Profiling TCP Traffic in Optical Burst Switching Networks

Kostas Ramantas and Kyriakos Vlachos

Computer Engineering and Informatics Department & Research Academic Computer Technology Institute, University of Patras, Rio, Greece
{ramantas,kvlachos}@ceid.upatras.gr

Abstract. The efficient transmission of TCP traffic over OBS networks is a challenging problem, due to the high sensitivity of TCP congestion control mechanism to losses. In this paper, a traffic profiling scheme is proposed for gathering TCP flow statistics, which are used to estimate steady-state performance of TCP traffic over OBS networks. The burst assembly unit, in parallel to the assembly process, can inspect TCP traffic, keeping traffic statistics that can be used for throughput estimations. In this paper, we detail the traffic profiling and estimation mechanism and also provide analytical and simulation results to assess its performance.

Keywords: TCP profiling, traffic measurement, Optical Burst Switching, Burst Assembly.

1 Introduction

The transmission of TCP traffic over OBS [1], has been extensively studied in the related literature. Various burst assembly and burst scheduling algorithms have been proposed [2] to enhance the efficient transmission of TCP over OBS networks, but it still remains an open problem, since the (relatively) high burst loss ratio experienced in OBS networks is incompatible with TCP congestion control mechanism. It has been observed that burst losses have a significant impact on the TCP end-to-end performance, as shown in [3], [4]. In particular, TCP transmission over OBS networks suffers from the high number of segments that are lost, upon a single burst drop. This typically results in many sources timing out and which will subsequently enter a slow start phase. This may also result in synchronizing TCP transmissions with an imminent effect on link utilization, [5]. Further, the introduction of an unpredictable delay, challenges the window mechanism used by TCP protocol for congestion control. In this work, we focus on TCP flow profiling in OBS networks, which can provide insights regarding user-perceived performance, and serve as a basis for capacity planning.

TCP flow profiling is extremely useful, as TCP traffic represents a dominant and at the same time very predictable part of internet workload. According to recent internet measurement studies [8], TCP protocol is responsible for over 95% of total bytes

I. Tomkos et al. (Eds.): BROADNETS 2010, LNICST 66, pp. 328–340, 2012.

transmitted. Additionally, it has been established that internet workload is heavy-tailed and a large part of network capacity is dedicated to long-lived file transfers. Most of this traffic is mainly P2P file exchanges and web file transfers, as well as online video connections over HTTP (i.e. Youtube). Moreover, as shown in [7], the TCP throughput of large transfer TCP flows over OBS, collapses to a single fixed operating point (called equilibrium) enabling the accurate prediction of aggregate TCP throughput in a single OBS link. However, aggregate throughput statistics have a limited value, since they only reflect the network state when traffic measurements are taken. Network operators are mainly interested in flow-level statistics, exported for example by enterprise routers through Cisco Netflow protocol [6]. Flow-level statistics can provide insights for user-perceived performance, and they are useful for network management and capacity planning. Estimating flow-level statistics is a resource-intensive process, as it involves specialized monitoring equipment that captures packets from monitored flows and extracts statistics. Typically, a small percentage of the original network flows is monitored, denoted as "flow sample". There is a lot of research done on sampling TCP flows [14], and optimally estimating flow statistics from flow samples [15], a process that is called "inversion".

In this paper, we propose a TCP flow profiler for OBS networks, which performs on-line estimation of active TCP flow statistics, as well as network parameters like burst loss ratio and round trip time. The rest of the paper is organized as follows. Section II discusses predictability issues of long-lived TCP traffic, while Section III presents the in detail the TCP traffic profiler architecture. Finally, Section IV presents performance evaluation results, obtained through simulations using ns-2.

2 TCP Traffic Profiling in OBS Networks

In this section, the main issues and challenges are discussed, for collecting TCP traffic statistics in OBS networks. In principle, collecting detailed traffic information for active TCP connections provides an intimate traffic knowledge, which can serve as a basis for traffic engineering. Flow-level metrics like flow size distribution, number of active flows, and average throughput are useful for network management and capacity planning.

