 summarizes the considered measurement scenarios depending on the parameters of the applied shapers along a three hop end-to-end path. Note that CIR_i and MBS_i represent the CIR and MBS at hop i. Importantly, we will assume that the PIR remains constant as it is a consequence of the conditions stated in the previous section. It is worth noting that in the typical multi-hop shaping scenario, the conformed speeds grow larger in the path from source to destination as ISPs usually aggregate traffic from different customers incrementally from their accesses to their core networks. Such scenarios correspond to classes 5 to 8 in Table 1.

First, we note that as the CIR imposed by the shaper decreases along the path from source to destination, the detection of the three shapers is only possible if the corresponding $MBSs$ are incremental. If MBS are not

Table 1: Measurement scenario characteristics in the presence of shapers and multiple hops—3 hops.

Class	CIR	**MBS**	**Measurable parameters**	
1	$CIR_1 > CIR_2 > CIR_3$	$MBS_1 < MBS_2 < MBS_3$	CIR_1	✓
			CIR_2	✓
			CIR_3	✓
2	$CIR_1 > CIR_2 > CIR_3$	$MBS_1 > MBS_2 > MBS_3$	CIR_1	×
			CIR_2	×
			CIR_3	✓
3	$CIR_1 > CIR_2 > CIR_3$	$MBS_1 < MBS_2 > MBS_3$ $MBS_1 > MBS_3$	CIR_1	×
			CIR_2	×
			CIR_3	✓
4	$CIR_1 > CIR_2 > CIR_3$	$MBS_1 > MBS_2 < MBS_3$	CIR_1	×
			CIR_2	✓
			CIR_3	✓
5	$CIR_1 < CIR_2 < CIR_3$	$MBS_1 < MBS_2 < MBS_3$	CIR_1	✓
			CIR_2	×
			CIR_3	×
6	$CIR_1 < CIR_2 < CIR_3$	$MBS_1 > MBS_2 > MBS_3$	CIR_1	✓
			CIR_2	✓
			CIR_3	✓
7	$CIR_1 < CIR_2 < CIR_3$	$MBS_1 < MBS_2 > MBS_3$ $MBS_1 > MBS_3$	CIR_1	✓
			CIR_2	×
			CIR_3	×
8	$CIR_1 < CIR_2 < CIR_3$	$MBS_1 > MBS_2 < MBS_3$	CIR_1	✓
			CIR_2	✓
			CIR_3	×

incremental then traffic is forwarded along hops without being conformed at each hop (as there are enough tokens) and only the most restrictive one shapes the traffic. On the other hand, when the CIR increases along the path the shapers can be detected if and only if the $MBSs$ are decreasing. In this case, if the MBS are not incremental the traffic stream rate will be limited to the CIR of an intermediate hop avoiding the exhaustion of tokens in the next shaper. For example if $CIR_1 = 6$ Mbps, $CIR_2 = 10$ Mbps and $CIR_3 = 20$ Mbps and $MBS_1 = 100$ KB, $MBS_2 = 10$ KB and $MBS_3 = 1000$ KB, traffic will traverse hop 1 at maximum rate and after exhausting the tokens (10 KB) at hop 2, the output traffic will be conformed at 10 Mbps speed. Thus, the input traffic at hop 3 will be lower than the CIR at hop 3 (20 Mbps) and will not be able to exhaust the tokens at hop 3 making impossible the estimation of both CIR and MBS at hop 3.

5.3. Result in single-hop scenarios

As a first step, the method has been validated estimating the parameters of the shaper on the single-hop scenario with several combinations of CIR and MBS. Table 2 shows the relative error in the estimations of the three parameters (CIR, PIR and MBS), with a CIR ranging from 6 Mbps to 80 Mbps and a MBS ranging from 10 KB to 10000 KB. A PIR value of 100 Mbps was selected for all the experiments and the packet length (B) to 1400 bytes. Regarding the train length, for each test we have used the double of the MBS (in number of packets) as it suffices to deplete the bucket. That is, if the MBS is 100 KB, we have used 200 packets. For the case of CIR=80 Mbps we have used trains of length 50000 packets as more packets are needed to completely deplete the bucket. Each experiment was repeated 10 times and the mean and standard deviation of the relative error was calculated. As it can be observed, the achieved accuracy is remarkable with an average relative error lower than 1% for the CIR parameter estimation and lower than 5% for the MBS parameter estimation in most of the cases. Note that these experiments have been performed using the optimal values of k and w obtained from the space parameter analysis explained in Section 4.2. Similarly, Table 3 shows the relative error in the estimations of the three parameters (CIR, PIR and MBS) with theoretical CIR values up to 800 Mbps and PIR=1 Gbps. As in the low-speed scenario, the error in the estimation of the CIR and MBS are below 2% and 13% respectively for all cases.

To test the effect of cross-traffic on the estimation of the token-bucket parameters, we have generated UDP cross-traffic at different constant rates ranging from 5% to 50% of the CIR value following the previously described methodology. Table 4 shows the relative error in the estimations of the three parameters (CIR, PIR and MBS) with theoretical CIR values up to 800 Mbps, PIR=1 Gbps and cross-traffic. Each experiment was repeated 10 times, and the mean and standard deviation of the relative error was calculated. For the sake of brevity we only show the worst-case (larger value of the error metric ϵ, defined in Equation 8) for each combination of CIR and MBS.

