On the Stationarity of TCP Bulk Data Transfers

G. Urvoy-Keller

Institut Eurecom, 2229, route des crêtes, 06904 Sophia-Antipolis, France
urvoy@eurecom.fr

Abstract. While the Internet offers a single best-effort service, we remark that (i) core backbones are in general over provisioned, (ii) end users have increasingly faster access and (iii) CDN and p2p solutions can mitigate network variations. As a consequence, the Internet is to some extent already mature enough for the deployment of multimedia applications and applications that require long and fast transfers, e.g. software or OS updates. In this paper, we devise a tool to investigate the stationarity of long TCP transfers over the Internet, based on the Kolomogorov-Smirnov goodness of fit test. We use BitTorrent to obtain a set of long bulk transfers and test our tool. Experimental results show that our tool correctly identify noticeable changes in the throughput of connections. We also focus on receiver window limited connections to try to relate the stationarity observed by our tool to typical connection behaviors.

1 Introduction

The current Internet offers a single best-effort service to all applications. As a consequence, losses and delay variations are managed by end-hosts. Applications in the Internet can be classified into two classes: elastic applications, e.g. web or e-mail, that can tolerate throughputs and delays variations; and real time applications, that are delay sensitive (e.g. voice over IP) or throughput sensitive (e.g. video-on-demand). With respect to the above classification, a common belief is that the current Internet with its single best-effort service requires additional functionality (e.g. DiffServ, MPLS) to enable mass deployment of real-time applications. Still, a number of facts contradict, or at least attenuate, this belief: (i) recent traffic analysis studies have deemed the Internet backbone ready to provide real-time services [15]; (ii) the fraction of residential users with high speed access, e.g. ADSL or cable, increases rapidly; (iii) network-aware coding schemes, e.g. mpeg4-fgs [7], combined with new methods of transmission like peer-to-peer (p2p) techniques, e.g. Splitstream [5], have paved the way toward the deployment of real-time applications over the Internet.
The above statements have lead us to investigate the variability of the service provided by the Internet from an end connection point of view. As TCP is carrying most of the bytes in the Internet [10], our approach is to concentrate on long lived TCP connections. Bulk data transfers represent a significant portion of the current Internet traffic load, especially with p2p applications [2]. By analyzing bulk data transfers, we expect to better understand the actual interaction

between TCP and the Internet. This is important for future applications [1] and also for CDN providers that rely on migrating traffic on the "best path" from central to surrogate servers [8]. CDN providers generally rely on bandwidth estimation tools, either proprietary or public tools [12] to perform path selection. However, the jury is still out on the stationarity horizon provided by such tools, i.e. how long will the estimation provided by the tool remain valid or at least reasonable. In the present work, we propose and evaluate a tool that should help solving these issue. The rest of this paper is organized as follows. In Section 2, we review the related work. In Section 3, we present our dataset. In Section 4, we present our tool to extract stationarity periods in a given connection. In Section 5, we discuss results obtained on our dataset. Conclusions and future work directions are presented in Section 6.

2 Related Work

Mathematically speaking, a stochastic process $X(t)$ is stationary if its statistical properties (marginal distribution, correlation structure) remain constant over time.
Paxson et al. [17] have studied the stationarity of the throughput of short TCP connections (transfers of 1Mbytes) between NIMI hosts. The major difference between this work and the present work is that we consider long bulk data transfer (several tens of minutes) and our dataset is (obviously) more recent with hosts with varying access capacity, whereas NIMI machines consistently had good Internet connectivity. Other studies [4, 13] have concentrated on the non stationarity observed on high speed link with a high number of aggregated flows. They studied the time scales at which non stationarity appears and the causes behind it. Also, recently, the processing of data streams has emerged as an active domain in the database research community. The objective is to use database techniques to process on-line stream at high speed (e.g. Internet traffic on a high speed link). In the data stream context, detection of changes is a crucial task [14, 3].

