A TCP-Specific Traffic Profiling and Prediction Scheme for Performance Optimization in OBS Networks

Kostas Ramantas and Kyriakos Vlachos

***Abstract*—The efficient transmission of TCP traffic over optical burst-switched (OBS) networks is a challenging problem, due to the high sensitivity of the TCP congestion control mechanism to losses. In this paper, a TCP-specific traffic profiling and traffic prediction scheme is proposed, for optimizing TCP transmission over one-way OBS networks. In the proposed scheme, the burst assembly unit inspects TCP packet headers in parallel to the assembly process, keeping flow-level traffic statistics. These are then exploited to derive accurate traffic predictions, in at least one flow round trip time-long prediction window. This allows notifying traffic schedulers of upcoming traffic changes in advance, in order to optimally reschedule their resource reservations. In this paper, we detail the traffic profiling and prediction mechanism and also provide analytical and simulation results to assess its performance. The performance gains when using the prediction scheme are shown with a modified one-way OBS reservation protocol, which efficiently and in advance reserves resources at the burst level.**

***Index Terms*—Burst assembly; Optical burst switching; TCP profiling; Traffic estimation.**

I. Introduction

TCP transmission over optical burst-switched (OBS) networks [1] has been extensively studied in the related literature. Various burst assembly and burst scheduling algorithms have been proposed, to enhance the efficient transmission of TCP-over-OBS networks. However, it still remains an open problem, since the relatively high burst loss ratio experienced in OBS networks is incompatible with a TCP congestion control mechanism. It has been observed that burst losses have a significant impact on the TCP end-to-end performance. 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 subsequently entering the slow-start phase. It may also result in synchronizing TCP transmissions with an imminent effect on link utilization [2]. The introduction of an unpredictable delay challenges the window mechanism used by the TCP protocol for congestion control.

Manuscript received March 28, 2011; revised October 4, 2011; accepted October 26, 2011; published November 7, 2011 (Doc. ID 144773).

The authors are with the Computer Engineering and Informatics Department & Research Academic Computer Technology Institute University of Patras, Rio, Greece (e-mail: kvlachos@ceid.upatras.gr).

Digital Object Identifier 10.1364/JOCN.3.000001

Enhancing TCP throughput over OBS is critical, since TCP traffic represents a dominant part of Internet traffic. Recent Internet measurement studies [3] show that TCP protocol is responsible for over 95% of the total Internet traffic. This has led many researchers to cope with the performance evaluation [4] and enhancement [5–7] of TCP performance over OBS, which nevertheless remains an open problem. One very interesting approach for the enhancement of TCP performance over OBS is traffic prediction. If it were possible to accurately predict the throughput of TCP flows, it would also be possible to predict burst sizes. That would allow making reservations of the appropriate resources in advance, enhancing network performance and improving bandwidth utilization. This approach has proved promising in previous works, which exploit burst size predictions to perform forward resource reservations, achieving performance enhancements and reduced edge delay [8].

Predicting aggregate traffic at an edge node is treated as a time-series modeling problem in the literature, in which one attempts to predict the future value of the aggregated throughput, based on a traffic history record. This approach requires the use of traffic models that capture the underlying traffic structure, and then perform time-series analysis. The most commonly used self-similar processes for representing long range dependent (LRD) traffic are fractional Gaussian noise (fGn) and fractional auto-regressive integrated moving average (fARIMA) processes [9]. However, such methods have a high computational complexity, and are considered inappropriate for short-term predictions. Another approach that makes no assumptions about underlying traffic structure concerns the use of minimum mean square error predictors like the least mean square (LMS) filter [10]. The predictions are output of a Linear Prediction Filter, whose weights are updated online, in a way that minimizes the mean square error between predicted and true values. An added benefit of this approach is the relatively low computational complexity.

However, using LMS filters for traffic forecasting also bears a significant limitation, which is the lack of traffic information at the flow level. Traffic measurements of aggregated throughput only reflect network conditions at the time that the measurements were taken. Thus, changes in the network state (such as the burst loss ratio) are enough to invalidate them, taking time for the filter to converge.

In this work, predictions are based on flow-level modeling of network traffic, combined with a TCP-specific traffic profiler. The traffic profiler periodically outputs flow-level traffic

statistics, compiled in real time and exploited in the burst prediction process. This approach, although more resource intensive than simply keeping track of the aggregated throughput, is advantageous for achieving fast convergence times in the case of sudden state changes, as it takes advantage of the inherent predictability of TCP flows.

The contribution of this work is twofold. First, we propose a TCP traffic profiler and traffic predictor for OBS networks that is integrated with the burst assembly process. The proposed prediction scheme performs traffic and burst size predictions based on flow-level traffic statistics computed in real time by the profiler. A detailed evaluation of the proposed prediction scheme proves accurate for short-term forecasting for intervals of a few flow round trip times (RTTs). Second, we disclose the performance gains of the predictive mechanism using a novel one-way resource reservation protocol that performs predictive batch reservation of resources at the burst level.

The rest of the work is organized as follows. In Section II, some background information and related work is discussed. Section III presents the TCP profiling architecture for OBS networks, while Section IV presents a flow-level model for representing TCP traffic in backbone networks, as a multiplex of a finite number of TCP connections. Section V presents a novel burst prediction mechanism, based on traffic statistics gathered by the flow profiler, that is then used (Section VI) in a prediction-based reservation scheme for one-way TCP-over-OBS networks. Finally, Section VII presents detailed performance evaluation results, investigating the accuracy of the proposed prediction scheme and the performance gains obtained with the proposed prediction-based reservation protocol.

II. Background and Related Work

In the TCP-over-OBS literature, two classes of solutions have been proposed for the enhancement of TCP performance over OBS networks. One class of solutions tries to mask the effect of bursts losses to the TCP protocol using deflection routing [11], burst retransmissions [7], or a coordinated multi-layer approach [12]. A second class of solutions focuses on improving TCP protocol to detect false congestion events and not trigger the congestion avoidance mechanism for packets lost due to contention. These techniques are based either on feedback received from burst losses, communicated to the TCP sender as in BTCP [5], or analyzing RTT statistics in a short time frame, as in SAIMD [6], to infer congestion.

Predictive resource reservation is another solution for performance enhancement in OBS networks that lies in the OBS domain (i.e., it does not require changes to the underlying transport protocols). It has attracted considerable research attention [8,10], as it can achieve a low burst loss ratio, comparable to that of two-way OBS, but with a reduced edge delay. In these works, bandwidth allocation in the network core is based on burst size predictions, output by an N-order LMS filter. This allows the reservation process to begin before the actual burst size is known, resulting in a latency reduction. Another advantage is that LMS filters make no assumptions about underlying traffic structure concerns the use of a Linear Prediction Filter [10], whose weights are updated online, in a way that minimizes the mean square error between predicted and true values. Q4

In this work, we extend this approach, relying on flow-level TCP traffic profiling to predict future burst sizes. Network traffic is modeled as the superposition of TCP connections, allowing us to take advantage of TCP predictability at the flow level, which is captured by well-known performance modeling formulas [4]. Burst size predictions are obtained from online traffic measurements, compiled in real time from a TCP flow profiler. This is advantageous for achieving fast convergence times in the case of sudden changes in the traffic profile (i.e., due to the sharp increases in the flow arrival rate), which is a scenario that LMS predictors suffer [8]. Further, we also propose a new predictive resource reservation protocol that exploits the above-mentioned burst-level predictions. Overall, the contributions of this work are as follows.

- Design of a TCP-over-OBS flow profiling scheme for extracting detailed flow-level traffic statistics.
- Design of a TCP throughput predictor, that periodically outputs an estimation of the aggregated TCP throughput and the future burst sizes, based on traffic statistics.
- Contribution of a new predictive resource reservation protocol that communicates to the core nodes the upcoming traffic changes and in advance performs burst-level reservations.

The proposed scheme is compatible only with TCP traffic, since it relies on profiling the window dynamics of TCP protocol. UDP flows can be present on the network and they can be assembled together with TCP flows, but these will not be taken into account in the prediction process but rather considered as background traffic. In scenarios where UDP flows have a significant contribution to overall bytes transmitted (i.e., >5%) other predictions schemes, such as the ones referenced in [8,10], can be applied specifically to UDP traffic.