2.1 Collecting Traffic Statistics in OBS Networks

For gathering flow-level traffic statistics, a monitoring system is required, that stores and constantly updates a record of sampled flows. Cisco Netflow [6] is an example of such system, deployed in commercial enterprise-class routers. For gathering network-wide traffic statistics, a node with access to all packets transmitted is needed. In OBS networks, this node can be the burst assembler. Burst assembler is well suited for gathering traffic statistics, having access to all packets received at the edge node. Packet headers have to be examined, before being assigned to the appropriate burst assembly queue, assuming that a separate queue for each source-destination pair is

kept. Thus, keeping traffic statistics does not bear a significant overhead for edge nodes, provided that very short-lived TCP connections are filtered-out.

It has to be noted that in real-world networks, long-lived TCP flows constitute both the dominant (in terms of bytes transmitted) and the most predictable part of network traffic. Internet workloads are regarded to be heavy-tailed [13], i.e. their flow length distribution Z follows a probability distribution such that the right tail has power-law decay, or:

$$P[Z > z] \sim z^{-a}, \qquad z \to \infty . \tag{1}$$

Long-lived flows, also known as "elephant flows", are responsible for 80% of the overall bytes transmitted. This has been verified by recent internet measurement studies in [8], where it has been found that the majority of data transmissions over the internet is due to large file transfers, online video downloads over HTTP (i.e. YouTube) and P2P file exchanges. On the other hand, the average size of data transmitted from interactive Web request is fairly small, in the order of 26–30 kB, with a small contribution the aggregated throughput and a high degree of burstiness. To this end, it becomes clear that it is preferable to filter-out short-lived flows and gather statistics solely for long-lived TCP flows. This requires the profiler to effectively distinguish short-lived from long-lived flows, and can be carried out by defining a *flow lifetime threshold* θ, that is intuitive and easy to implement.

In what follows, we focus on the TCP traffic predictability issues, and argue that formula-based TCP throughput estimation for long-lived TCP traffic is a viable solution.

2.2 Predictability of Long-Lived TCP Flows in OBS Networks

The feedback control mechanism of TCP protocol is a well known mechanism for introducing a degree of predictability in TCP traffic. TCP flows are prevented from transmitting at the full link capacity, to avoid saturating the network, and they can only transmit as many packets as their congestion window (*cwnd*) allows. Newly arrived flows have to "probe" for network capacity, by starting their transmission with a *cwnd* of 1, and which is doubled in every lossless round. TCP flows ultimately reach steady state (or congestion avoidance state) after the first segment loss is detected, which corresponds to network equilibrium. In such a case, all flows fairly share the available bandwidth.

TCP congestion avoidance algorithm operates in a purely deterministic way, while TCP window in an idealized scenario follows a periodic "saw tooth" profile. This traffic pattern is common to all TCP implementations that only differ in the fast recovery phase, with a small contribution in the overall throughput.

It has been established in [9] that traffic dynamics of large transfer TCP flows can be accurately estimated by making use of appropriate performance modeling formulas. This is due to the fact that long-lived flows' congestion window ultimately converges to a steady state, reaching equilibrium. In [7] the authors show that they can accurately predict the network fixed point (i.e. the steady state input rates of

long-lived TCP flows) over an OBS link. Formula based throughput estimation requires up-to-date measurements of burst loss ratio p, flow round trip time (RTT) and segment per burst distribution. Studies on TCP predictability have concluded that accuracy is negatively impacted by unpredictable queuing delays in congested paths. This results in fluctuating RTTs and affects the accuracy of the TCP performance estimation formulas. However, due to the bufferless nature of OBS networks, large transfer TCP flows over OBS are expected to be very predictable. The accuracy of TCP bandwidth estimation depends on the accuracy of the estimated network parameters, like burst loss ratio, as well as TCP flow statistics like flow access rates and access delays.

3 TCP Flow Profiler Architecture

In this section, the profiler architecture is presented, as well as how flow statistics are formed from a subset of active flows, called the *flow sample.* Flow sampling is employed for limiting the consumption of resources required to extract flow statistics from traffic measurements, which would be prohibitive if all packets and flows were taken into account. Thus, it follows that the flow statistics from unsampled flows must be extracted from the sampled flow statistics. In this work, unbiased estimation of flow statistics is achieved by selecting the flow sample independently of measured statistics (unbiased selection). As an example, for performing an unbiased estimation of flow length distribution, sampled flows are selected independently of the flow length. Consequently, the statistics of the sampled flows are expected to converge to the ones of the unsampled flows, as the number of sampled flows approaches the overall number of active flows.