3 Dataset

Our objective is to devise a tool to assess the stationarity of TCP bulk data transfers. To check the effectiveness of the tool, we need to gather samples, i.e. long TCP transfers, from a wide set of hosts in the Internet. A simple way to attract traffic from a variety of destinations around the world is to use a p2p application. As we are interested in long data transfers, we used BitTorrent, a popular file replication application [11]. A BitTorrent session consists in the

[1] Our focus in the present work is on throughput, which is an important QoS metrics for some multimedia applications, e.g. VoD, but arguably not all multimedia applications, a typical counter-example being VoIP.

replication of a single large file on a set of peers. BitTorrent uses specific algorithms to enforce cooperations among peers. The data transfer phase is based on the swarming technique where the file to be replicated is broken into chunks (typical chunk size is 256 kbytes) that peers exchange with one another. The BitTorrent terminology distinguishes between peers involved in a session that have not yet completed the transfer of the file, which are called *leechers* and peers that have already completed the transfer, which are called *seeds*. Seeds remain in the session to serve leechers. Connections between peers are permanent TCP connections. Due to the BitTorrent algorithms [11], a typical connection between two hosts is a sequence of on periods (data transfers) and off periods (where only keep-alive messages are transfered). Figure 1, where y axis values are one second throughputs samples, depicts a typical one way connection of approximately 14 hours with clear on and off phases .

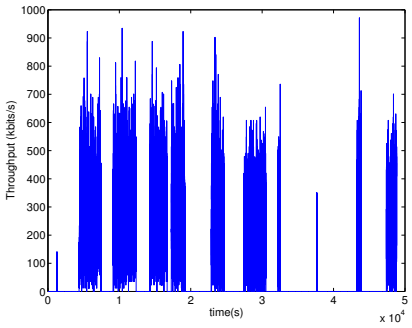

Fig. 1. A typical (one-way) BitTorrent connection

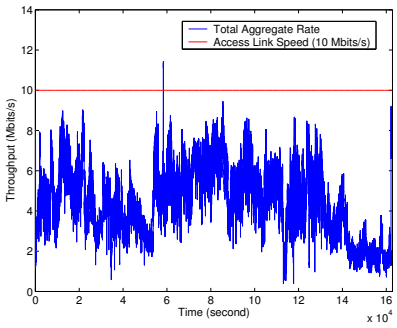

Fig. 2. Aggregate rate of the BitTorrent application during the experiment

The dataset we have collected consists of connections to about 200 peers that were downloading (part of) the file (latest Linux Mandrake release) from a seed located at Eurecom. More precisely, a tcpdump trace of 10 Gbytes was generated during a measurement period of about 44 hours. While the 200 connections are all rooted at Eurecom, the 10 Mbits/s access link of Eurecom should not constitute a shared bottleneck for two reasons. First, with BitTorrent, a client (leecher or seed) does not send to all its peers simultaneously but only to 4 of them, for sake of efficiency. Second, the total aggregate throughput remains in general far below the 10 Mbits/s as shown in figure 2 while the average traffic generally observed on this link (to be added to the traffic generated by our BitTorrent client to obtain the total offered load for the link) exhibits an average rate around 1 Mbits/s with a peak rate below 2 Mbits/s.
To illustrate the diversity of these 200 peers, we have used the maxmind service (http://www.maxmind.com/) to assess the origin country of the peers. In table 1, we ranked countries based on the peers that originate from each of them. Unsurprisingly, we observe a lot of US peers (similar observation was made in [11] for a

similar torrent, i.e. Linux Redhat 9.0) while the other peers are distributed over a wide range of 27 countries (see http://encyclopedia.thefreedictionary.com/ISO%203166-1 for the meaning of the abbreviations used in table 1).

Our objective is to study long bulk data transfers in the Internet. To obtain

Country	# peers	Country	# peers	Country	# peers	Country	# peers
US	87	NL	4	BR	2	YU	1
UK	24	DE	3	LT	2	BE	1
CA	14	AU	3	CN	1	AT	1
FR	12	PE	3	NO	1	ES	1
IT	8	AE	3	SI	1	CH	1
SE	8	CL	2	TW	1		
PL	7	PT	2	CZ	1		

Table 1. Origin countries of the 200 peers

meaningful samples, we extracted the on periods from the 200 connections, resulting in a total of 399 flows. The algorithm used to identify off-periods is to detect periods of at least 15 seconds where less than 15 kbytes of data are sent, as BitTorrent clients exchange keep-alive messages at a low rate (typically less than 1000 bytes per second) during periods where no data transfer is performed. We further restricted ourselves to the 184 flows whose duration is higher than 1600 seconds ($\sim$ 26.6 minutes), for reasons that will be detailed in section 4. We call flow or initial flow an on-period and stationary flow a part of a flow that is deemed stationary. For each flow, we generate a time series that represents the throughput for each 1 second time interval. The average individual throughput of these 184 flows is quite high, 444 kbits/s. Overall, these flows correspond to the transfer of about 50 Gbytes of data over a cumulated period of about 224 hours (the flows of duration less than 1600 seconds represent about 14 Gbytes of data). Due to its size, we cannot claim that our dataset is representative of the bulk transfers in the Internet. It is however sufficiently large to demonstrate the effectiveness of our tool. It also shows that BitTorrent is a very effective application to collect long TCP transfers from a variety of hosts in terms of geographical location and access link speed (even if it is unlikely to observe clients behind modem lines, as downloading large file behind a modem line is unrealistic).