A. TCP Traffic Predictability

Formula-based estimation of TCP throughput has been shown to be a viable solution for predicting network state in OBS networks. The authors of [13] were able to effectively derive the TCP equilibrium point in an OBS link, assuming a fixed number of active long-lived TCP connections. The feedback control mechanism of TCP protocol is known to introduce a degree of predictability in TCP traffic, as it operates in a purely deterministic way. The TCP congestion window (usually written as cwnd) follows a periodic *saw-tooth* profile in the steady state, a pattern which is common to all TCP implementations. Differences in the implementation of the fast-retransmit/fast-recovery phase have a small contribution to the overall throughput for practical segment loss ratios. Newly arrived TCP flows enter the slow-start phase, where they have to probe for network capacity, by starting their transmission with a cwnd of 1, which is then doubled in every lossless round. Conversely, on a TCP round with at least one loss the cwnd is divided by 2, and it enters the steady-state phase (where cwnd increases by one segment per

round). TCP predictability is known to be negatively impacted by unpredictable queueing delays in congested paths. This results in fluctuating RTTs that affect the accuracy of the TCP performance estimation formulas. However, due to the bufferless nature of OBS networks, large TCP flow transfers over OBS are very predictable.

TCP steady-state and slow-start performance in OBS networks can be accurately predicted by making use of appropriate performance modeling formulas. Long-lived TCP flows reach a steady state (or congestion avoidance state) which corresponds to an equilibrium point, where the network capacity is equally shared among all active flows. Thus, their performance is dominated by steady-state TCP performance over OBS [4]. The performance model for steady-state flows requires up-to-date measurements of the burst loss ratio p, as well as the number of *segments per burst* (SPB) transmitted. Q5 On the other hand, the performance of short-lived flows is dominated by RTT and TCP slow-start performance, as they typically conclude their transmission before experiencing a loss. In this work, we take into account both short-lived and long-lived flows, employing appropriate modeling to predict their performance in the slow-start and the steady-state phases.

B. Traffic Profiling

Traffic profiling is vital in commercial enterprise-class networks, because it serves as a basis for network management, capacity planning, and for gaining insights on user-perceived performance. Monitoring devices are typically deployed on the network edge, to capture packets from monitored flows and extract flow-level traffic statistics. Cisco Netflow [14] is an example of such a system, deployed in commercial networks. It operates in a passive mode (i.e., it does not inject or alter traffic but it measures existing traffic passing through), and it is able to extract detailed traffic statistics in the network points, where traffic measurements are carried out. These statistics concern flow performance variables such as throughput and loss ratio. It must be noted that estimating flow-level statistics is a resource-intensive process, which uses up significant resources of forwarding devices. Thus, flow monitoring architectures typically resort to a flow sampling technique to reduce the measurement and profiling overhead. According to this technique, per-flow counters are updated solely for a small subset of flows, which typically does not exceed 1/1000 of the active ones. In this work, a new TCP flow profiling architecture for OBS networks is proposed, which uses the well-known hash-based sampling technique for extracting detailed network-wide traffic statistics. Flow sampling is integrated with the burst assembly process, in the edge node.

III. TCP Traffic Profiling

In this section, a TCP profiler architecture for OBS networks is presented. Its goal is to provide running online measurements of flow parameters and performance counters, and extract flow-level traffic statistics. Estimating TCP workload characteristics constitutes the basis for our traffic prediction and bandwidth provisioning scheme. It is performed at the edge nodes of an OBS network, and it is integrated with the burst assembly process [15].

A. Flow Sampling Architecture

Traffic measurements are typically performed on sampled packet substreams, a process called *traffic sampling*, in order to limit the consumption of resources. These measurements are subsequently used to compile flow-level statistics of active TCP flows. One of the key issues in sample-based estimation is the unbiased selection of sampled packets, which must be independent of the measured quantities, in order to avoid any *selection bias*. For example, probabilistic sampling used in the Cisco Netflow system for estimating the flow length distribution is known to be prone to selection bias, and to underestimate the frequency of short flows [16].

The TCP flow profiler proposed in this work employs the hash-based sampling technique [17], which is unbiased and has a straightforward and efficient implementation. This technique performs a random (Bernoulli) selection of flows and retains all packets received from sampled flows. Selected (or *sampled*) flows are called the *flow sample*, and they account for a small percentage (typically 1/1000) of overall active flows. The flow sample is stored on a hash table, which maintains a set of per-flow counters, updated on a packet-by-packet basis. This hash table is indexed by the *flow*-tuple.

For every packet that is received by the burstifier, the flow profiler determines whether a flow record is active for that flowID. If a flow record is active, the flow statistics are updated on reception of the packet. If not, the profiler must decide whether the packet will be retained. Since hash algorithms are designed with an objective to evenly distribute a stream of (possibly correlated) values, the flowID is assumed uniformly distributed and thus can serve as an unbiased criterion of flow selection. Specifically, the selection probability is $p_s = 1/N$, where $1/N$ is the sampling rate, and u is a normalization of the flowID, $u \in [0,1]$. Then, if $u \le p_s$, the flow is sampled or else it is not. If the flow is sampled, then the profiler instantiates a new record with the packet's flowID.

Hash-based sampling is carried out in parallel with the burst assembly process, where the packets' headers are, in any case, inspected for being assigned to the appropriate assembly queue. Thus, the overhead of the scheme is limited to a few clock cycles for the computation of the flow hash for the unsampled packets and an additional read/write memory cycle for updating the performance counters of sampled packets. To this end, the prime constraints are memory bandwidth and memory size, which are proportional to the sampling rate. In other words, the higher the sampling rate is, the more the active flows that are being stored in the hash table and the more the read/write cycles for updating their statistics.

B. Flow Performance Counters

The flow profiler stores an array of counters per sampled flow, to measure a set of flow characteristics. After receiving a

packet that belongs to the flow sample, it updates one or more of the corresponding counters. These per-flow counters are as follows.

1) *Flow length*: This measures the number of TCP data packets transmitted from the flow. It is incremented on a packet-by-packet basis.
2) *Flow RTT*: This measures the round trip time from the flow source to the destination. It is calculated once during the three-way handshake period, a procedure detailed in [18]. The profiler must keep track of the time each control packet (SYN, SYN-ACK, and ACK) was received. Then, the flow RTT is estimated as the sum of Profiler-to-Server and Client-to-Profiler round trip times.
3) *SPB ratio*: This counter measures the number of segments of the flow assembled in a burst. Flow SPB is determined by the flow sending rate and the duration of burst assembly timer (T_{bat}), i.e., for flow i with a sending rate r_i, $E[\text{SPB}_i] = E[r_i] \times T_{\text{bat}}$.

The storage requirements for each per-flow record are less than 100 bytes, since we only have to store the aforementioned flow counters as well as the flowID. It is clear that, for very high sampling rates, the memory bandwidth required to update the aforementioned counters can become significant. For example for a 40G link, updating counters on a packet-by-packet basis (i.e., within a few ns) requires a fast SRAM for storing the flow sample, trading high memory bandwidth for a significantly smaller capacity. Alternatively a hybrid counter architecture can be used, combining high bandwidth of SRAMs for storing fast-changing "shallow" counters, with high capacities of DRAMs for storing slowly changing "deep" counters [19].

C. Extracting Sampled Flow Statistics

In parallel to updating flow counters, the flow profiler also compiles flow-level traffic statistics. Specifically, it estimates the empirical density functions of the flow SPB and RTT counters. Given $(x_1, x_2, \ldots, x_n)$ an array of n RTT or SPB samples, the empirical density function can be output from a density estimator, such as a simple histogram estimator, or a kernel density estimator. The latter, for kernel K and bandwidth h, calculates the unknown empirical density function $\hat{f}$ as

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{1}$$

Thus, along with the flow counters, the profiler also outputs RTT(.) and SPB(.) densities, updated periodically during a window of measurements. Time is divided into measurement epochs (e.g., 5 min periods) during which RTT and SPB samples are gathered by the profiler. At the end of each measurement epoch, the density estimates are updated using a non-parametric density estimation method. It must be noted that the flow-based sampling technique employed by the profiler is known to guarantee an independent selection of the flow sample. Thus, sample-based traffic statistics can serve as unbiased estimates of unsampled ones [16].

D. Estimating Burst Loss Ratio

The estimation of the burst loss ratio is carried out by the profiler based on the signaling messages received at the edge router's control unit. For every dropped burst, the core node returns a message 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 to be 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 p is derived as

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

For the online estimation of the burst loss ratio, we use the sliding window averaging technique that discards aging values, older than the vector length. According to this technique, given that X_k is the k^{th} 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 [20]. The estimated burst loss ratio error for a vector length W, a real burst loss ratio p, accuracy a, and confidence interval 95% is derived as

$$W = \left(\frac{1}{p} - 1\right)\left(\frac{196}{a}\right)^2. \tag{4}$$

For example, for a burst loss ratio of 1% and an accuracy of ±10%, W corresponds to a bit vector length of 38032.