As it can be observed, the CIR relative error is still below 2% for all cases while PIR relative error is below 30%. However, the MBS relative errors are greater than the ones observed in absence of cross-traffic. This behavior is not surprising, as cross-traffic consumes tokens when traversing the shaper. That is, when measurement traffic is injected, it is likely that the bucket is

Table 2: Estimated Relative Error [%] for single-hop token-bucket parameters using *TBCheck*: CIR, PIR and MBS (no cross-traffic, B = 1400 Bytes, PIR = 100 Mbps and several cases of CIR and MBS). Mean and standard deviation. * denotes no error nor variance in the results.

CIR [Mbps]	MBS [KB]	$\widehat{CIR}$ Relative Error $\%[\mu \pm \sigma]$	$\widehat{PIR}$ Relative Error $\%[\mu \pm \sigma]$	$\widehat{MBS}$ Relative Error $\%[\mu \pm \sigma]$	w	k
6	10	0.025 ± 0.009	2.008 ± 0.77	2.94 ± 0.04	3	1.4
	100	0.021 ± 0.001	0.189 ± 0.40	0.254 ± 0.02	3	1.4
	1000	0.0178 ± 0.001	*	0.11 ± 0.001	3	2.9
	10000	0.021 ± 0.001	*	0.011 ± 0.001	3	2.9
10	10	*	2.37 ± 0.51	10.02 ± 0.06	3	2
	100	*	0.18 ± 0.40	0.09 ± 0.04	4	1.4
	1000	*	*	0.03 ± 0.01	4	1.1
	10000	*	*	0.03 ± 0	9	1.1
20	10	*	2.39 ± 1.43	5.59 ± 4.98	3	1.1
	100	*	*	0.62 ± 0.01	5	1.1
	1000	0.035 ± 0.001	*	0.02 ± 0.02	5	2
	10000	*	*	0.01 ± 0	3	2.6
50	10	*	2.84 ± 0.903	3.68 ± 2.1	4	1.1
	100	*	1.33 ± 0.81	13.43 ± 5.42	13	1.4
	1000	*	*	6.79 ± 3.31	12	1.7
	10000	*	0 ± 0.1	0.01 ± 0.002	3	1.4
80	10	*	0.36 ± 0.14	0.15 ± 0.040	6	1.1
	100	*	*	0.57 ± 0.16	10	1.1
	1000	*	*	6.02 ± 1.52	14	1.1
	10000	*	*	19.03 ± 1.283	14	1.1

either depleted or with a number of tokens that do not correspond to the maximum bucket capacity. As a result, in presence of cross-traffic (especially with sustained rates), MBS may not be estimated correctly. Remarkably, this evaluation exposed that the measured parameters errors hardly depend on the amount of cross traffic, but just on its presence.

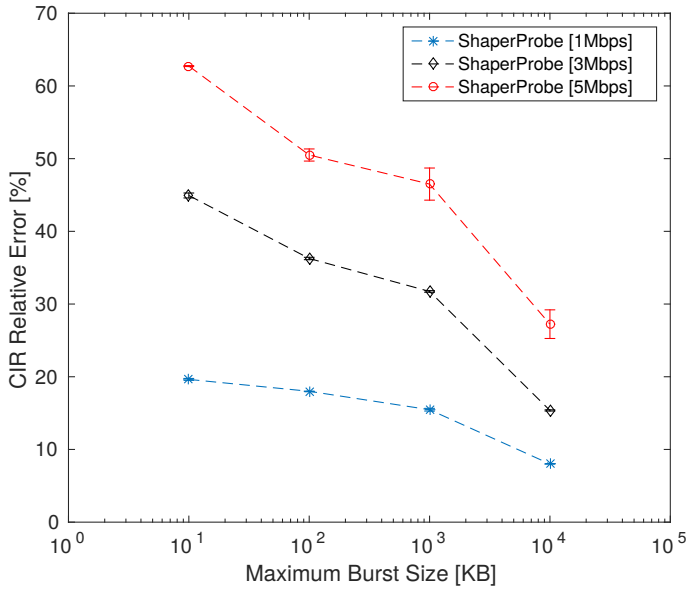
In the light of these results, the CIR and PIR values can be estimated during busy-hours of the network (as their estimations are not particularly affected by concurrent-traffic), while MBS should be estimated during off-peak hours in order to obtain accurate results. These restrictions allow us to define a shaping detection policy that must be applied when measuring token-bucket based systems.

Table 3: Estimated Relative Error [%] for single-hop token-bucket parameters using *TBCheck*: CIR, PIR and MBS (no cross-traffic, $B = 1400$ Bytes, $PIR = 1000$ Mbps and several cases of CIR and MBS). Mean and standard deviation. * denotes no error nor variance in the results.