4 Stationarity Analysis Tool

4.1 Kolmogorov-Smirnov (K-S) test

Given two i.i.d samples $X_1(t)_{t\in\{1,...n\}}$ and $X_2(t)_{t\in\{1,...n\}}$, the Kolmogorov-Smirnov test enables us to determine whether the two samples are drawn from the same distributions or not. The test is based on calculating the empirical cumulative distribution functions of both samples and evaluating the absolute maximum

difference $D_{\max}$ between these two functions. The limit distribution of $D_{\max}$ under the null hypothesis (X_1 and X_2 drawn from the same distribution) is known and thus $D_{\max}$ is the statistics the test is built upon. In the sequel of this paper, we used the matlab implementation of the K-S test with 95% confidence levels.

4.2 K-S test for change point detection

Our objective is to detect stationary regions in time series, or equivalently to detect change points (i.e. border points between stationary regions). We used the K-S test to achieve this goal. Previous work as already used the K-S test to detect changes [9, 3], though not in the context of traffic analysis.
The basic idea behind our tool is to use two back-to-back windows of size w sliding along the time series samples and applying the K-S test at each shift of the windows. If we assume a time series of size n, then application of the K-S test leads to a new binary time series of size $n - 2w$, with value zero whenever the null hypothesis could not be rejected and one otherwise. The next step is to devise a criterion to decide if a '1' in the binary time series corresponds to a false alarm or not. Indeed, it is possible to show that even if all samples originate from the same underlying distribution, the K-S test (or any other goodness of fit test [16]) can lead to spurious '1' values. The criterion we use to deem detection of a change point is that at least $w_{\min} \approx \frac{w}{2}$ consecutive ones must be observed in the binary time series. $w_{\min}$ controls the sensitivity of the algorithm. The intuition behind setting $w_{\min}$ to a value close to $\frac{w}{2}$ is that we expect the K-S test to almost consistently output '1' from the moment when the right-size window contains about 25% of points from the "new" distribution (the distribution after the change point) up to the moment when the left-size window contains about 25% of points from the "old" distribution. In practice, a visual inspection of some samples revealed that using such values for $w_{\min}$ allows to correctly detect obvious changes in the time series. Figure 3 presents an example on one of our TCP flows time series (aggregated at a 10 seconds time scale - see next section for details) along with the scaled binary time series output by the tool and the change points (vertical bars). This example illustrates the ability of the test to isolate stationary regions. Note also that the output of the binary time series that represents the output of the K-S test for each window position (dash line in figure 3) exhibits a noticeable consistency. This is encouraging as oscillations in the output of the test would mean that great care should be taken in the design of the change point criterion. As this is apparently not the case, we can expect our simple criterion ($w_{\min}$ consecutive '1' values to detect a change) to be effective.

4.3 K-S test in the presence of correlation

We want to apply the K-S change point tool described in the previous section to detect changes in the throughput time series described in section 3. However, we have to pay attention that, due to the close loop nature of TCP, consecutive

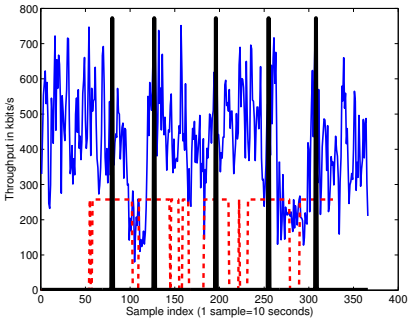

Fig. 3. Initial time series (thin line), binary time series (dash line) and change points (thick bars)