IV. TCP Traffic Modeling

Many authors have analyzed the Internet traffic and have shown that it behaves in agreement with LRD and self-similar processes [9]. Typically, time-series models are used for modeling Internet traffic, in the form of an aggregate rate function. Time-series models are hard to calibrate (i.e., estimate their parameters for a given network workload), due to the high degree of multiplexing of numerous flows, whose behavior is strongly influenced by the transport protocol and the network state. Additionally, time-series models lack information at the flow level, and measurements of aggregated throughput only reflect network conditions at the time that the traffic measurements were taken.

A. Traffic Model

In this work, we model network traffic as the multiplex of a finite number of TCP flows, an approach proposed in [21].

TCP flow dynamics are a function of the flow congestion window, whose evolution is a function of a small set of parameters, i.e., the round trip time, segment-per-burst ratio, and the burst loss ratio. The TCP flow profiler detailed in the previous section keeps traffic statistics for TCP performance parameters as well as the number of active TCP flows. Thus, the aggregated throughput of N active TCP flows can be derived as $R(t) = \sum_N X^m(t)$, with $X^m(t)$ being the flow rate process that measures the transmission rate of flow m.

In what follows, time is divided into control time intervals of duration τ, termed *prediction intervals*. We define $S_k^{(m)}$ as the discrete time equivalent of the flow-m rate process. It returns the number of segments transmitted from flow m during the k^{th} prediction interval, from time $t = k \cdot \tau$ to $t = (k+1)\tau$, based on its rate process $X^{(m)}$:

$$S_k^{(m)} = \int_{\kappa\times\tau}^{(\kappa+1)\times\tau} X^{(m)}(t)dt. \tag{5}$$

The flow rate process depends on the TCP protocol dynamics and the flow connection size, to be discussed in the following sections.

B. Flow Connection Size Model

According to the flow-level model employed in this work, network traffic is represented as the superposition of TCP connections that arrive, transmit a file at the rate allowed by their congestion window, and then leave the system. Connection sizes are drawn from a heavy-tailed Pareto distribution. This is a commonly accepted model, as it explains the self-similarity of Internet traffic [9]. In what follows, the connection size distribution is modeled with a Pareto random variable X, whose complementary CDF, defined as $P(x) = Pr(X > x)$, is of the form

$$P(x) = \left(\frac{x}{x_{\min}}\right)^{-\alpha}. \tag{6}$$

It can be seen that $P(x)$ is parameterized with the $x_{\min}$ parameter, as well as the tail index parameter α of the heavy-tailed distribution. For estimating the index parameter α, the profiler uses maximum likelihood estimation (MLE). The estimation is based on a sample of connection size observations, maintained by the profiler for the duration of a measurement epoch (typically a few minutes long). For every sampled flow that terminates, the flow profiler keeps a record of its length at the termination time. The MLE that outputs an estimate of the tail index α based on these observations is [22]

$$\widehat{\alpha} = n\left[\sum_{i=1}^{n} \ln\left(\frac{x_i}{x_{\min}}\right)\right]^{-1}, \tag{7}$$

where x_i, $i \in [1\ldots n]$, are the observed values of the flow connection sizes. The minimum connection size $x_{\min}$ is assumed known or easily extracted directly from just observing the data. This procedure is repeated periodically to take into account a possible shift on the flow size distribution over the course of time.

It must be noted that the flow connection size is not directly observable when random packet sampling is employed in the traffic profiler, as in [23]. In that case, a thinned version of the connection size distribution is output from the profiler, making the process of estimating the index parameter more complex and subject to a high variance. However in the flow-based sampling scheme employed in this work, all packets from sampled flows are retained, allowing a more efficient solution for the estimation of the tail index parameter.

In what follows, we refer to $P(x)$ as the flow *survival function*, as it captures the probability of a flow surviving, i.e., staying active, to transmit at least x more bytes. The survival function is used in the calculation of the *expected residual life* $e(x)$ of a flow with given length x (i.e., the average number of bytes that the flow transmits before concluding its transmission), and it is defined as

$$e(x) = \int_x^{\infty} \frac{P(u)}{P(x)}du. \tag{8}$$

C. Active Flow Histogram

For fully characterizing the TCP workload, the traffic profiler must count the number of active TCP flows along with the service they have received so far from the network, which is represented by the number of packets they have transmitted so far (i.e., their flow length). All monitored TCP flows are organized in a flow histogram, based on their current flow length, which is updated online on a packet-by-packet basis. Inactive flows, i.e., the ones exceeding a time threshold of inactivity, are removed from the histogram. On this basis, it is possible to estimate the overall number of active flows and their expected residual life, which constitute the two fundamental metrics for characterizing any work-conserving queueing system and predicting its evolution.

Flow length denotes the service that a flow has received so far from the network, and it serves as a basis to estimate its expected residual life. Specifically, given *flow* i with length x_i, the probability that flow i transmits y_i bytes before completion (i.e., its *survival probability*) is derived as

$$R_i = Pr\left[X > x_i + y_i | X > x_i\right] = \frac{P\left(x_i + y_i\right)}{P\left(x_i\right)} = \left(\frac{x_i}{x_i + y_i}\right)^{\alpha}, \tag{9}$$

where X is the Pareto random variable that corresponds to the flow length and $P(.)$ the flow survival function output by the flow profiler.

Since it is infeasible to keep track of every active flow and its residual life, a frequency histogram is constructed for keeping track of the length distribution of sampled flows. To construct the histogram, the flow sample is partitioned into a number of discrete intervals, or bins, with each bin corresponding to a range of flow lengths. For example the k^{th} bin, denoted as B_k, corresponds to NF_k flows with a length in the range of $(r_k, r_{k+1}]$ bytes. Due to the unbiased selection of the flow sample, the number of unsampled flows that fall into B_k and have a length of $(r_k, r_{k+1}]$ converges to the number of sampled flows NF_k multiplied by N, which is the inverse of the sampling rate. Thus, the calculation of $N \cdot NF_k$ constitutes an

unbiased scaling estimation of the number of unsampled active flows with a length of $(r_k, r_{k+1}]$. Regarding the standard error of this estimation, it is derived from the number of active flows on NF_k as well as the profiler sampling rate N, and is given by [16]

$$\frac{\sqrt{\mathrm{Var}(\widehat{NF}_k)}}{NF_k} = \sqrt{\frac{N}{NF_k}}. \tag{10}$$

It can be seen that the standard estimation error is small, and it can be made arbitrarily small using a high sampling rate. It is also evident that histogram estimators are not efficient for estimating small frequencies, and thus a very small NF_k (compared to the sampling ratio) leads to a high variance. Due to the heavy-tailed nature of flow duration, very long flows appear with small probabilities, leading to almost empty rightmost bins of the flow histogram, which can cause a high variance, as shown in Eq. (10). Thus, for a given error bound, long-lived flows exceeding a certain threshold termed the *long-lived flow threshold* are pooled to the rightmost bin of the histogram. The survival probability of these flows is approximated as 1, based on a fundamental property of the heavy-tailed random variable $X \lim_{x_i \to \infty} Pr\left[X > x_i + y_i | X > x_i\right] = 1$.

Another source of errors in frequency histograms is the smoothing error induced on the frequency distribution, as information of individual flow lengths in a bin is lost. Flow lengths in B_k are represented by the mid-value L_k, thus leading to an error $\Delta x_i = |x_i - L_k|, x_i \in B_k$. This error is quantified with the standard deviation metric $stdev(B_k)$ and clearly depends on the bin width. The uncertainty in the flow length $x_i \in B_k$, which is approximated as $x_i \approx L_k$, further propagates to the calculation of the flow survival probability R_i. The standard deviation metric $stdev(R_i)$ is derived using well-known error propagation formulas in Eq. (9) as

$$\frac{stdev(R_i)}{R_i} = \alpha \times \frac{stdev(B_k)}{x_i} \times \frac{y_i}{x_i + y_i}. \tag{11}$$

Based on the error estimates derived from Eqs. (10) and (11) per histogram bin, we empirically design the active flow histogram by selecting the appropriate bin widths so that the standard estimation error stays under 5%. As regards the edge bin that extends to infinity, the long-lived flow threshold serves as its left bound, indirectly determining the overall number of bins.

V. Online Burst Size Prediction

OBS can offer significant statistical multiplexing gains, due to its ability to support sub-wavelength switching in the optical domain. However, its bufferless nature and unacknowledged transmission of bursts can lead to a high degree of losses due to contention. A viable solution that has been proposed in the literature is advance reservation of resources [8], which closely keeps track of the prevailing traffic characteristics. The implementation of such a dynamic scheme requires an efficient burst prediction mechanism as well as a signaling protocol that communicates to the core routers upcoming traffic changes. This approach, given a reasonably accurate prediction method, can combine the performance and quality of service (QoS) guarantees of two-way signaling with the low latency of one-way signaling [10].