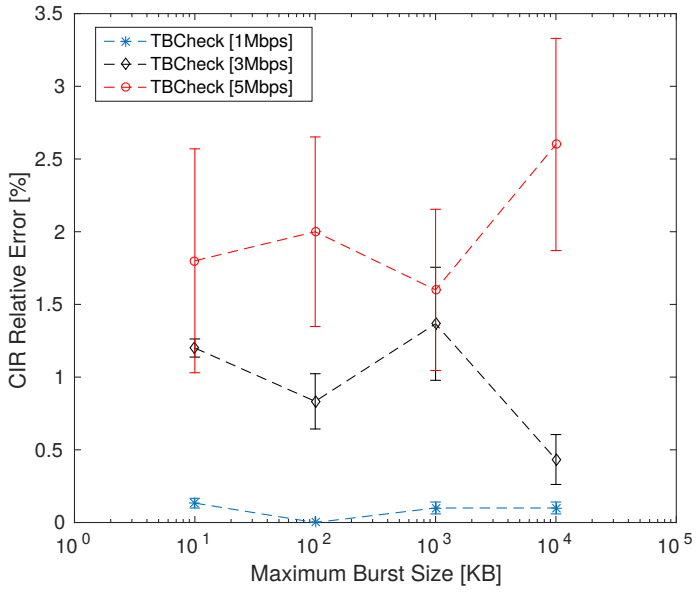
CIR [Mbps]	MBS [KB]	$\widehat{CIR}$ Relative Error $\%[\mu \pm \sigma]$	$\widehat{PIR}$ Relative Error $\%[\mu \pm \sigma]$	$\widehat{MBS}$ Relative Error $\%[\mu \pm \sigma]$	w	k
60	10	0.210 ± 0.072	9.059 ± 4.459	2.128 ± 0.393	3	1.1
	100	0.178 ± 0	12.205 ± 3.773	1.288 ± 0.541	12	1.1
	1000	0.177 ± 0	9.610 ± 3.861	2.856 ± 13.921	8	1.7
	10000	0.178 ± 0	1.970 ± 0.885	4.524 ± 18.371	5	2
100	10	0.034 ± 0.176	8.682 ± 4.489	9.605 ± 0.773	3	1.1
	100	*	12.453 ± 3.629	3.977 ± 0.491	14	1.5
	1000	*	8.581 ± 3.229	0.815 ± 0.337	14	2.3
	10000	*	1.818 ± 0	0.419 ± 0.004	14	2.3
200	10	*	11.592 ± 4.729	6.481 ± 3.094	3	1.1
	100	*	15.247 ± 3.192	3.454 ± 4.669	8	1.4
	1000	*	7.863 ± 2.721	1.058 ± 0.556	10	2.0
	10000	*	1.979 ± 0.885	4.372 ± 16.136	7	1.7
500	10	1.821 ± 0	9.982 ± 4.012	12.768 ± 5.258	6	1.1
	100	1.818 ± 0	15.692 ± 2.868	1.573 ± 2.692	10	1.2
	1000	1.817 ± 0	6.181 ± 1.479	1.786 ± 1.954	13	1.7
	10000	1.818 ± 0	1.818 ± 0	2.455 ± 8.132	6	1.4
800	10	*	2.813 ± 0.201	11.657 ± 3.417	6	1.2
	100	*	1.818 ± 0	10.302 ± 1.152	14	1.2
	1000	*	1.818 ± 0	10.273 ± 3.273	14	1.2
	10000	*	1.812 ± 0	9.463 ± 0.280	14	1.2

Once the accuracy of *TBCheck* has been assessed, we compare the results with the state-of-the-art bandwidth estimation tools. Table 5 shows the CIR estimation using common bandwidth-estimation tools such as: *Speedtest*, *Iperf* (UDP and TCP), *Capprobe*, a File-transfer tool based on the ETSI guide 202 057-4 [26] and a also a generic packet-train technique. We note that the generic packet-train technique estimates the capacity of an end-to-end path using the mean inter-arrival time and is affected by shapers as discussed in Section 4.1. Additionally, the shaper detection tool *ShaperProbe* has been added to the comparison.

As it can be observed in Table 5, as the burst length value increases, all tools tend to overestimate the CIR value with the exception of *TBCheck* and *ShaperProbe*. These results highlight the importance of using specifi-



(a) *ShaperProbe*.



(b) *TBCheck*.

Figure 7: CIR Relative Error ($PIR = 100$ Mbps, $CIR = 6$ Mbps, several cases of MBS and cross traffic [1, 3 and 5 Mbps]).Mean and standard error.

Table 4: Estimated Relative Error [%] for single-hop token-bucket parameters using *TBCheck*: CIR, PIR and MBS ($B = 1400$ Bytes, $PIR = 1000$ Mbps and several cases of CIR, MBS). Mean and standard deviation. * denotes no error nor variance in the results. Cross-traffic varies between 5% and 50% of CIR value for each case. Only worst results are shown.

CIR [Mbps]	MBS [KB]	Cross-traffic [Mb/s]	$\widehat{CIR}$ Relative Error $\%[\mu \pm \sigma]$	$\widehat{PIR}$ Relative Error $\%[\mu \pm \sigma]$	$\widehat{MBS}$ Relative Error $\%[\mu \pm \sigma]$	w	k
60	10	3	0.178 ± 0	28.099 ± 17.112	6.812 ± 4.560	3	1.1
	100	9	0.178 ± 0	25.764 ± 36.960	26.761 ± 41.454	12	1.1
	1000	18	0.178 ± 0	1.818 ± 0.000	68.534 ± 26.675	8	1.7
	10000	15	0.176 ± 0	2.780 ± 2.044	96.912 ± 0.905	5	2
100	10	5	*	27.696± 6.103	12.656 ± 15.376	3	1.1
	100	30	*	6.117 ± 7.706	8.629 ± 14.013	14	1.5
	1000	30	*	2.303 ± 1.533	40.635 ± 22.681	14	2.3
	10000	50	*	1.818 ± 1.148	4.654 ± 3.446	14	2.3
200	10	10	*	19.523 ± 14.653	12.377 ± 6.880	3	1.1
	100	40	*	3.854 ± 4.293	16.440 ± 27.803	8	1.4
	1000	90	*	1.81 ± 0.001	45.997 ± 24.787	10	2.0
	10000	70	*	2.303 ± 1.533	98.722 ± 0.345	7	1.7
500	10	225	1.818 ± 0	20.715 ± 10.983	31.498 ± 11.102	6	1.1
	100	25	1.818 ± 0	4.332 ± 7.542	28.009 ± 23.203	10	1.2
	1000	150	1.818 ± 0	2.303 ± 1.533	25.246 ± 23.601	13	1.7
	10000	150	1.818 ± 0	1.818 ± 0.000	62.588 ± 47.577	6	1.4
800	10	80	*	3.488 ± 3.572	35.959 ± 24.202	6	1.2
	100	80	*	2.303 ± 1.533	49.912 ± 17.388	14	1.2
	1000	280	*	2.303 ± 1.532	41.964 ± 42.413	14	1.2
	10000	400	*	2.303 ± 1.533	70.446 ± 46.594	14	1.2

cally tailored estimation methods when traffic shapers are present. Since only *TBCheck* and *ShaperProbe* perform accurate measurements for traffic shapers, we will only consider such tools in what follows.