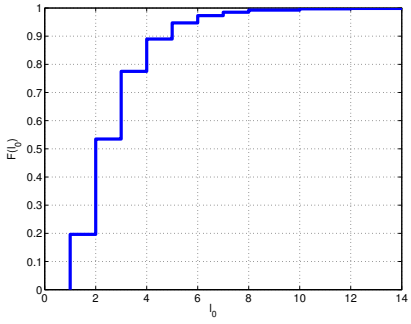

Fig. 4. Cumulative distri. function of l_0

one-second throughputs samples are correlated[2]. If all samples are drawn from the same underlying distribution, a simple heuristic to build an uncorrelated time series out of a correlated time series is to (i) compute the auto-correlation function of the initial time series, (ii) choose a lag l_0 at which correlation is close enough to zero and (iii) aggregate the initial time series over time intervals of size l_0. Specifically, let $X(t)_{t\in\{1,\ldots n\}}$ be the initial time series. Its auto-correlation function is $AC(f) = \frac{\sum_{i=1}^{n-f} \bar{X}(i+f)\bar{X}(i)}{n\sigma^2_{\bar{X}}}$, where $\bar{X}(t) \triangleq X(t) - E[X]$ and $\sigma^2_{\bar{X}}$ is the variance of $\bar{X}$. $AC(f)$ measures the amount of correlation between samples located at positions t and $t+f$. If ever the time series is i.i.d., then $|AC(f)|$ should be upper bounded by $\frac{2}{\sqrt{n}}$ for $f > 1$ [6]. For a correlated time series, we can choose l_0 such that $\forall f > l_0, |AC(f)| \leq \frac{2}{\sqrt{n}}$. We then generate the aggregate time series $Y(t)_{t\in\{1,\ldots\lceil\frac{n}{l_0}\rceil\}}$ where $Y(t) = \frac{\sum_{u=t\times l_0+1}^{(t+1)\times l_0} X(u)}{l_0}$. This method is however not applicable to our TCP time series as changes in the network conditions prevent us from assuming the same underlying distributions over the whole duration of a flow.

To overcome this difficulty and be able to use the K-S test, we aggregate each time series at a fixed value of $l_0 = 10$. This means that we average the initial time series over intervals of 10 seconds. As the average throughput of the flows is 444 kbits/s, an average flow will send more than 400 packets (of size 1500 bytes) in a 10 second time interval, which is reasonably large enough for a TCP connection to have lost memory of its past history (e.g. to have fully recovered from a loss). To assess the level of correlation that persists in the time series after aggregation at the 10 second time scale, we have computed, for each stationary interval obtained with our tool, the autocorrelation function of the process in this interval. We then derive the lag l_0 after which the autocorrelation function

[2] While correlation and independence are not the same, we expect that removing correlation will be sufficient in our context to obtain some almost independent samples.

remains (for 95% of the cases) in the interval $\left[-\frac{2}{\sqrt{n}}, \frac{2}{\sqrt{n}}\right]$. Figure 4 represents the cumulative distribution function of l_0. We notice that about 95% of the l_0 values are below 5, which indicates that the "remaining" correlation is of short term kind only.

Based on the result of figure 4, one could however still argue that we should continue further the aggregation of the time series for which the correlation is apparently too large, say for $l_0 \geq 3$. Note however that the choice of the time scale at which one works directly impacts the separation ability of the K-S test. Indeed, as we use windows of w samples, a window corresponds to a time interval of $10 \times w$ seconds, and we won't be able to observe stationary periods of less than $10 \times w$ seconds. For example, the results presented in section 5 are obtained with $w = 40$, which means that we won't be able to observe stationary periods of less than 400 seconds (~ 6.7 minutes). Thus, there exists a trade-off between the correlation of the TCP throughput time series that calls for aggregating over large time intervals and the separation ability of the test that calls for having as much small windows as possible.

A second reason why we have chosen to aggregate at a fixed 10 second time scale value is that we expect our tool to be robust in the presence of short term correlation. We investigate this claim in the next section, on synthetic data, where we can tune the amount of correlation. While by no means exhaustive, this method allows us to obtain insights on the behavior of K-S test in the presence of correlation.

4.4 Test of the robustness of the tool with synthetic data

We consider a first-order auto-regressive process X with $X(t) = aX(t-1) + Z(t), \forall t\{1, \ldots n\}$ where Z is a purely random process with a fixed distribution. We choose two distributions for Z (leading to Z_1 and Z_2) to generate two samples $X_1(t)$ and $X_2(t)$. We then form the compound vector $[X_1(t)X_2(t)]$ and apply the K-S change point test. We can vary the a parameter to tune the amount of correlation and test how the K-S change point test behaves. Specifically, we consider $a \in \{0.2, 0.5, 0.9\}$ as these values roughly correspond to l_0 values (as defined in the previous section) equal respectively to 2, 5 and 20. With respect to the results presented in figure 4, we expect the K-S test to behave properly for $a \leq 0.5$ (i.e. $l_0 \leq 5$). In table 2, we present results obtained when $Z_1(t)$ and $Z_2(t)$ are derived from normal distributions with respective means and variances $(0.4, 0.3)$ and $(1.5, 1.5)$ where a given sample $Z_1(t)$ (resp. $Z_2(t)$) is obtained by averaging 10 independent samples drawn from the normal distribution with parameters $(0.4, 0.3)$ (resp. $(1.5, 1.5)$). The main idea behind this averaging phase is to smooth $X_1(t)$ and $X_2(t)$ in a similar fashion that the throughput samples are smoothed at a 10 second time scale in the case of our BitTorrent dataset. As the transition between $X_1(t)$ and $X_2(t)$ is sharp thanks to the difference in mean between Z_1 and Z_2, we expect that the change point tool will correctly detect it. Now, depending on the correlation structure, it might happen that more change points are detected. This is reflected by the results presented in table 2, where for