In this section, we detail the prediction mechanism required to estimate burst sizes, based on the online traffic measurements obtained from the traffic profiler. The proposed prediction scheme takes advantage of traffic predictability at the flow level to obtain aggregate TCP traffic and eventually burst size predictions. Specifically, it exploits the fact that newly arrived flows have to probe for network capacity, by starting their transmission with a congestion window of one segment which is doubled in every lossless round. Thus, it is possible to analytically and in real time estimate the expected TCP throughput increase due to the increase of flows' congestion windows or decrease due to burst losses [24].

In what follows, the time is divided into prediction intervals, whose duration is denoted with τ. The discrete time rate process of flow m is denoted with $S_n^{(m)}$, and is derived from Eq. (5) as the number of segments transmitted in the n^{th} prediction interval. The predictor at the end of each prediction interval, at $t = n \cdot \tau$, estimates the aggregate number of bytes to be transmitted in the following prediction interval, that is up to $t = (n+1)\tau$. As mentioned in the previous section, the aggregate rate $R(t)$ is obtained as the multiplex of individual TCP flows. Thus, according to the law of large numbers, the aggregate number of packets transmitted by N_F active flows in the n^{th} prediction interval converges to

$$E[R(t)] = N_F \times E[S_n]. \tag{12}$$

In the above equation, we use the assumption of uncongested backbone links, which allows us to claim that the TCP flow rates are independent. This is a valid assumption, as in most OBS studies the burst loss ratio never exceeds 1%. In what follows, we will focus on predicting the expected number of bytes transmitted from active flows within one prediction interval, based on flow statistics maintained by the flow profiler.

A. Predicting TCP Flow Rate

There is a large body of literature on modeling the throughput of TCP flows over OBS as a function of flow and network parameters. Typically, macroscopic TCP models assume perfect periodicity of TCP flows, which are assumed to follow a saw-tooth profile and have unlimited data to send. Then, TCP performance is reduced to a closed-form equation that only takes into account steady-state performance. However, this approach yields a small accuracy in real-world networks, where flows have a finite duration, and it can take a significant amount of time, compared to flow lifetime, to reach the steady state. In this case, slow-start performance becomes significant and thus it has to be taken into account.

In this work, we take into account both the slow-start and steady-state phases, to predict the sending rate of TCP flows, as defined in the discrete time rate process in Eq. (5). We assume an idealized scenario in which the trajectory followed by the TCP congestion window is deterministic. Each TCP flow starts its transmission in the slow-start phase with a congestion window of one segment. It then exponentially grows

to the steady-state window, until it concludes its transmission. The fast-retransmit and time-out periods are not taken into account, as they have a small contribution to the TCP throughput. The aforementioned simplifications are common to TCP models, and do not introduce significant inaccuracies as long as the burst loss ratio is kept small. For both the slow-start and steady-state phases, the TCP throughput is obtained from well-known performance modeling formulas. Specifically, we predict the sending rate of a TCP flow, denoted as flow i, as the number of segments S_i transmitted in the following *prediction interval*. S_i is derived as a function of flow parameters SPB_i, RTT_i, and L_i, i.e., the segment-per-burst ratio, round trip time, and flow length, respectively:

$$S_i = S\left(\mathrm{SPB}_i, R_i, L_i\right). \tag{13}$$

The steady-state TCP window is obtained from [4], a formula which is accurate for small values of burst loss ratio p (*typically* $p < 1\%$):

$$\overline{\mathrm{cwnd}} = \min\left\{\frac{\sqrt{1.5 \times \mathrm{SPB}_i}}{\sqrt{p}}, W_m\right\}, \tag{14}$$

where W_m is the TCP maximum window size (in segments) that is constrained from the advertised TCP window as well as the flow access rate. As long as the flow congestion window cwnd_i does not exceed $\overline{\mathrm{cwnd}}$, the flow is assumed to be in the slow-start phase. The instant flow congestion window in the slow-start phase can be derived from its flow length [25] as

$$\mathrm{cwnd}_i = \frac{L_i}{2}. \tag{15}$$

Regarding the slow-start performance, it is modeled in rounds of RTT duration. Given τ the duration of one prediction interval, we denote $r = \tau/\mathrm{RTT}_i$ as the duration of the prediction interval in TCP rounds. The number of segments transmitted from a (lossless) slow-start flow in r rounds, starting with a congestion window of cwnd_k, is [25]

$$S_i = \mathrm{cwnd}_i \times (2^r - 1). \tag{16}$$

After cwnd_i exceeds $\overline{\mathrm{cwnd}}$, we assume that the flow has reached a steady state. As soon as the flow reaches the steady state, $\overline{\mathrm{cwnd}}$ segments are transmitted on average per round, or $r \cdot \overline{\mathrm{cwnd}}$ segments per prediction interval. Overall, the number of segments transmitted within a prediction interval from flow i either in the slow-start phase or the steady-state phase is

$$S_i = \begin{cases} \mathrm{cwnd}_i \times \left(2^r - 1\right), & \text{when } \mathrm{cwnd}_i \le \overline{\mathrm{cwnd}} \\ r \times \overline{\mathrm{cwnd}}, & \text{when } \mathrm{cwnd}_i > \overline{\mathrm{cwnd}} \end{cases}. \tag{17}$$

B. Predicting Burst Sizes

In previous sections, we have shown how to characterize the network workload with the proposed traffic model, and how to predict the evolution of individual TCP flow rates. These are combined for deriving aggregate TCP throughput and burst size predictions for the following prediction interval. It must be noted here that the complexity and burstiness of the flow arrival process can significantly affect the number of active flows and their distribution in the active flow histogram. For example, a sudden burst of connection arrivals propagates from the leftmost bins of the histogram to the rightmost ones, as the newly arrived flows transmit packets, gradually increasing their congestion windows.

The flow profiler detailed in the previous section provides online estimates of the pairs $\{NF_k, length_k\}$ that correspond to the number of active flows NF_k assigned to the k^{th} bin of the histogram, and have a flow length of $length_k$. In what follows, we rely on flow-level statistics of active TCP connections output by the flow profiler along with expected residual life estimates, as expressed in Eq. (8), to analytically derive the aggregate TCP throughput for the following prediction interval.

Given SPB_i, RTT_i, and L_i, the TCP parameters of flow i, and $S(\mathrm{SPB}_i, \mathrm{RTT}_i, L_i)$ its sending rate derived from Eq. (17), the expected number of bytes it transmits in the following time interval, before concluding its transmission, is obtained by integrating the flow survival function $P(.)$:

$$B(\mathrm{SPB}_i, \mathrm{RTT}_i, L_i) = \int_{y_i}^{y_i+S} \left(\mathrm{SPB}_i, \mathrm{RTT}_i, L_i\right) \times \mathrm{MSS}\frac{P(u)}{P(y_i)}du, \tag{18}$$

where $y_i = \mathrm{MSS} \cdot L_i$ is the byte length of flow i and MSS the network maximum segment size. Next, we derive the expected number of bytes transmitted from all active flows assigned to one histogram bin, denoted as bin k. For doing so, we assume that all flows in a bin have the same length, which corresponds to the mid-value of the bin range and is $length_k$. Further, using the empirical distributions of the round trip times and segment-per-burst ratios estimated by the profiler, i.e., $\mathrm{RTT}(i)$ corresponds to the frequency of RTT_i value and $\mathrm{SPB}(i)$ to the frequency of SPB_i value, the expected number of bytes transmitted from all flows assigned to the bin k histogram bin H_k is derived as

$$E[H_k] = NF_k \times \sum_{i=1}^{\infty} \mathrm{SPB}(i) \times \sum_{j=1}^{\infty} \mathrm{RTT}(j) \times B\left(\mathrm{SPB}_i, \mathrm{RTT}_j, L_k\right). \tag{19}$$

Thus, the predicted number of bytes transmitted from all active flows, assuming M bins in the flow histogram, is

$$\widetilde{B} = \sum_{k=1}^{M} E[H_k]. \tag{20}$$

Finally, the predicted burst size is obtained by equally dividing $\widetilde{B}$ with the number of bursts created during the prediction interval τ , assuming that T_{bat} is the assembly time:

$$\widetilde{L} = \left(\frac{T_{\mathrm{bat}}}{\tau}\right) \times \widetilde{B}. \tag{21}$$

Clearly, this scheme assumes constant burst sizes within the same prediction interval. This assumption is valid when the traffic profile within the prediction interval is relatively

smooth, which depends on input load variations. The effect of this error in burst size prediction will be evaluated in the following sections. In any case, it must be noted that TCP traffic is expected to exhibit a degree of sub-RTT burstiness, which cannot be captured by the proposed scheme. However, the accuracy in the proposed burst size prediction scheme can be improved using either smaller interval periods and/or a worst-case correction parameter, to be discussed in the following section.