Additionally, we now focus on the MBS estimation. We followed the same approach of obtaining ten different MBS estimates per test and calculating the standard deviation of the relative error. Table 6 provides the relative error in the MBS estimation for *TBCheck* and *ShaperProbe* tools in a single-hop scenario with $PIR = 100$ Mbps and $CIR = 6$ Mbps. As shown, *TBCheck* obtains accurate values. Using *ShaperProbe* similar estimation results are observed except for MBS values of 10 KB. In such a case, the tool does not provide the MBS estimation. A careful examination of its code revealed that *ShaperProbe* estimation method required more than 11 packets to estimate the MBS value. In the 10 KB case, the level shift between CIR and PIR takes place near the 9th packet and the tool starts estimating the MBS value *after* the change point.

Table 5: CIR Measurement (no cross-traffic, PIR = 100 Mbps, CIR = 6 Mbps and several cases of MBS).Mean and standard error.

	CIR [$\mu \pm SEM$]			
	MBS [KB]			
Tool	10	100	1000	10000
TBCheck	6.01 ± 0.00	5.99± 0.00	6.00± 0.00	6.01± 0.00
iperf(UDP)	94.84± 0.16	94.86± 0.12	95.00± 0.10	94.80± 0.06
iperf(TCP)	6.47± 0.00	6.55±0.00	7.23± 0.00	13.76± 0.06
CapProbe	91.84± 2.49	92.46± 1.45	91.38± 2.93	94.27± 1.83
SpeedTest	5.70± 0.00	5.70± 0.01	6.27± 0.52	18.64± 1.76
File-transfer	5.74± 0.00	5.81± 0.00	6.77± 0.00	94.20± 0.00
Packet-train	6.24± 0.00	18.55± 0.31	96.74± 0.18	96.80± 0.18
ShaperProbe	5.84± 0.03	5.84± 0.03	6.12± 0.41	5.78± 0.00

To compare the influence of cross-traffic on the shaping detection tools, we injected UDP cross-traffic at three different rates (1, 3 and 5 Mbps) concurrently with the measurement. The cross-traffic has been generated with *hping3* tool in combination with *tc netem* tool, in order to simulate random delays according to a Pareto-Normal distribution [27]. Such distribution is a weighted sum of a Normal distribution with weight 25% and a Pareto distribution with weight 75% and $\alpha = 3$ (finite variance). We note that this cross-traffic pattern is less demanding than the previous tests to assess the performance of our tool.

Figures 7a and 7b show the average CIR relative error estimation values for *TBCheck* and *ShaperProbe* tools for the three cross-traffic rates. As shown, *TBCheck* accurately estimates the theoretical CIR value (6 Mbps) even if the cross-traffic rate represents more than the 80% of the theoretical CIR. On the contrary, *ShaperProbe* tends to underestimate the CIR value even in low-speed scenarios due to the influence of cross-traffic mainly by two different reasons. First, *ShaperProbe* uses a predefined window size for CIR estimation. That is, instead of considering the expansion of inter-arrivals, its rate estimation is performed by averaging the number of received bytes along the time interval —which partly explains the underestimations, as the number of bytes received in a constant length time interval decreases. Second, *ShaperProbe* performs estimation of shaper parameters at user level and only

Table 6: Estimated Relative Error [%] for MBS using *TBCheck* and *ShaperProbe*: no cross-traffic, $PIR = 100$ Mbps and $CIR = 6$ Mbps. Mean and standard deviation.

Maximum Burst Size [KB]	Relative Error *TBCheck* (%)$[\mu \pm \sigma]$	Relative Error *ShaperProbe* (%)$[\mu \pm \sigma]$
10	2.94 ± 0.04	N/A
100	0.254 ± 0.02	1.3 ± 1.77
1000	0.11 ± 0.0001	2.46 ± 2.59
10000	0.001 ± 0.0001	3.58 ± 10.32

takes into account UDP payload size and not all underlying headers—UDP, IP and link layer.

Regarding intrusiveness, *TBCheck* generates packet trains of 50,000 MTU-sized UDP packets in the worst case scenario (MBS equal to 10,000 KB), namely 71 MB of traffic, in contrast with the 480 MB generated by *ShaperProbe*, nearly 7 times more. This is a very significant difference that surely affects the measured link available bandwidth.

5.4. Results in multi-hop scenarios

Up to this section, both *TBCheck* and *ShaperProbe* have been evaluated in a single-hop scenario even in the worst case when cross-traffic is present. Let us focus on the performance evaluation of the shaper parameter estimation in the multi-hop scenario whereby three shapers are applied sequentially. We note that *ShaperProbe* tool has been designed to detect the presence of a single shaper only, namely the most restrictive one. Thus, from now on this tool is not considered in the evaluation. Table 7 shows the different configurations and parameters used in the multi-hop performance evaluation testbed, grouped by classes presented in Table 1. The tested CIR values are 20 Mbps, 10 Mbps and 6 Mbps. PIR value reflects a typical SLA-based enterprise connection at 100 Mbps, and MBS values are 10 KB, 100 KB and 1000 KB.