In what follows, a TCP flow profiler is designed, which is integrated with the burst assembly process. Its goal is to provide running online measurements of flow and network performance metrics. These metrics are constantly updated, so that they don't become outdated. Being in the middle of the flow end-to-end path, the flow profiler has access to all segments transmitted from TCP sources assigned to the burstifier, and it is able to inspect TCP segment headers without incurring excessive overheads. It does not have information on TCP segments lost before the burst assembly process, or on internal TCP sender state (like congestion window). The profiling module proposed here performs an estimation of the following metrics:

- Burst loss ratio
- Number of active long-lived TCP flows
- TCP performance variables (round trip time and access rate)
- Steady-state TCP throughput.

Passive estimation techniques are used in our profiler for compiling flow statistics, as opposed to active estimation techniques, where probing sessions are initiated by the profiler. Passive techniques allow monitoring of a large number of flows in a diverse set of network paths without interfering with traffic flows. Most of the metrics can’t be estimated in a stateless manner. In order to compile flow statistics, a table of flow

records, indexed by *flowID* must be maintained. A representative, unbiased flow sample has to be stored in this flow table, so that traffic statistics of the sampled flows match the traffic statistics of the unsampled flows. Usually, the traffic sample is a small percentage of the overall flow population (typically less that 1%) as the overall number of flows assigned to a burstifier is in the order of millions.

In the proposed scheme, TCP profiler divides active flows in sampled and un-sampled ones according to the flow sampling rate 1/N, which denotes that only 1 in N flows is sampled on average. For every packet that is received by the burstifier, the profiler determines if the packet will be retained and whether a flow record is active for it's flowID. If a flow record is active, the flow statistics are updated on the reception of the packet. If not, it instantiates a new record with the packet's flowID. The primary constraints of flow-level profilers are the memory bandwidth and memory size limitations, since millions of flows can be assigned to a single monitoring device. The memory bandwidth of DRAMs utilized in typical Netflow capable routers [6] is not sufficient to lookup the flowID of all incoming packets, thus a small sampling rate is used. To cope with this problem, in [16] an alternative profiling architecture was proposed, based on fast SRAMs. However, SRAMs are expensive and come in very small sizes as compared to DRAMs – thus being capable of storing a smaller flow sample.

In this work, we employ hash-based sampling technique, which avoids memory lookup for packets belonging to unsampled flows, resulting in large memory bandwidth savings. According to this technique, the profiler calculates a hash value for the flowID of every packet received, i.e. $h = f(flowID)$, which is normalized in the interval $h \in [0,1]$. Since hash algorithms are designed with an objective to evenly distribute a stream of (possibly correlated) values, the hash value is uniformly distributed and thus can serve as an unbiased criterion of flow selection. Thus, if $h \leq 1/N$, with 1/N the flow sampling rate, the flow is sampled or else it is not.

The abovementioned selection technique is unbiased and efficient, since one can track as many flows as the DRAM memory bandwidth allows, and accordingly set the value of the sampling rate. Additionally, since in OBS networks the burst assembler has to inspect packet headers in order to assign them to the appropriate queue, this technique only adds-up a few clock cycles overhead per received packet. Similar works that calculate hashes of received TCP packet *flowIDs* were able to track millions of flows at line speeds [10].

The efficiency of the abovementioned technique can be further enhanced by filtering out slow flows. One-packet long TCP flows have a disproportionally large frequency, owing to HTTP protocol (clients send http requests to web servers encapsulated in a single packet). To avoid wasting resources for storing one-packet flows, in the proposed scheme, new records for slow flows (i.e. flows with a single data packet in a burst) are not created.