VI. Prediction-Based Reservation of Resources

The prediction of future burst sizes using TCP dynamics, detailed in the previous sections, can be used for advance notifying the core nodes of upcoming traffic changes per prediction interval. In this section, we propose a resource reservation scheme, in which edge nodes send a single setup (or refresh) message per end destination per prediction interval, in order to communicate to the core nodes across the path the aggregate traffic of the next prediction interval. This allows core nodes to reschedule their resources in view of the updated (predicted) traffic conditions. In this way, optimization of the link utilization and minimization of burst losses can be achieved.

A number of one-way reservation schemes has been proposed for OBS networks, including JET, JIT, and Horizon [26]. Resource reservation in one-way OBS networks is typically handled from link schedulers, which assign wavelengths to incoming bursts based on an online scheduling algorithm. However, online scheduling algorithms such as LAUC [27] are best effort in nature and often make sub-optimal decisions. At the reception of a burst header packet (BHP), the link scheduler searches for unscheduled channels that can service the corresponding burst and selects the best available at that time. Since the scheduler has no information for future arrivals, bursts are never blocked as long as a feasible wavelength can be found, even if dropping a burst would maximize the overall number of accepted ones. Additionally, channel selection in GOLs is sub-optimal when BHPs arrive at a random order, and not in the order of burst arrival times [28]. This issue is common in OBS networks, where differences in the burst offset times and propagation times can lead to burst losses even in scenarios where a feasible scheduling is possible [28]. Here, we propose the use of one SETUP message per source–destination pair in the beginning of each prediction interval to notify the core routers of the future upcoming burst sizes. Thus, core routers can use batch scheduling algorithms, achieving a decreased burst loss ratio due to the more efficient use of resources. It is important to note that the use of one SETUP message does not affect the different classes of service that may be employed in the network. The single setup message is enough to carry the burst overhead information (burst start time, predicted burst size) from all priority classes per source/destination pair.

A. Signaling Protocol

The proposed reservation protocol is based on the observation that the arrival time of the bursts at the core nodes

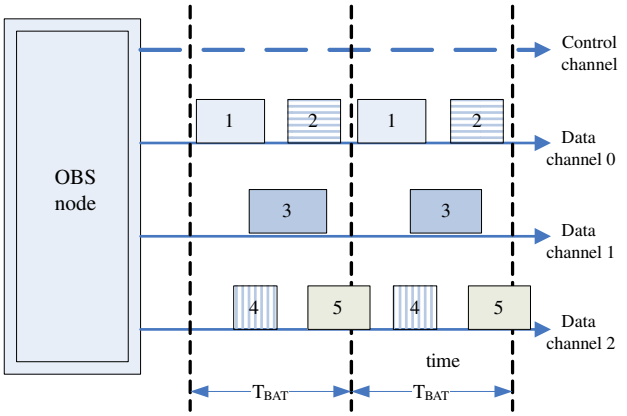

Fig. 1. (Color online) Link scheduler utilization profile, repeated periodically on a T_{BAT} basis.

is periodic, with a period equal to the burst assembly time (T_{bat}). This periodicity allows burst arrival times at individual core nodes to be accurately predicted. This is shown in Fig. 1, which illustrates the scheduler's utilization profile for a single outgoing link. The utilization profile consists of a set of burst arrivals, each scheduled for a specific wavelength. Bursts that belong to the same source–destination pair (and thus have been assigned to the same assembly queue) are marked with the same color. These are assumed to belong to the same Class of Service (CoS), and are assigned the same CoS identifier.

The proposed reservation protocol augments the standard one-way JET signaling with a SETUP message that carries the predicted burst size. One such message is sent per source/destination pair at the beginning of the prediction interval. As seen in Fig. 2, the SETUP message propagates downstream and notifies the link schedulers along the path with the burst arrival time as well as their predicted size for the following prediction interval. At the reception of this SETUP message by a core node, resources are reserved for all the bursts of the prediction interval, while actual bursts will arrive at a later time according to the predicted arrival times. This scheme allows schedulers to obtain a priori knowledge of all bursts competing for each outgoing link, and use a more efficient batch channel scheduling algorithm. The assignment of wavelengths to bursts can thus be performed once per prediction interval for each CoS (or equivalently source–destination pair), alleviating the need for per-burst scheduling.

The wavelength allocation in each link scheduler is performed as soon as a new SETUP message is received. Upon its reception a scheduling algorithm is executed looking for a scheduling solution that satisfies the requirements of all source–destination pairs competing for the same outgoing link. In order to accommodate the updated (or new) burst size, rescheduling of previously scheduled bursts in a different wavelength or even burst dropping may be required. The CoS identifiers of unscheduled bursts (i.e., failed to be scheduled due to lack of resources) are piggy-backed in the SETUP message, so they are not considered for scheduling in the downstream nodes. Link-state information in the proposed scheme is stored in one ordered list per outgoing link, sorted by burst arrival time. For each source–destination pair, the

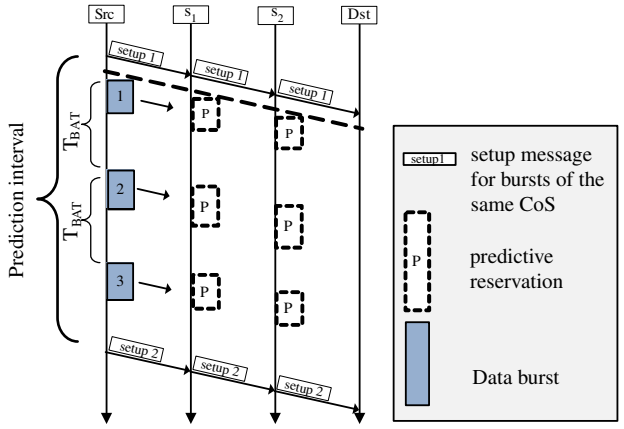

Fig. 2. (Color online) Timing considerations of the proposed predictive reservation protocol. Dots denote burst reservations based on predictions, which are communicated to all nodes across the path with a single setup message. Actual bursts arrive at a later (predicted) time.

scheduler stores the state information of the burst arrival time and the predicted burst length, as well as its assigned outgoing wavelength.

B. Scheduling Algorithm

As mentioned in the previous section and shown in Fig. 1, the arrival time of the bursts at the core nodes is periodic, with a period of T_{BAT}. Thus, the link-state timing information can be mapped in a $[0, T_{BAT})$ interval using modular (residue) arithmetic. The timing information of burst b_i, i.e., its start time s_i and its end time e_i, is represented as

$$s_i = \text{start\_time}(i) \bmod T_{\text{bat}}$$
$$e_i = \big(s_i + \text{length}(i)\big) \bmod T_{\text{bat}},$$

where start_time(i) and length(i) are the arrival time and the predicted length of b_i. It must be noted here that the definition of modulo operator is extended to include real numbers. For positive real numbers a and n, we define $a \bmod n = a - i \times n$, for integer $i \geq 0$ and $0 \leq (a \bmod n) < n$. Bursts that belong to the same CoS have the same arrival time and the same (predicted) end time in modulo representation, and thus burst scheduling is performed per CoS.

The batch scheduling and wavelength assignment problem can be solved optimally by the Arkin and Silverberg (AS) algorithm [29], which takes as input all $[s_i, e_i]$ intervals that correspond to burst start and end times. Bursts that wrap around T_{bat} and have $e_i < s_i$ are split into two parts, i.e., $[0, e_i]$ and $[s_i, T_{\text{bat}}]$, with the added constraint that they must both be scheduled on the same wavelength. However, the AS algorithm has a high computational complexity of n^{k+1}, which is prohibitive for networks with a high number of wavelengths (for example, $k \geq 5$). Thus, we considered here more efficient albeit non-optimal scheduling algorithms, suitable for scheduling problems with fixed start and end times. Such a scheduling problem can be modeled as a special case of the interval scheduling with machine availabilities (ISMA) problem. Given a set of k wavelengths, we first schedule each split interval in a different wavelength w_i, as all split intervals overlap with each other. Thus, wavelength w_i is now only *available* in a subset of the interval $[0, T_{\text{bat}})$. Then, we schedule the rest of the bursts using the GOL [30]. GOL traverses the intervals stored in the scheduler's list sequentially, sorted in the order of their arrival time. In this way, bursts are scheduled in a greedy fashion and there is no backtracking. The basic algorithmic steps of GOL that output a solution for the ISMA problem in $O(n \times k)$ time are as follows.