For each class, ten experiments were performed and mean and standard deviation were calculated. Figures 8, 9 and 10 show the mean relative estimation error and standard deviation for CIR, MBS and PIR respectively

Table 7: Multi-hop testbed parameters grouped by class.

Class	CIR_1 **[Mbps]**	CIR_2 **[Mbps]**	CIR_3 **[Mbps]**	MBS_1 **[KB]**	MBS_2 **[KB]**	MBS_3 **[KB]**
1	20	10	6	10	100	1000
2	20	10	6	1000	100	10
3	20	10	6	100	1000	10
4	20	10	6	1000	10	100
5	6	10	20	10	100	1000
6	6	10	20	1000	100	10
7	6	10	20	100	1000	10
8	6	10	20	1000	10	100

in each of the 8 classes. Note that, in Figures 8 and 9, the estimation error in each hop, per class, is depicted. However, according to Table 1 there are several cases for which it is not possible to estimate either the CIR or the MBS or both. In such cases, no data is displayed.

The results show that multi-hop shaper detection and parameter estimation is achieved (whenever possible) with an estimation relative error lower than 1% for CIR parameter and lower than 15% for the most MBS parameter values. Regarding PIR estimation, the relative error achieved is lower than 8% in all cases. In the typical network scenario where $CIR_1 < CIR_2 < CIR_3$ and $MBS_1 > MBS_2 > MBS_3$, all the shapers are detected and the estimated relative error for the CIR parameter in all hops is close to 0%. For the MBS parameter, the relative error is lower than 14% in all three hops.

Furthermore, the previous multi-hop experiments are repeated in the presence of cross-traffic to simulate a real production network. More specifically, UDP cross-traffic with random delays according to Pareto-Normal distribution was generated at each hop using *hping3*, in order to obtain an average traffic load ranging from 10% to 60% at each hop. The 60% load upper limit is typically used to protect against traffic bursts as suggested in [28]. The experiments were performed using class 1 and 6 scenarios since they are the most typical in real-world scenarios.

Figure 11 and 12 shows the CIR relative estimation error for each class in the presence of concurrent traffic. As it can be observed, in class 1 scenario the estimation relative error is lower than 1% even when the link load is 60%. On the other hand, in the class 6 scenario the relative estimation error

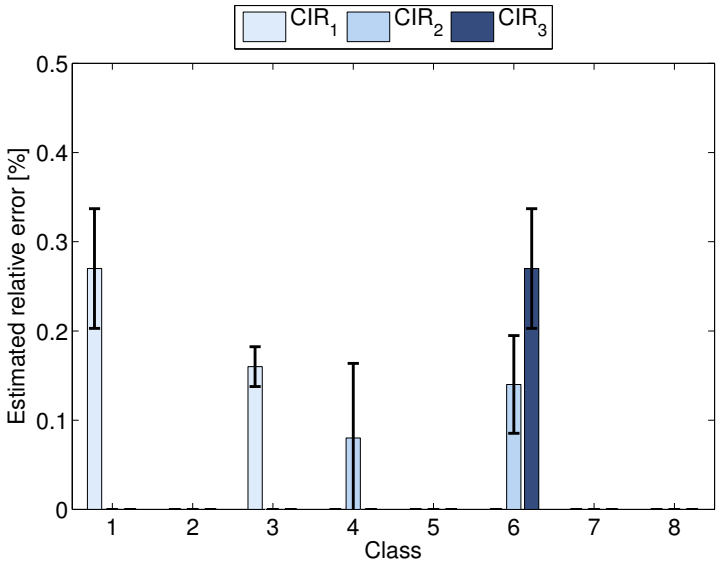


Figure 8: Estimated CIR relative error for multi-hop —no cross-traffic, 8 classes and 3 hops. Mean and standard deviation.

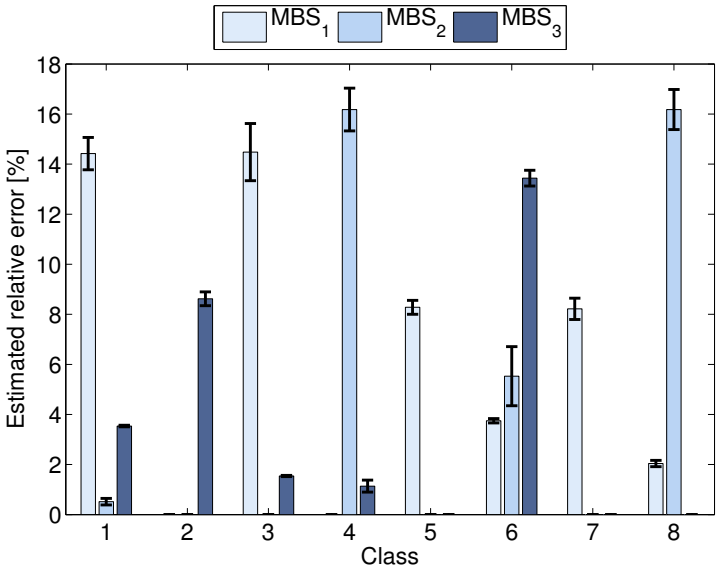


Figure 9: Estimated MBS relative error for multi-hop —no cross-traffic, 8 classes and 3 hops. Mean and standard deviation.