3.1 Burst Loss Ratio Estimation

The estimation of the burst loss ratio is carried out using the signaling messages received by the edge router's control unit. For every dropped burst the core node

returns a message (BHP_DROP) to the edge router to report the loss. This is communicated to the profiler, which stores a bit vector of bursts successfully transmitted (denoted with '0') and bursts lost (denoted with '1'). Bit values X_k are assumed independent Bernoulli random variables that take value '1' with a probability equal to the burst loss ratio. The burst loss ratio is thus estimated as the sum of '1' values in the vector divided by the vector length *W*. Thus, the burst loss ratio is defined as:

$$p = \frac{\sum_1^W X_k}{W}. \tag{2}$$

For the online estimation of burst loss ratio, we use the well known sliding window averaging technique that discards aging values, older than the vector length. According to this technique, proposed in [11], for packet loss ratio estimation, given that X_k is the kth bit value of the vector corresponding to the k^{th} burst transmission and *W* is the vector length, a running estimation of burst loss ratio is obtained by:

$$\hat{X}_{i+1} = \frac{1}{W} \sum_{k=i-W+1}^{i} X_k \,. \tag{3}$$

The vector length *W* is calculated based on the desired accuracy, using the analytic model proposed in [11]. The estimated burst loss ratio error for a vector length *W*, a real burst loss ratio *p* and a confidence interval of 95% is derived as:

$$e = \frac{1.96\sqrt{p(1-p)}/W}{p}. \tag{4}$$

Thus, for a burst loss ratio of 1% and an error rate of 0.002, this corresponds to a bit vector length of 4.000 values, which is the one used in our experiments.

3.2 Access Rate and RTT Estimation

Flow access rate depends on the bottleneck link across the path from the sender to the edge router, for which the burstifier has no knowledge. It can be estimated by finding the maximum number of segments that a flow injects in a burst. Thus, it requires estimating segment-per-burst ratio for the sampled flows, throughout their duration while retaining its maximum value. Then, the flow access rate for flow-*i,* denoted as r_i, can be estimated as:

$$r_i = \frac{MAX\{SPB_i\} * MSS}{T_{MAX}}. \tag{5}$$

Where *SPB* is the segment-per-burst ratio, *MSS* is the maximum segment size and T_{MAX} is the burst assembly time. With respect to the round-trip time of the sampled flows, it can be easily estimated at the flow setup phase during the three-way handshake period, a procedure detailed in [12]. The profiler must keep track of the time each control packet was received (SYN, SYN-ACK and ACK), which allows the direct computation of the Profiler-to-Server and Client-to-Profiler round trip times. The flow RTT is estimated as the sum of these two values.

3.3 Long Lived Flows Threshold

For distinguishing flows into long and short lived, we define a threshold in the flow length, where length refers to the number of packets transmitted until the flow concludes its transmission. If the observed flow length is smaller than the threshold, denoted with θ, then it is classified as short lived – otherwise it is classified as long-lived. It has been mentioned previously that steady-state throughput of large file transfers constitutes a significant part of internet traffic, while long-lived flow dynamics are very predictable. Thus, we may set threshold θ as the expected flow length of the first packet loss. After the first loss, the flow will enter the congestion avoidance phase. This approach is much more efficient than passively estimating TCP state of individual active flows, a process which is TCP implementation dependent and very computationally intensive. In contrast, the proposed technique is more straightforward, easy to implement, and it is implementation-agnostic, as it assumes an idealized congestion window evolution. It assumes that TCP flows begin their transmission with a congestion window of one segment that doubles per lossless round, while after the first segment loss the flows enter the congestion avoidance state, in which they stay until they conclude their transmission. Fast retransmit/fast recovery and Time-Out phases are not taken into account, as they have a minimal effect on TCP performance for small burst loss ratios.