1. Empty the set S, which corresponds to the set of scheduled intervals.
2. Sequentially consider the intervals, stored in the scheduler's ordered list, in the order of their arrival times.
3. Add the next interval to S. If there is no available wavelength to accommodate it, remove from S the interval with the latest ending time.

Interestingly, the solution output by GOL is very close to the optimal solution for the ISMA problem. The authors in [30] showed that GOL outputs a schedule that is only $k-1$ jobs off the theoretical optimum in the worst case, where k is the number of wavelengths, independent of the number n of scheduled bursts.

C. Correction Parameter

Since batch burst-level resource reservations over the end-to-end path are based on the predicted burst size, there is a possibility of overestimation or underestimation of the actual burst size. An overestimation wastes an amount of resources, while an underestimation can result in an insufficient resource reservation and result in dropped bursts. The probability of underestimation in predictive reservation protocols is usually compensated with a correction parameter δ, which is added to the predicted value $\tilde{L}$ and thus incurs a percentage of bandwidth wastage. There is an obvious trade-off in the selection of the correction parameter, between minimizing the underestimation probability and increasing the bandwidth cost. As shown in [8], the probability P_s that the predicted burst size exceeds the real burst size, for $Q(.)$ the Q-function and $e(k)$ the prediction error distribution, is

$$P_s = P\left(e(k) < d\right) = \int_{-\infty}^{\delta} f\left(e(k)\right) de(k). \tag{22}$$

Assuming that the error distribution $e(k)$ resembles white noise, i.e., zero-mean Gaussian distribution with standard deviation σ, we obtain

$$P_s = \frac{1}{\sqrt{2\pi}\cdot\sigma} \int_{-\infty}^{\delta} e^{\frac{-e^2(k)}{2\sigma^2}} de(k) = 1 - Q\left(\frac{\delta}{\sigma}\right). \tag{23}$$

It can be seen from Eq. (23) that, by choosing the parameter δ to be a multiple of the root mean square error (RMSE) of the prediction error, $\delta = a \cdot \sigma$, we can explicitly bound the underestimation probability. For example, for $a = 2$, the underestimation probability becomes $1 - P_s = Q(2) = 5\%$.

VII. Performance Evaluation

In this section, we evaluate the proposed TCP-specific prediction mechanism, as well as the proposed predictive reservation protocol using the ns-2 simulation platform. In the first set of experiments, the TCP flow profiler as a prediction mechanism in OBS networks is evaluated using a simple three-node topology. In the second set of experiments, we evaluate the proposed predictive reservation protocol over the NSF network topology to disclose the performance gains obtained from the TCP prediction mechanism.

A. Evaluation of the Traffic Predictor

For evaluating the proposed TCP-specific profiling and prediction mechanism a simple three-node network topology was modeled, which consists of two edge nodes and one core node. A timer-based burst assembly scheme was implemented at the edge nodes with a time threshold (T_{bat}) of 3 ms, while a JET signaling scheme was employed for one-way resource reservations. The network round trip time was set to 15 ms, while all clients had a uniformly selected access rate of 20 Mpbs, 50 Mpbs and 100 Mpbs. With respect to network traffic, a representative scenario of typical Internet workloads was modeled, which consists of a traffic mix of short-lived connection requests with a mean size of 50 kB, and long-lived file transfers with sizes drawn from a Pareto distribution, with an average of 3 MB and an index parameter of 1.6. Short-lived traffic was generated with packmime HTTP 1.1 traffic generator [31], so as to model interactive web requests.

The flow profiler proposed in this work was implemented as a separate ns-2 module and integrated in the edge nodes' burst assembly units. The flow profiler updates the active flow histogram in real time, assigning sampled flows to the corresponding histogram bins. Figure 3 displays the output of the flow histogram over time for a snapshot of the simulation cycle, in the form of number of flows assigned to bins 1 to 4. The increases/decreases in the arrival rate can be clearly seen, propagating from bin 1 to bin 4, with a small lag that corresponds to the RTT. At the same time, the flow congestion window doubles per RTT, until the flow reaches a steady state (or it concludes its transmission). This lag is exploited by the proposed scheme to perform short-term predictions (in the order of a few RTTs). As expected, a percentage of flows (depending on the flow size density function) will conclude before being assigned to the next bin, and thus flow population decreases when moving from bin 1 to bin 4. In particular, bin 1 has on average 420 active flows, while bin 4 has only 200.

At the beginning of each prediction interval, the average TCP throughput of active flows was predicted, based on flow statistics gathered by the profiler. This provides an estimate of the expected burst size for the next prediction interval, which was compared to the actual burst size. Figure 4 displays the aggregated throughput of TCP flows compared with the predicted value in a snapshot of the simulation cycle, for prediction intervals of 1, 2, and 4 times the RTT. From Fig. 4, it can be seen that the predicted values closely follow the true one of the aggregated throughput, while they converge quickly enough to meet the changes in the flow arrival rate. It is

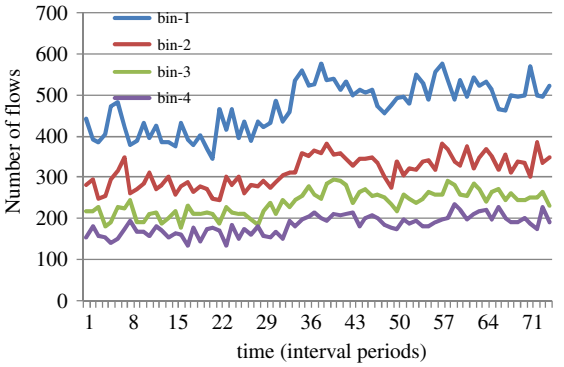

Fig. 3. (Color online) Evolution of number of active flows per bin.

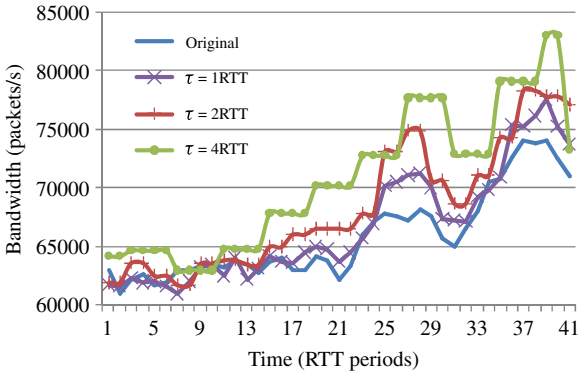

Fig. 4. (Color online) Evolution of aggregated throughput for prediction intervals of 1, 2, and 4 times the RTT.

evident that the lag in the convergence time is proportional to the interval length, which can cause a high variance for long time intervals (i.e., 4 times the RTT) especially when a sudden change in the arrival rate occurs. Thus, the best performance is obtained with the shortest prediction interval, that of 1 RTT.

For evaluating the accuracy of the proposed burst prediction scheme for different prediction intervals, the relative prediction error CDF (see Fig. 5) was calculated. It must be noted here that the predicted burst size is calculated once at the beginning of each prediction interval, and then assumed constant. Thus, even for the smallest prediction interval of 1 RTT, we expect a degree of variability when predicting burst sizes, due to the sub-RTT burstiness caused by TCP protocol dynamics. Our analysis shows that the relative prediction error for the 95% of the bursts is less than 8%, for a prediction interval of 1 RTT. For longer prediction intervals (>2 RTTs) there is an increase in the burst prediction error up to 10%.

Finally, we evaluated the effect of the sampling rate on burst prediction accuracy by compiling a table (Table I) of the value of the coefficient of variation (CoV) for different sampling periods $N = \{2, 5, 10, 20\}$ and different prediction intervals. As expected, a smaller sampling rate leads to lower prediction accuracy, due to the loss of information incurred by traffic sampling. It must be noted here that our traffic profile constitutes a scaled-down version of a typical Internet workload, which would consist of millions instead of thousands of flows. This explains the small

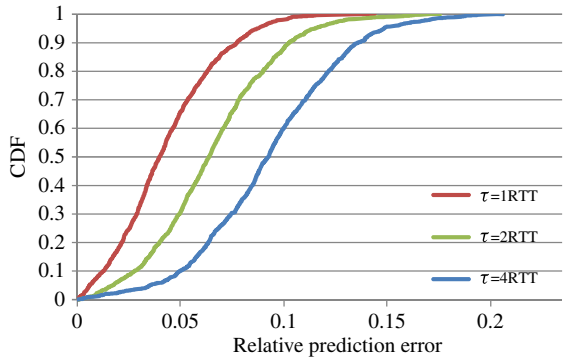

Fig. 5. (Color online) CDF of the burst size relative prediction error for prediction intervals of 1, 2, and 4 times the RTT.