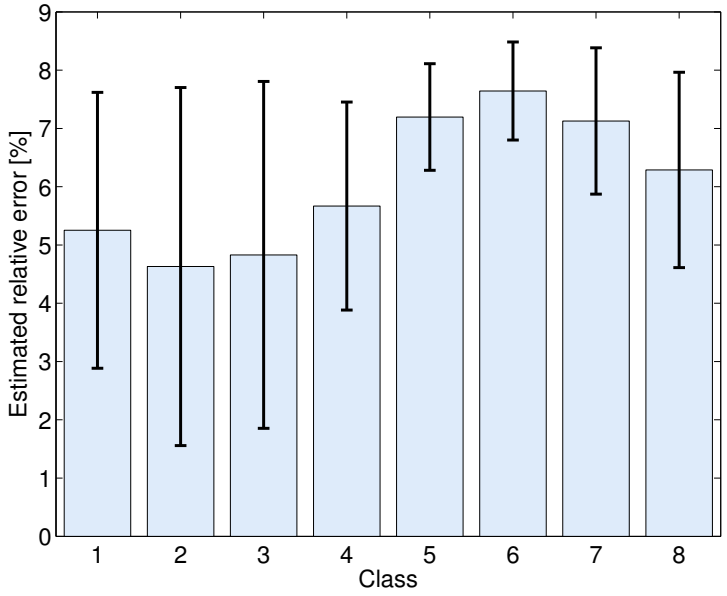


Figure 10: Estimated PIR relative error for multi-hop —no cross-traffic, 8 classes. Mean and standard deviation.

is lower than 5% in all cases with the exception of the link load of 60%. Note that a link occupied at 60% is considered as a highly-loaded link and, typically, operators and network managers keep the load below this value for stability reasons.

Figure 13 and 14 represent the MBS relative estimation error for each class in the presence of concurrent traffic. In this case, the estimation error is lower than 30% in both cases when the link load is low (10%-30%). When the link load increases the estimation error grows up to 50%. As it turns out, when the link load is high, the concurrent traffic consumes more tokens from the token-bucket mechanisms, which reduces the accuracy of the estimation. Note that such problem is insurmountable and inherent to the token-bucket mechanism independently of the estimation technique used. Note that, if traffic at each hop is unknown, it is not possible to accurately estimate the MBS as there is no information about the number and size of concurrent packets. Such results call for a measurement strategy where the CIR can be estimated at all times while the MBS estimation should be relegated to low utilization periods, namely night hours or lunch periods.

Concerning the PIR estimation, Figure 15 shows the PIR estimation error in both scenarios —class 1 and 6. As it can be observed the relative error is lower than 10% even in high link-load scenarios.

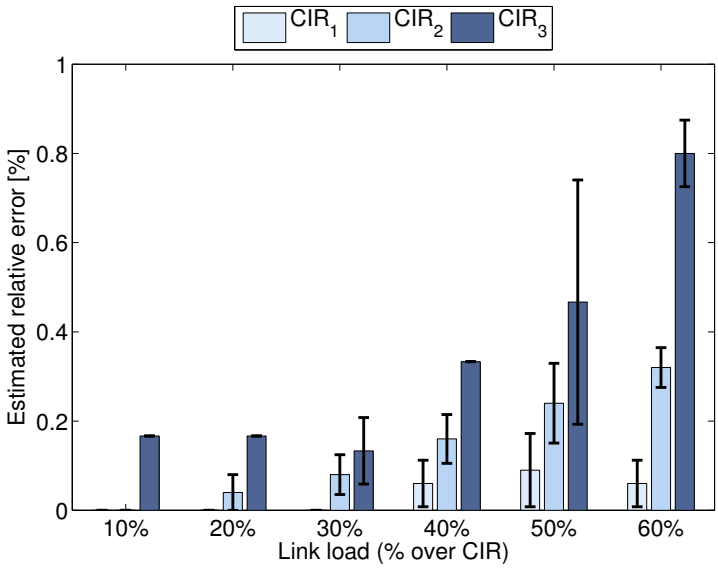


Figure 11: Estimated CIR relative error for multi-hop class 1 with variable cross-traffic. Mean and standard deviation.

6. Conclusions

In this paper a simple, generic and minimal-interfering technique for multi-hop token-bucket parameter estimation has been presented. Additionally, an algorithm (*TBCheck*) has been proposed and validated for typical CIR ranging from 6 Mbps to 800 Mbps and different cross-traffic scenarios. For the sake of completeness, the algorithm has been compared with other single-hop measurement techniques in the state of the art, obtaining the best results both in terms of relative error and intrusiveness even in the presence of cross-traffic.

In case of multi-hop scenarios, a taxonomy of shaping scenarios has been presented and the most typical real-world scenarios have been identified. For each scenario, *TBCheck* proves accurate as it correctly estimates all the parameters when there is no cross-traffic. When cross-traffic is present, the

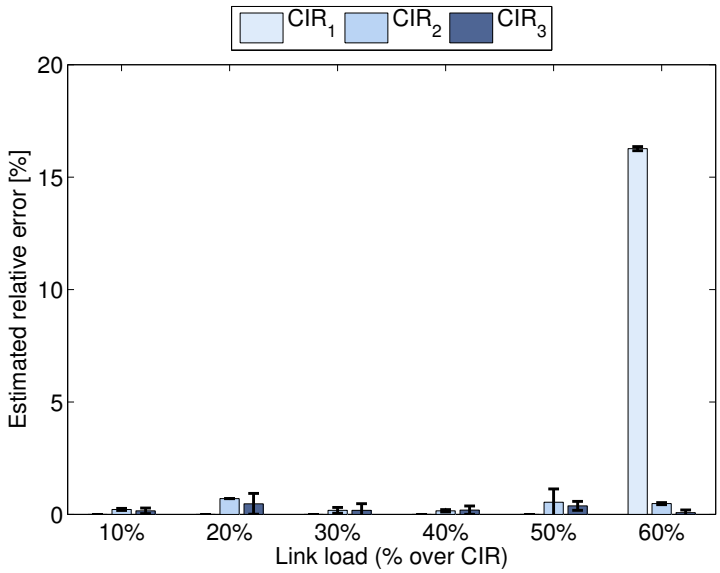


Figure 12: Estimated CIR relative error for multi-hop class 6 with variable cross-traffic. Mean and standard deviation.