The expected number of packets transmitted until the first loss, which serves as the long-lived flow threshold, is estimated analytically. In an OBS network packet losses are correlated, due to the assembly of multiple packets in a single burst. Thus, packet loss ratio is not identical to the burst loss ratio. One burst loss carrying at least one packet of a flow in the slow-start phase is enough to agitate the flow to its congestion avoidance (steady-state) phase, regardless of the number of assembled packets carried by the burst. Assuming that burst losses in OBS networks are independent events, the probability of transmitting k bursts before a loss occurs and assuming a burst loss ratio p is:

$$P[B = k] = (1 - p)^k p\,. \tag{6}$$

The number of segments that are assembled in a single burst denoted as *SPB* (segments-per-burst ratio) depends on the flow access rate, the burst assembly period and also by the flow congestion window. Thus, the number of segments transmitted before the first loss occurs can be approximated as:

$$E[P] = S * E[B] = S * \sum_{0}^{\infty} (1 - p)^k p = \frac{SPB}{p}. \tag{7}$$

The value SPB/p is equal to the TCP Triple Duplicate Period (TDP) over OBS, derived from [3]. Alternatively, for flows that had no loss before saturating their local capacity, the bandwidth-delay product r_i*RTT_i can be used as a threshold, where both values are estimated by the profiler for flow-i. For a typical burst loss ratio of 1% and a value of *SPB=4*, the long-lived flow threshold corresponds to 400 segments. This value is considerably higher than the average HTTP request size, and thus can successfully serve as a definite criterion of differentiation.

Next, we argue that in heavy-tailed workloads like the ones in the internet, the majority of bytes are transmitted after this threshold has been reached, i.e. in the steady-state phase. In [13], web traffic is classified in web requests, shown to follow a heavy-tailed distribution with parameter α = 1.21 and web file transfers shown to follow a heavy-tailed distribution with heavier tail weights, and α = 1.1. Due to the heavy-tailedness of the workload, long-lived flows are very likely to continue being active a long time after they have been identified as such, i.e. long after they have exceeded the long-lived flow threshold. Heavy-tailed flow length distribution Z has the following fundamental property:

$$\lim_{z\to\infty} \Pr[Z > z + k \mid Z > z] = 1\,. \tag{8}$$

This denotes that for flows with a long duration (z→∞) the probability of transmitting k extra bytes before concluding their transmission is large, i.e. $\Pr[Z > z + k] \to 1$ as $z \to \infty$. In other words, the longer the flow duration is, the higher it's expected residual life is. As an example, assuming that file sizes follow Pareto distribution, with an index parameter $\alpha = 1.1$, then assuming that the flow reached the threshold θ, its *Mean Residual Life* (i.e. the expected number of bytes the flow is expected to transmit before concluding its transmission) is [17]:

$$MRL(\theta) = \frac{1}{1-\alpha}\theta = 10 * \theta\,. \tag{9}$$

Thus, flows characterized as long-lived, transmit more than 90% of their file size after having reached threshold θ, in the steady-state phase.

3.4 Long-Lived Flow Counting

In the proposed architecture, the number of active long-lived flows that have been assigned to the burstifier, along with their TCP state has to be known. A sampled flow is regarded active as long as it's *flowID* is stored in the flow table. The *flowID* is inserted after the first packet from the flow is received at the burstifier, and it is removed after a time threshold of inactivity (few RTTs). Thus, the number of active sampled flows is the number of distinct *flowIDs* in the flow table. In addition, the number of active sampled long-lived flows, is the number of distinct *flowIDs* exceeding threshold θ, as defined in previous section. This is a valid assumption, since the proposed TCP profiler architecture guarantees unbiased selection of the flow sample. Sampled flows have a fixed probability of being selected, irrespective of their duration or their ON period. This can be formulated as (following the same rationale as in [14]): Assuming that the number of active long-lived flows at a given time on the network is NF_{LL}, each one of them (due to unbiased selection) is modeled with an independent Bernoulli random variable w_i with a selection probability of *1/N*. The expected number of selected flows (assuming a selected flow contributes '1' to the sum) is:

$$\widehat{NF}_{LL} = N * \sum_{i=1}^{Nf} wi\,. \tag{10}$$

where NF_{LL} is the number of steady-state flows and N is the inverse the sample rate. Thus, the true number of active long-lived flows converges to the number of sampled long-lived flows multiplied by N, i.e. the above equation performs an unbiased estimation of the number of long-lived flows. Regarding the variance of the estimation, it is bounded by the number of active long-lived flows on the system (NF_{LL}) as well as the profiler sampling ratio *1/N*. Specifically, the standard error of the estimation is:

$$\frac{\sqrt{Var(\widehat{NF}_{LL})}}{NF_{LL}} = \sqrt{\frac{N}{NF_{LL}}}. \tag{11}$$

It can be seen that the estimator variance is small and can be made arbitrarily small when a high sample ratio is used.