TABLE I
COEFFICIENT OF VARIATION (COV) FOR DIFFERENT PREDICTION INTERVALS AND SAMPLING RATES

	$N = 2$	$N = 5$	$N = 10$	$N = 20$
1RTT	0.023	0.041	0.07	0.11
2RTT	0.038	0.05	0.076	0.11
4RTT	0.064	0.073	0.094	0.13

sampling periods used in our experiments, which were again 2–3 orders of magnitude smaller than the original.

B. Evaluation of the Predictive Reservation Protocol

The second set of experiments was performed on the 14-node NSFnet topology, for evaluating the performance of the proposed predictive reservation protocol in a real-world network topology. As shown in the previous set of experiments, the selection of the prediction interval has a significant effect in prediction accuracy, with the best performance results obtained with a value equal to the path RTT. In this set of experiments, we consider the prediction interval set equal to the path RTT for each source–destination pair. This means that different prediction intervals are employed for the different source–destination pairs. In what follows, our goal is to explore the performance gains of the proposed scheme in terms of burst loss ratio for different sampling rates. In our experiments, we considered four wavelengths per link with full wavelength conversion capability, at 10 Gbps capacity. The traffic profile used was a mix of short-lived web requests generated with packmime tool that accounted for 20% of bytes transmitted and a set of long-lived file transfers that accounted for 80% of bytes transmitted. File transfer requests were initiated among all source–destination pairs in the network, with file sizes drawn from a Pareto distribution with an average of 3 MB. What is more, the request interarrival times follow an exponential distribution, whose mean value is determined by the network load. It must be noted here that the multiplexing of Pareto traffic flows is expected to yield self-similar traffic, whose Hurst parameter H (a metric of traffic burstiness) is determined by the selection of the Pareto shape parameter. Simulation experiments were set to 150 s that correspond to up to 50 000 generated bursts (depending on the traffic load) with a 3 ms timer. All results were obtained as average values of three independent runs.

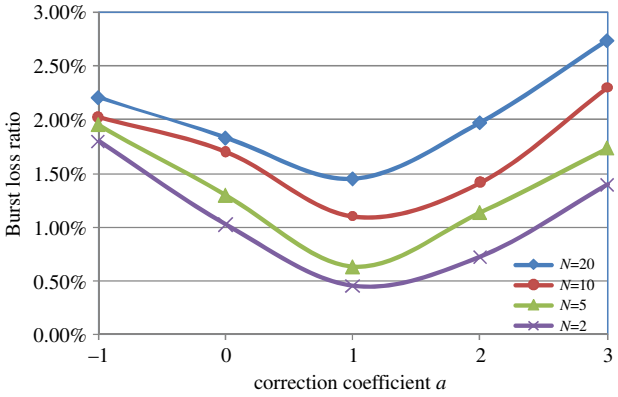

Fig. 6. (Color online) Burst loss ratio for different α coefficient of the correction parameter δ, for a constant 0.7 load and 0.7 Hurst parameter.

A correction parameter δ, as described in the previous section, is added in the predicted burst length to alleviate the predictor's underestimation probability. In our first experiment, we evaluated the effect of the correction parameter, when varying the α coefficient, on the performance of the reservation protocol, keeping the input load constant. Figure 6 displays the burst loss ratio versus α coefficient for different sampling ratios. It can be seen that $\alpha = 1$ is the value that better balances losses, due to burst length underestimations and wasted capacity due to increased resource reservations. For $\alpha > 1$, the bandwidth wastage overshadows the gains of the decreased underestimation probability. Thus, in what follows we consider $\delta = \sigma$ to be the correction parameter of choice, where σ is the RMSE of the burst prediction. It is worth noting here that the expected degree of bandwidth wastage in the network can be quantified with the CoV metric (see Table I), which is defined as the ratio of the RMSE to the mean burst size. For example, for a CoV of 0.07, the percentage of burst overestimation (equivalently percentage of bandwidth waste) is 7%.

In the next experiment, we compare the performance of the proposed reservation protocol with the standard JET reservation protocol. The performance evaluation (see Fig. 7) is based on the burst loss ratio for different network loads and sampling periods. We used the LAUC-VF scheduling algorithm with JET reservation protocol and GOL scheduling with the proposed predictive reservation protocol. It must be noted here that both reservation protocols use one-way signaling and online scheduling algorithms. The performance enhancements achieved with the proposed reservation protocol are due to the efficient channel scheduling facilitated by the predictive batch reservation of resources, without increasing the edge delay. To further evaluate the effectiveness of the proposed protocol, it is also compared with the AS optimal scheduling algorithm solely for the case of $N = 2$.

From Fig. 7, it can be seen that the proposed reservation protocol can achieve significant performance enhancements compared to the standard JET protocol with LAUC signaling. The performance gains are affected by the sampling period used, but, even with $N = 20$, the gain in burst loss ratio can be clearly seen. Further, the use of an optimal scheduling

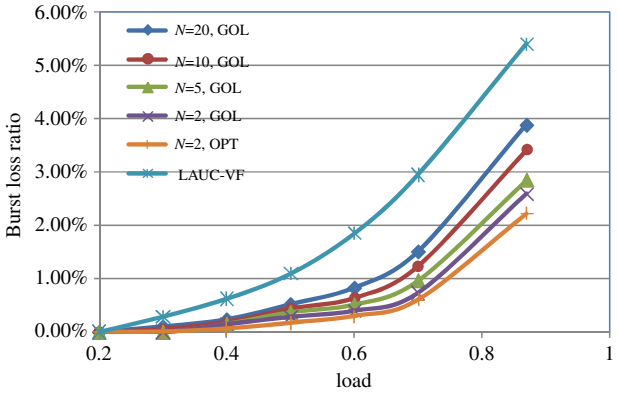

Fig. 7. (Color online) Burst loss ratio versus load comparison of proposed reservation protocol and JET, with different sampling periods N and scheduling algorithms. The Hurst parameter was kept constant at $H = 0.7$.

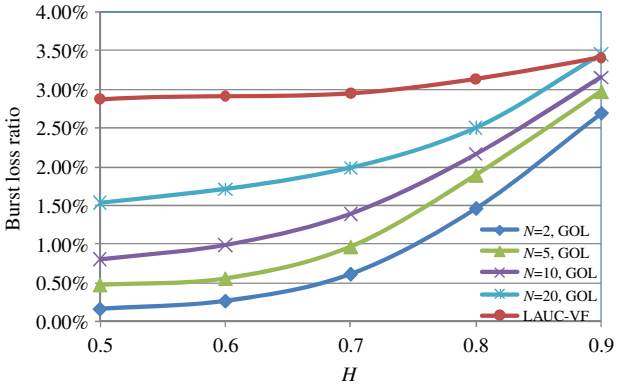

Fig. 8. (Color online) Burst loss ratio versus Hurst parameter for predictive reservation and JET. We kept the load constant, at 0.7.

algorithm (marked as OPT in Fig. 7) reveals only a marginal decrease in burst loss ratio. This confirms the effectiveness of the proposed predictive resource reservation scheme that uses a TCP-specific profiling mechanism and a greedy online scheduling algorithm, at a fraction of the computation time. In Fig. 7, load corresponds to the normalized amount of traffic generated per node, which is a fraction of the link capacity (in our simulations it varies from 0.2 to 0.9). Burst loss ratios at lower loads are not reported due to the low confidence of measurements (a few losses per thousands of bursts transmitted). The accuracy of the measured burst loss ratios can be derived from Eq. (4) using $W = 3 \times 50\,000$, since three independent runs were performed per simulation cycle and 50 000 bursts were generated in each one of them.

Finally, we evaluated the effect of traffic burstiness on the predictive protocol performance. By varying the shape parameter of the Pareto process that outputs the burst sizes, we were able to generate traffic with a varying Hurst parameter. The performance of the JET protocol with LAUC-VF scheduling was also measured for reference. As the Hurst parameter increases, traffic by definition becomes more bursty and unpredictable. This can undermine our assumption of constant burst sizes per prediction interval and adversely affect our protocol's performance. Indeed, from Fig. 8, it can be seen that the difficulty of predicting burst sizes in very bursty traffic (when $H \to 1$) has a negative impact on the burst loss ratio, especially for small sampling ratios. Due to the high prediction error, the performance continuously deteriorates, but in all cases it performs better than JET with LAUC-VF.

In particular, for the most common Hurst parameters (i.e., $H \leq 0.75$), the proposed reservation protocol clearly outperforms LAUC-VF.