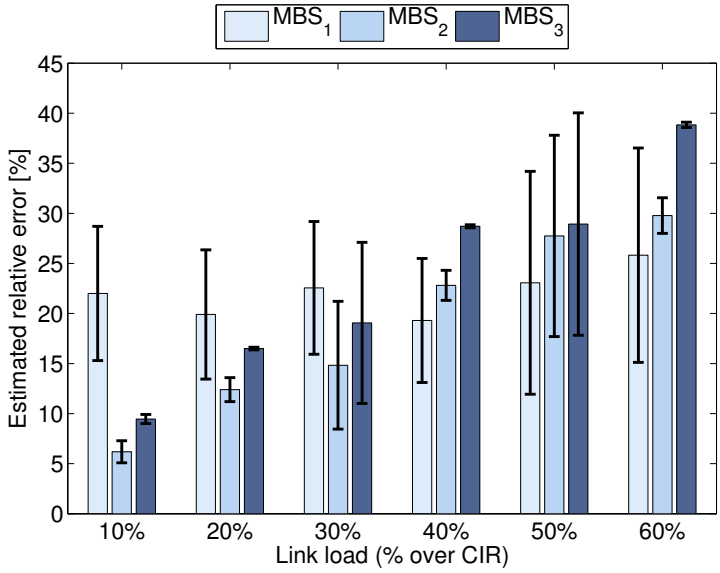


Figure 13: Estimated MBS relative error for multi-hop class 1 with variable cross-traffic. Mean and standard deviation.

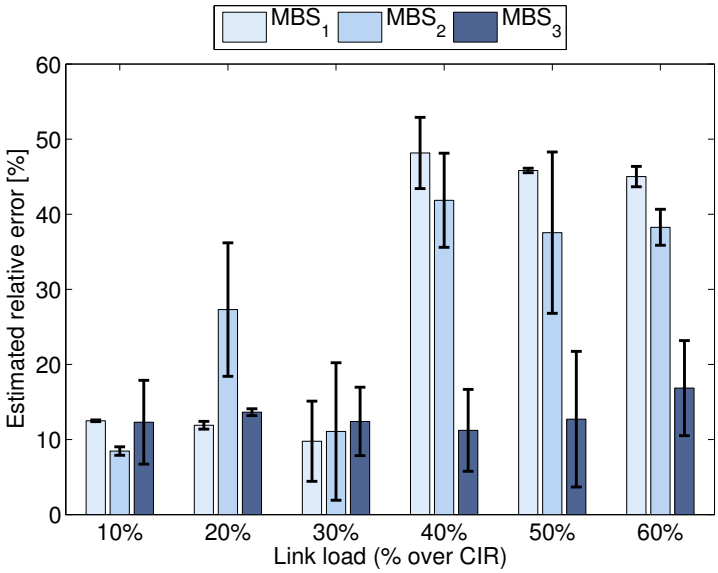


Figure 14: Estimated MBS relative error for multi-hop class 6 with variable cross-traffic. Mean and standard deviation.

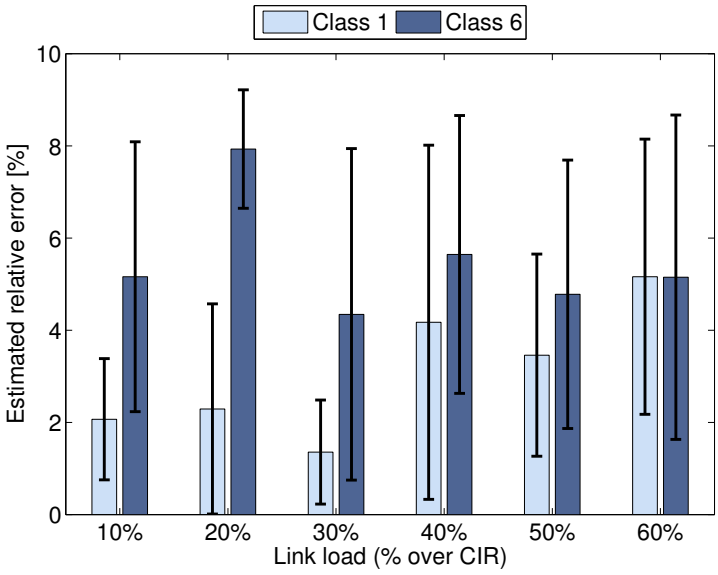


Figure 15: Estimated PIR relative error for multi-hop classes 1 and 6 with variable cross-traffic. Mean and standard deviation.

algorithm is able to estimate the CIR and PIR with a relative error lower than 15% even when the link presents a high load. Concerning the MBS, *TBCheck* attains relative error rates lower than 30% when the link load is below 30%. As it turns out, MBS estimation is unfeasible at high link loads, regardless of the measurement technique, as the bucket will always be empty. Nevertheless, our algorithm accurately estimates the MBS at reasonable utilization levels that allow the token-bucket to fill up eventually.

In the light of the obtained results, we propose a shaping estimation policy for multi-hop scenarios whereby CIR and PIR are estimated during busy hours while MBS is estimated during off-peak hours in order to obtain accurate results. This methodology has been successfully applied to measure the performance parameters of operational deployments in enterprise and large ISPs networks. Furthermore, besides the results presented in this paper, we plan to evaluate the detection and measurement of further shaping algorithms, such as Shaped Round Robin (SRR).