3.5 Estimating Aggregated Steady-State Throughput

The goal of the steady-state throughput estimation process is to calculate the aggregated steady-state throughput of file transfers, based on TCP traffic statistics at the flow level. TCP throughput calculations are based on network parameters and flow statistics as estimated by the profiler, such as round-trip delay and segments-per-burst distribution. By constantly updating these, the steady-state throughput is estimated, taking into account potential changes to the number of active TCP flows, network state etc. This approach is advantageous for achieving fast conversion times caused by sudden state changes. TCP flow profiling and estimation of the network parameters allows pro-active network management, capacity planning and ultimately enhancing network performance and improving bandwidth utilization.

The performance of a single TCP flow over OBS has been analyzed in [3] giving closed formulas for estimating steady-state throughput for a given burst loss ratio, RTT and number of segments-per-burst. Here, we derive steady-state TCP throughput, assuming that the burst loss ratio, the segments-per-burst distribution and RTT statistics are evolving over time, and are constantly updated by the profiler.

The steady state throughput of a single TCP flow in an OBS network with a known number of segments per burst (*SPB*) and round trip time (RTT) and a known burst loss ratio *p* is obtained by the formula:

$$B(SPB, RTT) = \sqrt{1.5 * SPB} \Big/ RTT * \sqrt{p}. \tag{12}$$

Assuming different RTTs and access rates per TCP flow, with *SPB(i)* being the empirical distribution of segments per burst, and *RTT(i)* being the distribution of round-trip times, we calculate the average TCP steady-state throughput over OBS as:

$$\bar{B} = E\left[\frac{1}{RTT}\right] * E\left[\frac{\sqrt{1.5 * SPB}}{\sqrt{p}}\right]. \tag{13}$$

Or equivalently:

$$\bar{B} = \left(\overline{\frac{1}{RTT}}\right)\sum_{i=1}^{\infty} P\{SPB = SPBi\} * \frac{\sqrt{1.5 * SPBi}}{\sqrt{p}} * MSS\,. \tag{14}$$

Both $\left(\overline{1/RTT}\right)$ value and segment-per-burst distribution $SPB(i)$ are estimated by the traffic profiler, and so is burst loss ratio p. Coupled with the measured number of steady-state flows at time t, $N_F(t)$, the above formula can provide a constantly updated estimate of aggregated TCP steady-state throughput over OBS, denoted by R(t):

$$\mathrm{R(t)} = N_F\,(t) * \bar{B}\,. \tag{15}$$

4 Profiler Evaluation

The TCP profiler was evaluated with simulations using ns-2 platform. A simple 3-node topology was used, consisting of two edge routers, denoted as E1 and E2 interconnected via a single core router, denoted as C. Clients are assigned to edge node E1 and initiate file transfers on servers assigned to E2. The modeled OBS network uses JET protocol for resource reservations. Burst assembly process is performed at the edge nodes, using a timer-based aggregation algorithm (T_{MAX}) with a time threshold of 3ms. The network round trip time was set equal to 15ms, while all clients had a uniformly distributed access delay in the interval [0,2] ms and a uniformly selected access rate of 20Mb, 50Mb and 100Mb. A realistic traffic scenario was modeled, that consists of TCP connection requests, whose arrivals follow Poisson distribution and their sizes are drawn from Pareto distribution. The traffic profile is representative of typical internet workloads, with many short-lived TCP connections representing web requests, and fewer long-lived file transfers, responsible for the 90% of the overall bytes transmitted. In what follows, numerical results are based only on long-lived flows, since short flows spending their lifetimes in the slow-start phase were filtered-out by the profiler.

The exact parameters of the each file transfer were estimated by the profiler, which was evaluated with three different sampling rates, $\{1/2, 1/10, 1/20\}$. In what follows, we present experimental results comparing real and estimated TCP flow parameters for all sampling rates.