different values of a, w and $w_{\min}$, we compute over 1000 independent trajectories, the average number of detections made by the algorithm (without false alarm, we should obtain 1) and the percentage of cases for which a change is detected in the interval $[450, 550]$ that corresponds to the border between $X_1(t)$ and $X_2(t)$ in the compound vector $[X_1(t)X_2(t)]$, each vector having a size of 500 samples. When the latter metric falls below 100%, it indicates that the correlation is such that our tool does not ncessarily notice the border between $X_1(t)$ and $X_2(t)$ any more. From table 2, we note that such a situation occurs only for $a = 0.9$. Also, when the amount of correlation increases, the average number of points detected increases dramatically, as the correlation structure of the process triggers false alarms as illustrated by the trajectory depicted in figure 5. For a given w value, increasing the treshold $w_{\min}$ helps reducing the rate of false. Note that while the results obtained here on synthetic data seem to be better for a criterion $w_{\min} = 40$, we used $w_{\min} = 15$ on our dataset as it was giving visually better results. A possible reason is the small variance of the throughput time series as compared to the corresponding mean for our dataset. More generally, we note that tuning w and $w_{\min}$ is necessary to tailor the tool to the specific needs of a user or an application.

a	w	$w_{\min}$	% of cases with one detection in $[450, 550]$	Average number of detections
0.2	40	15	100	2.4
0.2	40	40	100	1
0.2	80	15	100	2.4
0.2	80	40	100	1.2
0.5	40	15	100	5.6
0.5	40	40	100	1.1
0.5	80	15	100	4.5
0.5	80	40	100	1.9
0.9	40	15	100	14.6
0.9	40	40	99	6.5
0.9	80	15	89.9	8
0.9	80	40	90.7	5.6

Table 2. Change point detection tool performance in the presence of correlation

4.5 Empirical validation on real data

For the results obtained in this section and the rest of the paper, we used $w = 40$ as special care must be taken when using the K-S test for smaller values [16]. Also, we consider $w_{\min} = 15$ as it visually gives satisfying results on our dataset. In addition, to obtain meaningful results, we restrict the application of the tool to time series with at $4 \times w$ samples (the tool will thus output at least $2 \times w$

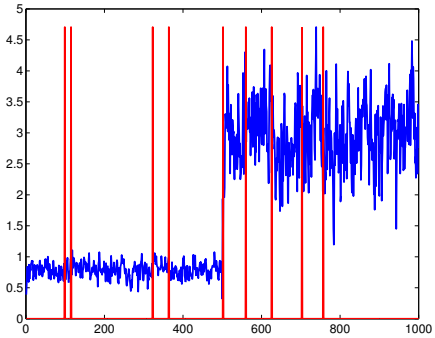

Fig. 5. Sample trajectory of $[X_1 X_2]$ with the detected change points (vertical bars) for $w = 40$ and $w_{\min} = 15$