Acknowledgements

This work was partially supported by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund under the project Tráfica (MINECO/FEDER TEC2015-69417-C2-1-R).

References

[1] J. Shi, S.-H. Chung, A traffic-aware quality-of-service control mechanism for software-defined networking-based virtualized networks, International Journal of Distributed Sensor Networks 13 (3) (2017) 1–13.

[2] K. S. Kim, The effect of ISP traffic shaping on user-perceived performance in broadband shared access networks, Computer Networks 70 (2014) 192 – 209.

[3] E. Atxutegi, F. Liberal, E. Saiz, E. Ibarrola, Toward standardized internet speed measurements for end users: current technical constraints, IEEE Communications Magazine 54 (9) (2016) 50–57.

[4] E. Zhang, L. Xu, Capacity and token rate estimation for networks with token bucket shapers, Computer Networks 88 (2015) 1 – 11.

[5] R. Olsson, Pktgen the Linux packet generator, in: Proceedings of the Linux Symposium, Ottawa, Canada, Vol. 2, 2005, pp. 11–24.

[6] C. Dovrolis, P. Ramanathan, D. Moore, What do packet dispersion techniques measure?, in: Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, Vol. 2, 2001, pp. 905–914.

[7] C. Dovrolis, P. Ramanathan, D. Moore, Packet-dispersion techniques and a capacity-estimation methodology, IEEE/ACM Transactions on Networking (TON) 12 (2004) 963–977.

[8] J. Ramos, P. M. Santiago, J. Aracil, J. E. López de Vergara, On the effect of concurrent applications in bandwidth measurement speedometers, Computer Networks 55 (6) (2011) 1435 – 1453.

[9] R. Prasad, M. Jain, C. Dovrolis, Effects of interrupt coalescence on network measurements, in: Proceedings of the 5th International Workshop on Passive and Active network Measurement, PAM, 2004, pp. 247–256.

[10] G. Procissi, A. Garg, M. Gerla, M. Sanadidi, Token bucket characterization of long-range dependent traffic, Computer Communications 25 (11) (2002) 1009–1017.

[11] K. S. Kim, On the excess bandwidth allocation in ISP traffic control for shared access networks, IEEE Communications Letters 18 (4) (2014) 692–695.

[12] L. Farmer, K. S. Kim, Cooperative ISP traffic shaping schemes in broadband shared access networks, in: Proceedings of the 4th International Workshop on Fiber Optics in Access Network (FOAN), 2013, pp. 21–25.

[13] N. Ahmed, X. Lu, L. Barbosa, An efficient parallel optimization algorithm for the token bucket control mechanism, Computer Communications 29 (12) (2006) 2281 – 2293.

[14] M. Marcon, M. Dischinger, K. Gummadi, A. Vahdat, The local and global effects of traffic shaping in the internet, in: Proceedings of the 3rd International Conference on Communication Systems and Networks (COMSNETS).

[15] M. P. R.G. Garroppo, S. Giordano, Estimation of token bucket parameters for aggregated VoIP sources, International Journal of Communication Systems 15 (10) (2002) 851–866.

[16] F. Zafari, F. Humayun, M. Babar, M. Zafar, M. Zuhairi, Performance analysis of a token bucket shaper for MPEG4 video and real audio signal, in: Proceedings of the IEEE International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA), 2013, pp. 1–4.

[17] V. Bashko, N. Melnikov, A. Sehgal, J. Schönwälder, Bonafide: A traffic shaping detection tool for mobile networks, in: IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013, pp. 328–335.

[18] M. Dischinger, M. Marcon, S. Guha, K. P. Gummadi, R. Mahajan, S. Saroiu, Glasnost: Enabling End Users to Detect Traffic Differentiation, in: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI, 2010, pp. 405–418.

[19] P. Kanuparthy, C. Dovrolis, ShaperProbe: end-to-end detection of ISP traffic shaping using active methods, in: Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, IMC, 2011, pp. 473–482.

[20] D. Moltchanov, Automatic bandwidth adjustment for content distribution in mpls networks, Advances in Multimedia 2008 (2) (2008) 3:1–3:15.

[21] Q. Yin, J. Kaur, F. Smith, Scaling bandwidth estimation to high speed networks, in: Proceedings of the 15th International Conference on Passive and Active network Measurement, PAM, 2014, pp. 258–261.

[22] M. Ruiz, J. Ramos, G. Sutter, J. E. L. de Vergara, S. López-Buedo, J. Aracil, Accurate and affordable packet-train testing systems for multi-gigabit-per-second networks, IEEE Communications Magazine 54 (3) (2016) 80–87.

[23] A. Saeed, N. Dukkipati, V. Valancius, V. The Lam, C. Contavalli, A. Vahdat, Carousel: Scalable traffic shaping at end hosts, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17, ACM, 2017, pp. 404–417.

[24] B. Lantz, B. Heller, N. McKeown, A network in a laptop: rapid prototyping for software-defined networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, p. 19.

[25] J. Sommers, P. Barford, B. Eriksson, On the prevalence and characteristics of MPLS deployments in the open internet, in: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC '11, ACM, 2011, pp. 445–462.

[26] E. T. S. Institute, ETSI guide 202 057-4: Speech Processing, Transmission and Quality Aspects (STQ);User related QoS parameter definitions and measurements;Part 4: Internet access, 2008.

[27] W. Zhang, J. He, Modeling end-to-end delay using pareto distribution, in: Proceedings of the 2nd International Conference on Internet Monitoring and Protection, ICIMP, 2007.

[28] B. Fortz, J. Rexford, M. Thorup, Traffic engineering with traditional IP routing protocols, IEEE Communications Magazine 40 (10) (2002) 118–124.