Figure 1 displays the evolution of the aggregated steady-state TCP throughput. It can be seen that the estimator's output closely follows the measured real value of the steady-state throughput, especially for high sampling rates, relying solely on the running estimation of burst loss ratio, number of active flows and flow-level statistics. Additionally, as expected, higher sampling rates yield more accurate throughput estimations.

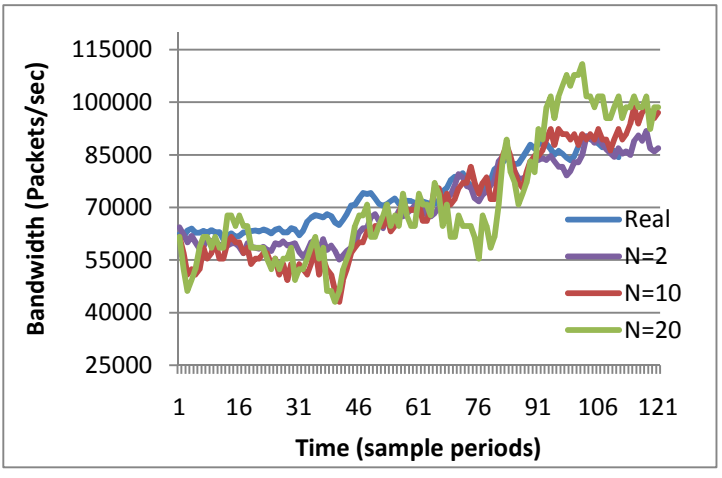

Fig. 1. Estimated versus true steady-state throughput, for different sample rates

Figure 2 displays the estimated number of active flows. Again, the estimated number of flows closely follows the real number of active flows while estimation accuracy improves with the increase of sampling rate. It must be noted however that the number of flows lacks the burstiness of TCP throughput and thus it is less sensitive to the selection of the flow sampling rate.

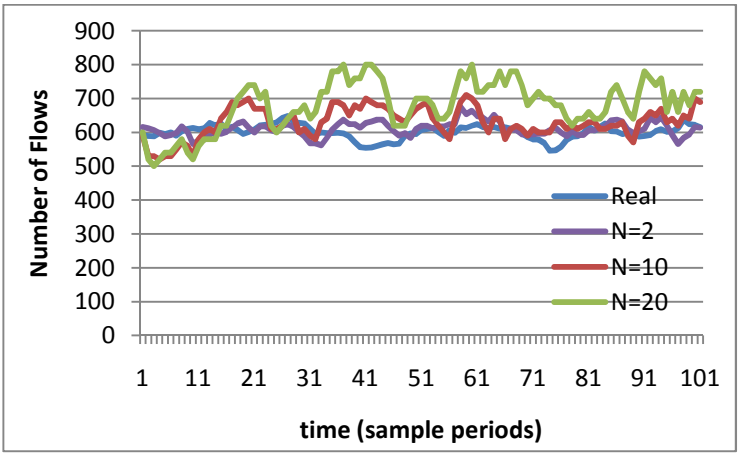

Fig. 2. Number of active flows estimation

Finally, for the characterization of the estimator's accuracy we have calculated the Cumulative Density Function (CDF) of the Relative Error of the estimation (see Figure 3) as well as the Coefficient of Variation (CoV) metric and the Root Mean Square Error (see Table 1), as an indication of the variance of the estimator.