results), i.e. to flows that last at least 1600 seconds.
Our change point analysis tool can be easily validated with synthetic data. However, we need to further check whether the results obtained on real traces are reasonable or not. We thus applied our tool on our 184 flows to obtain 818 stationary flows. To assess the relevance of the approach, we proceeded as follows: for any two neighboring stationary flows from the same flow, we compute their means μ_1 and μ_2 and their standard deviations σ_1 and σ_2. We then compute the "jump in mean" $\Delta_\mu = \frac{\mu_2 - \mu_1}{\mu_1} \times 100$ and "jump in standard deviation" $\Delta_\sigma = \frac{\sigma_2 - \sigma_1}{\sigma_1} \times 100$. We then break each stationary flows into two sub-flows of equal size and compute their means μ_1^i and μ_2^i and standard deviations σ_1^i and σ_2^i $(i = 1, 2)$. We can then define jumps in means and standard deviations between two sub-flows of a given stationary flow. The latter jumps are called intra jumps while the jumps between stationary flows are called inter jumps. The idea behind these definitions is to demonstrate that the distributions of intra jumps are more concentrated around their mean value than the distributions of inter jumps. To compare those distributions, we used boxplot representations. A boxplot of a distribution is a box where the upper line corresponds to the 75 percentile $\hat{p}_{0.75}$ of the distribution, the lower line to the 25 percentile $\hat{p}_{0.25}$ and the central line to the median. In addition, the $\hat{p}_{0.25} - 1.5 \times IQR$ and $\hat{p}_{0.75} + 1.5 \times IQR$ values ($IQR = \hat{p}_{0.75} - \hat{p}_{0.25}$ is the inter quantile range, which captures the variability of the sample) are also graphed while the samples falling outside these limits are marked with a cross. A boxplot allows to quickly compare two distributions and to assess the symmetry and the dispersion of a given distribution. In figure 6, we plotted the boxplots for the inter jump in mean (left side) and intra jump in mean (right side). From these representations, we immediately see that the intra jump distribution is thinner than the inter jumps distribution which complies with our initial intuition. Note also that the means of the inter and intra jump distributions are close to zero as the Δ_μ definition can result in positive or negative values and it is quite reasonable that overall, we observe as much positive

as negative jumps. Figure 7 depicts the boxplots for the inter and intra jumps in standard deviations. The results are somehow similar to the ones for jumps in mean although less pronounced and more skewed toward large positive values.

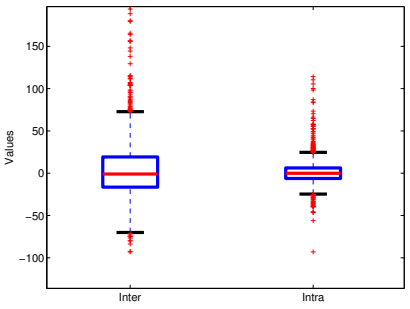

Fig. 6. Boxplot representation of the inter jump in mean (left side) and intra jump in mean (right side)

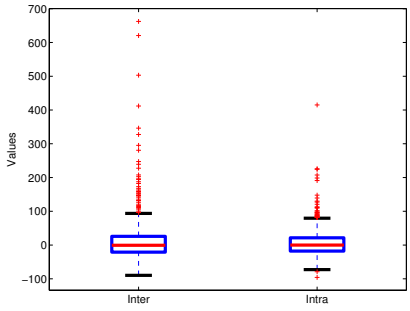

Fig. 7. Boxplot representation of the inter jump in standard deviation(left side) and intra jump in standard deviation (right side)

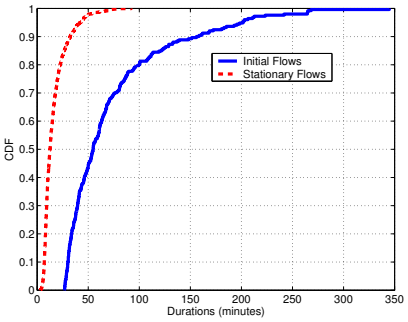

Fig. 8. CDFs of flows and stationary flows durations

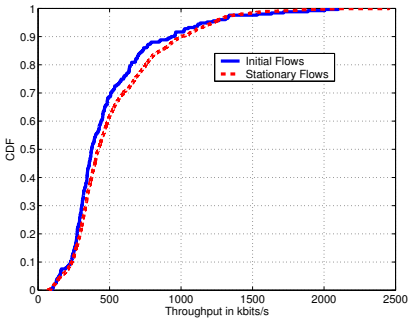

Fig. 9. CDFs of flows and stationary flows throughputs

5 Results on the BitTorrent Dataset

5.1 Stationary periods characterization

As stated in the previous section, the K-S change point tool has extracted 818 stationary flows out of the 184 initial flows. This means that, on average, a flow is cut into 4.45 stationary flows. Figure 8 represents the cumulative distribution functions (cdf) of the duration of stationary and initial flows. Stationary flows have an average duration of 16.4 minutes while initial flows have an average duration of 73 minutes.
Figure 9 represents the cumulative distribution functions of throughputs of the