VIII. Conclusions

In this paper, a TCP-specific profiling and prediction scheme is proposed, suitable for enhancing the performance of TCP transmission over OBS networks. The scheme relies on flow statistics, estimated by a flow profiler to perform running online estimations of parameters such as the burst loss ratio and number of active flows as well as other flow-level statistics such as segment-per-burst distribution and flow RTT. These are then used to estimate the aggregated TCP throughput and the corresponding burst sizes for time intervals relative to the RTT.

Performance evaluation results have shown that the proposed mechanism adequately profiles flow dynamics and estimates burst sizes at sub-RTT time intervals with an error of less than 10% for the majority of the bursts transmitted. The performance gains in an one-way OBS network were shown using a modified resource reservation protocol that reserves resources based on the burst length predictions for all the bursts within a prediction interval. The gain in burst loss ratio even for a low sampling ratio is significant.

Acknowledgments

This research has been co-financed by the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) (Research Funding Program: Heracleitus II) and also supported by 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] C. Qiao and M. Yoo, "Optical burst switching (OBS)-A new paradigm for an optical internet," *J. High Speed Netw.*, vol. 8, no. 1, pp. 69–84, 1999.

[2] O. González, A. M. Guidotti, C. Raffaelli, K. Ramantas, and K. Vlachos, "On transmission control protocol synchronization in optical burst switching," *Photonic Netw. Commun.*, vol. 18, no. 3, pp. 323–333, Mar. 2009.

[3] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proc. ACM IMC*, 2009.

[4] X. Yu, C. Qiao, Y. Liu, and D. Towsley, "Performance evaluation of TCP implementations in OBS networks," *Technical Report 2003-13*, CSE Dept., SUNY, Buffalo, 2003.

[5] X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false time out detection in OBS networks," in *Proc. of IEEE INFOCOM*, 2004, vol. 2, pp. 774–784.

[6] B. Shihada, P.-H. Ho, and Q. Zhang, "A novel congestion detection scheme in TCP over OBS networks," *J. Lightwave Technol.*, vol. 27, no. 4, pp. 386–395, 2009.

[7] Q. Zhang, V. M. Vokkarane, Y. Wang, and J. P. Jue, "Analysis of TCP over optical burst-switched networks with burst retransmission," *IEEE Globecom*, vol. 4, p. 6, 2005, 1983.

[8] J. Liu, N. Ansari, and T. Ott, "FRR for latency reduction and QoS provisioning in OBS networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 7, pp. 1210–1219, Sept. 2003.

[9] T. Karagiannis, M. Molle, and M. Faloutsos, "Long-range dependence ten years of Internet traffic modeling," *Internet Comput., IEEE*, vol. 8, no. 5, pp. 57–64, 2004.

[10] K. Vlachos and D. Monoyios, "A virtual one-way signaling protocol with aggressive resource reservation for improving burst transmission delay," *J. Lightwave technol.*, vol. 27, no. 4, pp. 2869–2875, 2009.

[11] O. Pedrola, S. Rumley, D. Careglio, M. Klinkowski, P. Pedroso, J. Sole-Pareta, and C. Gaumier, "A performance survey on deflection routing techniques for OBS networks," in *Proc. of 11th Int. Conf. Transparent Optical Networks (ICTON '09)*, 2009, pp. 1–6.

[12] R. R. C Bikram, N. Charbonneau, and V. M. Vokkarane, "Coordinated multi-layer loss recovery in TCP over optical burst-switched (OBS) networks," in *Proc. IEEE Int. Conf. Communications*, 2010, pp. 1–5.

[13] C. Cameron, H. Le Vu, J. Choi, S. Bilgrami, M. Zukerman, and M. Kang, "TCP over OBS—fixed-point load and loss," *Opt. Express*, vol. 13, no. 23, pp. 9167–9174, 2005.

[14] Cisco Systems, *NetFlow Services and Applications*, 2000, white paper.

[15] K. Ramantas and K. Vlachos, "Profiling TCP traffic in optical burst switching networks," in *Proc. of ICST BROADNETS*, 2010.

[16] N. Duffield, C. Lund, and M. Thorup, "Properties and prediction of flow statistics from sampled packet streams," in *Proc. of 2nd ACM SIGCOMM Workshop on Internet Measurement*, 2002.

[17] N.G. Duffield, "Sampling for passive internet measurement: a review," *Statist. Sci.*, vol. 19, no. 3, pp. 472–498, 2004.

[18] H. Jiang and C. Dovrolis, "Passive estimation of TCP round-trip times," in *Proc. of ACM SIGCOMM*, 2002.

[19] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani, "Counter braids: a novel counter architecture for per-flow measurement," in *Proc. of ACM SIGMETRICS '08*, 2008.

[20] M. Yajnik, S. B. Moon, J. Kurose, and D. Towsley, "Measurement and modeling of the temporal dependence in packet loss," *Proc. of IEEE INFOCOM*, vol. 1, pp. 345–352, 1999.

[21] C. Barakat, P. Thiran, G. Iannacone, C. Diot, and P. Owezarsky, "Modeling Internet backbone traffic at the flow level," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2111–2123, 2003.

[22] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Rev.*, vol. 51, no. 4, pp. 661–707, Feb. 2009.

[23] P. Tune and D. Veitch, "Towards optimal sampling for flow size estimation," in *Proc. 8th ACM SIGCOMM*, 2008.

[24] K. Ramantas and K. Vlachos, "A TCP prediction scheme for enhancing performance in OBS networks," in *Proc. of IEEE ICC*, 2011, pp. 1–6.

[25] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in *Proc. of IEEE INFOCOM*, 2000, vol. 3, pp. 1742–1751.

[26] J. Teng and G. Rouskas, "A detailed analysis and performance comparison of wavelength reservation schemes for optical burst switched networks," *Photonic Netw. Commun.*, vol. 9, no. 3, pp. 311–335, May 2005.

[27] Y. Xiong, M. Vandenhoute, and H. Cankaya, "Control architecture in optical burst-switched WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 18, pp. 1838–1851, Oct. 2000.

[28] J. Li, C. Qiao, J. Xu, and D. Xu, "Maximizing throughput for optical burst switching networks," *IEEE/ACM Trans. Netw. (TON)*, vol. 15, no. 5, pp. 1163–1176, 2007.

[29] E. Arkin and E. Silverberg, "Scheduling jobs with fixed start and end times," *Discrete Appl. Math.*, vol. 18, no. 1, pp. 1–8, 1987.

[30] U. Faigle, W. Kern, and W. M. Nawijn, "A greedy on-line algorithm for the k-track assignment problem," *J. Algorithms*, vol. 31, no. 1, Apr. 1999.

[31] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. C. Weigle, "Stochastic models for generating synthetic HTTP source traffic," in *Proc. of IEEE INFOCOM*, 2004, vol. 3, pp. 1546–1557.

Author Queries

Journal:	JOCN
Article id:	144773
Author:	Kostas Ramantas and Kyriakos Vlachos
Title:	A TCP-Specific Traffic Profiling and Prediction Scheme for Performance Optimization in OBS Networks

Q1 (Page 1)

The sense of 'one flow round trip time-long prediction window' is not clear. Do you perhaps mean in a protection window that is at least as long as the time needed for a round trip? Please check and amend if necessary.

Q2 (Page 1)

Please define TCP, BTCP, SAIMD, UDP, SYN, ACK, SRAM, DRAM, MSS, and LAUC.

Q3 (Page 1)

Should 'OBS' read 'an OBS network' here and in similar use later? 'networks' has been introduced above, to match the abstract. Please check and amend throughout if necessary.

Q4 (Page 2)

The 'sentence' starting 'Although' is not complete. Please check and amend if necessary.

Q5 (Page 3)

Does SPB always represent (plural) segments, or should this instance read SPBs, with others checked and amended accordingly for singular/plural sense? Please mark any changes to be made clearly.

Q6 (Page 5)

'*' has been replaced by '×' throughtout the file. Pleas check and correct if necessary.

Q7 (Page 6)

The sense of using formulas in an equation is not clear. Please check and amend if necessary.

Q8 (Page 7)

Ratios added here and other rewording done; OK? Please check and amend if necessary.

Q9 (Page 8)

The use of capitals for SETUP does not appear to be consistent. Please check throughout and amend if necessary.

Q10 (Page 11)

Shade has been removed from Table I. Please check and correct if necessary.

Q11 (Page 11)

In order to maintain sequential order, figures have been renumbered. Please check and correct if necessary.

Q12 (Page 13)

Please check the page number and year in Ref. [7].

Q13 (Page 13)

Please provide page number in Ref. [30].