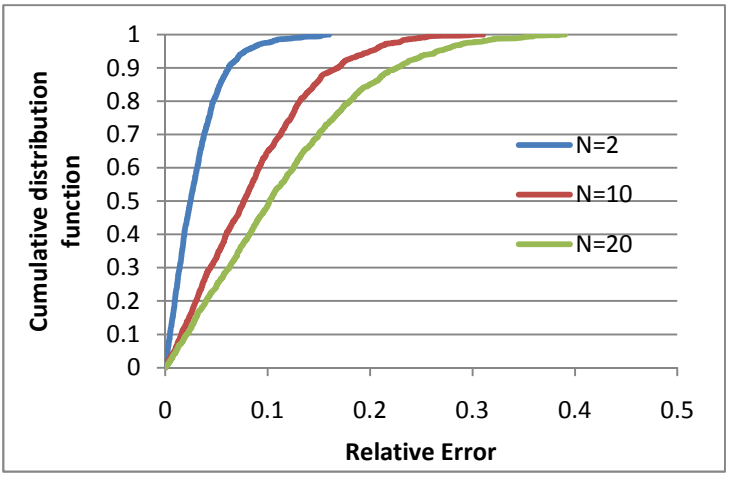

Fig. 3. Relative Error cumulative density function

Table 1. Estimator standard error

	p=0.5	p=0.1	p=0.05
Coefficient of Variation (CoV)	0,037	0,10	0,17
Root Mean Square Error (RMSE)	2945	8147	11326

5 Conclusions

In this paper, a TCP profiler has been designed for OBS networks, which is capable of estimating aggregated TCP throughput. The profiler operation relies solely on the running estimations of burst loss ratio, number of active flows and flow-level statistics like segments-per-burst distribution and RTT. These are used to estimate steady-state performance of TCP traffic over OBS networks. Simulation results have shown that the proposed scheme adequately profiles flow dynamics with a low accuracy variation and a low mean absolute error value.

Acknowledgements. The work described in this paper was carried out with the support of the BONE-project ("Building the Future Optical Network in Europe"), a Network of Excellence funded by the European Commission through the 7th ICT-Framework.

References

1. Qiao, C., Yoo, M.: Optical burst switching (OBS)-A new paradigm for an optical internet. J. High Speed Networks 8(1), 69–84 (1999)
2. Li, J., Qiao, C., Xu, J., Xu, D.: Maximizing throughput for optical burst switching networks. IEEE/ACM Transactions on Networking, TON (2007)

3. Yu, X., Qiao, C., Liu, Y., Towsley, D.: Performance evaluation of TCP implementations in OBS networks. Technical Report 2003-13, CSE Dept., SUNY, Buffalo (2003)
4. Yu, X., Li, J., Cao, X., Chen, Y., Qiao, C.: Traffic statistics and performance evaluation in optical burst switched networks. IEEE/OSA Journal of Lightwave Technology 22(12), 2722–2738 (2004)
5. González, O., Guidotti, A.M., Raffaelli, C., Ramantas, K., Vlachos, K.: On transmission control protocol synchronization in optical burst switching. Photonic Network Communication 18(3), 323–333 (2009)
6. Cisco Systems, NetFlow services and applications, White Paper (2000)
7. Cameron, C., Le Vu, H., Choi, J., Bilgrami, S., Zukerman, M., Kang, M.: TCP over OBS - fixed-point load and loss. Optics Express 13(23), 9167–9174 (2005)
8. Maier, G., Feldmann, A., Paxson, V., Allman, M.: On dominant characteristics of residential broadband internet traffic. In: Proc. ACM IMC (2009)
9. He, Q., Dovrolis, C., Ammar, M.: On the predictability of large transfer TCP throughput. In: ACM SIGCOMM (2005)
10. Schuehler, D.V., Lockwood, J.W.: TCP splitter: A TCP/IP flow monitor in reconfigurable hardware. IEEE Micro (2003)
11. Yajnik, M., Moon, S.B., Kurose, J., Towsley, D.: Measurement and modeling of the temporal dependence in packet loss. In: IEEE INFOCOM (1999)
12. Jiang, H., Dovrolis, C.: Passive estimation of TCP round-trip times. In: ACM SIGCOMM (2002)
13. Park, K., Kim, G., Crovella, M.: On the relationship between file sizes, transport protocols, and self-similar network traffic. In: Proc. IEEE ICNP (1996)
14. Duffield, N., Lund, C., Thorup, M.: Properties and prediction of flow statistics from sampled packet streams. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment (2002)
15. Tune, P., Veitch, D.: Towards optimal sampling for flow size estimation. In: Proceedings of the 8th ACM SIGCOMM (2008)
16. Estan, C., Varghese, G.: New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. ACM Transactions on Computer Systems (2003)
17. Luo, S., Li, J.H., Park, K., Levy, R.: Exploiting Heavy-Tailed Statistics for Predictable QoS Routing in Ad Hoc Wireless Networks. In: IEEE INFOCOM (2008)