stationary and initial flows. Overall, stationary flows tend to exhibit larger throughputs than initial flows. Indeed, the mean throughput of stationary flows is 493.5 kbits/s as compared to 444 kbit/s for the initial ones. This discrepancy is an indication that the K-S change point test is working properly as it extracts from the initial flows stationary periods where the throughputs significantly differ from the mean throughput of the flow. The cdfs differ at the end because whenever the K-S test exhibits a small period (relative to the flow it is extracted from) with high throughput, it will become one sample for the cdf of stationary flows, whereas it might have little impact for the corresponding sample for the cdf of the initial flows (if the high throughput part only corresponds to a small fraction of the initial flow).
Using our tool, we can also investigate transitions between consecutive stationary periods. The left boxplot of figure 6 allows us to look globally at transitions between stationary periods. From this figure, we can observe that most of the changes result in jumps of the mean value that are less than 20% in absolute values. This is encouraging for applications that can tolerate such changes in their observed throughput since they can expect to experience quite long stable periods, typically several tens of minutes (at least in the context of our dataset). However, a lot of values fall outside the plus or minus $1.5 \times IQR$ interval, meaning that some transitions are clearly more sharp than others.

5.2 The case of receiver window limited connections

In a effort to relate the stationarity observed by our tool to the intrinsic characteristics of the connections, we considered the case of receiver window limited flows. A receiver window limited flow is a flow whose throughput is limited by the advertised window of the receiver. The motivation behind this study is that as receiver window limited flows are mostly constrained by some end hosts characteristics (the advertised window of the receiver), they should exhibit longer stationary periods than other flows. Indeed, the intuition is that those other flows have to compete "more" for resources along their path with side traffic, which should affect their throughput, leading to change points.
We first have to devise a test that flags receiver window limited flows. We proceed as follows. For each flow, we generate two time series with a granularity of 10 seconds. The first time series, $Adv(t)$ represents the advertised window of the receiver while the second one, $Out(t)$ accounts for the difference between the maximum unacknowledged byte and the maximum acknowledged byte. The second time series provides an estimate of the number of outstanding bytes on the path at a given time instant. The $Out(t)$ time series is accurate except during loss periods. Note that the computation of $Out(t)$ is possible since our dataset was collected at the sender side, as the Eurecom peer in the BitTorrent session was acting as a seed during the measurement period. A flow is then flagged receiver window limited if the following condition holds:

$$\frac{\sum_{t=1}^{N} 1_{Adv(t)-3\times MSS \leq Out(t) \leq Adv(t)}}{N} \geq 0.8$$

where N is the size of the two time series and MSS is the maximum segment size of the path. The above criterion simply states that 80% of the time, the estimated number of outstanding packets must lie between the advertised window minus three MSS and the advertised window. By choosing a treshold of 80%, we expect to be conservative.
Application of the test on our dataset leads us to flag about 13.7% of the flows as receiver window limited. The next issue is to choose the non window limited flows. We adopt the following criterion:

$$\frac{\sum_{t=1}^{N} 1_{Out(t) \leq Adv(t) - 3 \times MSS}}{N} \geq 0.9$$

Applying the above criterion, we obtained about 14.4% of non receiver window limited flows. A straightforward comparison of the durations of the stationary flows extracted from the flows of the two families (receiver window limited and non receiver limited) is misleading as the duration of their respective connections is different. We thus use two other metrics. First, we compute the number of stationary flows into which a flow is cut in each family. We obtain that the receiver window limited flows are on average cut into 3.5 stationary flows while non receiver window limited flows are cut into 4.5 stationary flows. The second metric we consider is the relative size, in percentage, of the stationary flows with respect to the size of flow they are extracted from for the two familied. Figure 10 represents the cumulative distribution functions of the percentages for the two families. From this figure, we observe that receiver window limited stationary flows are relatively larger than non receiver window limited ones in most cases. Also, in figure 11, we plot the cumulative distributions of the throughput of the stationary flows for both families. We conclude from figure 11 that receiver window limited stationary flows exhibit significantly smaller throughputs values than non receiver window limited ones. This might mean that receiver limited flows correspond to paths with larger RTT than non receiver window limited ones, as this would prevent these flows from achieving high throughput values. This last point as well as our definition of window limited flows (we only considered around 28% of the flows of our dataset to obtain those results) would clearly deserve more investigation.

6 Conclusion and Outlook

Internet Traffic analysis becomes a crucial activity, e.g. for ISPs to do troubleshooting or for content providers and researchers that are willing to devise new multimedia services in the Internet. Once information on some path has been collected, its needs to be analyzed. The first step is to divide traces into somewhat homogeneous period and to flag anomalies. In this paper, we concentrate on the analysis of the service perceived by long TCP connections in the Internet. We have developed a change point analysis tool that extracts stationary periods within connections. We follow a non parametric approach and based our tool on the Kolmogorov-Smirnov goodness of fit test. We validated our change

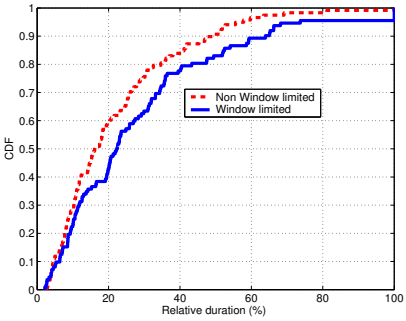

Fig. 10. Histogram of relative size of rec. window and non rec. window limited stationary flows

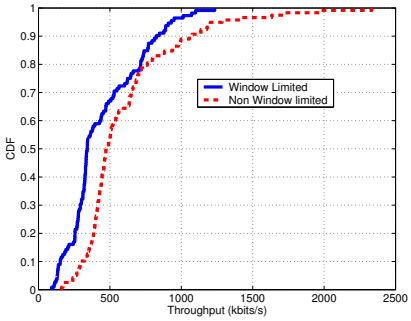

Fig. 11. Histogram of rec. window and non rec. window limited stationary flows throughputs

point tool in various ways on synthetic and operational datasets. Overall, the tool manages to correctly flag change points as long as little correlation persists at the time scale at which it is applied. We worked at the 10 second time scale, which is a reasonable time scale for some multimedia applications such as VoD. We also focused on receiver window limited connections to relate the stationarity observed by our tool to typical connection behaviors.
As future work, we intent to pursue in this direction by correlating the stationarity periods with some other network events like RTT variations or loss rates. We would also like to study the extent to which our tool could be used in real time and to investigate how it could be tailored to the need of some specific applications. It is also necessary to compare our tool with some other change point techniques [1].

Acknowledgment

The author is extremely grateful to the anonymous reviewers for their valuable comments and to M. Siekkinen for the trace collection and time series extraction.

References

1. M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes - Theory and Application*, Prentice-Hall, Inc. ISBN 0-13-126780-9, 1993.
2. N. Ben Azzouna, F. Clerot, C. Fricker, and F. Guillemin, "Modeling ADSL traffic on an IP backbone link", *Annals of Telecommunications*, December 2004.
3. S. Ben-David, J. Gehrke, and D. Kifer, "Detecting changes in data streams", In *Proceedings of the 30th International Conference on Very Large Databases*, 2004.
4. J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "On the nonstationarity of Internet traffic", In *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 102–112, ACM Press, 2001.

5. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in a cooperative environment", In *Proceedings of SOSP'03*, New York, USA, October 2003.
6. C. Chatfield, *The analysis of time series - An introduction*, Chapman & Hall, London, UK, 1996.
7. P. De Cuetos, P. Guillotel, K. Ross, and D. Thoreau, "Implementation of Adaptive Streaming of Stored MPEG-4 FGS Video over TCP", In *International Conference on Multimedia and Expo (ICME02)*, August 2002.
8. J. Dilley, B. Maggs, J. Parikh, H. Prokop, and R. Sitaraman, and B. Weihl, "Globally distributed content delivery", *Internet Computing, IEEE*, pp. 50–58, Sept.-Oct 2002.
9. H. Eghbali, "K-S Test for Detecting Changes from Landsat Imagery Data", *IEEE Trans Syst., Man & Cybernetics*, 9(1):17–23, January 1979.
10. M. Fomenkov, K. Keys, D. Moore, and k claffy, "Longitudinal study of Internet traffic from 1998-2003", Cooperative Association for Internet Data Analysis - CAIDA, 2003.
11. M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garcés-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime", In *Passive and Active Measurements 2004*, April 2004.
12. M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput", *IEEE/ACM Transactions on Networking*, 11(4):537–549, 2003.
13. T. Karagiannis and et al., "A Nonstationary Poisson View of Internet Traffic", In *Proc. Infocom 2004*, March 2004.
14. B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications", In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 234–247, ACM Press, 2003.
15. A. Markopoulou, F. Tobagi, and M. J. Karam, "Assessing the quality of voice communications over Internet backbones", *IEEE/ACM Transactions on Networking*, 11:747–760, October 2003.
16. S. Siegel and N. J. Castellan, *Nonparametric statistics for the Behavioral Sciences*, McGraw-Hill, 1988.
17. Y. Zhang, V. Paxson, and S. Shenker, "The Stationarity of Internet Path Properties: Routing, Loss, and Throughput", ACIRI, May 2000.