A Novel TCP with Dynamic Burst-Contention Loss Notification over OBS Networks

Basem Shihada[1] and Pin-Han Ho[1,2]
[1]D. R. Chiraton School of Computer Science, University of Waterloo, Waterloo, Canada
[2]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada

Abstract— **In this paper, a novel congestion-control scheme with dynamic Burst-Contention Loss notifications in Optical Burst Switching (OBS) networks is proposed. The proposed scheme, called TCP-BCL, aims to handle various OBS bursty conditions that negatively affect TCP throughput performance and fairness. The basic design principle of the scheme is to tune the congestion-control parameters α and β such that the congestion window sizes in the corresponding TCP senders can be adjusted with an explicit notification from the OBS edge node. The performance impact on TCP in terms of burst dropping due to random contention, which is also known as false congestion detection is considered and investigated. An analytical model is developed and further verified through extensive simulation.**

Index terms: Optical Burst Switching (OBS), false congestion detection, Transport Control Protocol (TCP), Additive Increase and Multiplication Decrease (AIMD)

I. Introduction

Optical Burst Switching (OBS) has been considered as one of the most promising supporting technologies for the next-generation all-optical networks. In OBS networks, data bursts of variable lengths are launched by the OBS edge nodes and asynchronously switched along predetermined physical routes to support the corresponding services. To transport a data burst across OBS network, the corresponding control packet is initiated at the OBS edge node and is sent prior to the data burst in order to configure the switching fabrics for each intermediate node along the route [3,5,6].

Due to its one-way resource reservation mechanism and dynamic characteristics, OBS has been widely considered to provision UDP-based real-time services. However, the Internet is essentially dominated with data traffic, in which the TCP bulk data transfer mechanism with strict QoS requirements in terms of data integrity, reliability, flow and congestion control are of extremely high importance [1,2]. To our best knowledge, only few studies have been reported dedicatedly to the TCP design over the OBS networks [3,5,6,11,12,13], and relatively limited understanding on the TCP behaviours over OBS networks has been gained in the presence of burst dropping at different traffic loads.

A number of TCP implementations have been proposed for detecting and controlling network congestion and segment losses in order to solve some specific problems. In the past years, TCP has been extensively modified and/or extended to cope with the problems incurred by data transmission in a number of different network environments, including mobile wireless networks, ad hoc networks, and some networks with a specific transmission technology in the physical layer such as optical circuit switching (OCS) and OBS. It is clear that each TCP enhancement may have its own design premises, and could be very effective in one circumstance while being much outperformed in others. It has also been proven that jointly considering the characteristics of the whole network environment in the design of the TCP modifications or extensions is necessary. These facts are especially distinguished when TCP is extended to OBS networks, where multiple TCP segments could be assembled in a single burst at an edge, and the burst traverses through the core nodes without any buffering facility.

Generally, TCP takes a segment-loss event as an indication of network congestion. However, if OBS network is lightly loaded, TCP segments could be occasionally dropped due to random burst contention, where two or more bursts try to access the same output port simultaneously. In this case, the corresponding TCP will be informed of network congestion which consequently results in the reduction of congestion window size (*cwnd*) even if the network is under-utilized. This is also referred to as false congestion detection [3], which may significantly impair the TCP throughput.

TCP false congestion detection may impose a fundamental vicious impact on the throughput performance in IP over OBS networks. It could be far from appropriate to simply cut *cwnd* by a half or turn to the slow-start stage in response to a TCP segment loss without regarding the OBS domain congestion status. This problem is getting more serious for those TCP senders of high-bandwidth and long-lasting flows, where the recovery from slow-start could take hours or days according to the currently adopted TCP additive-increase framework. Therefore, an enhanced TCP implementation at the TCP senders for *cwnd* adjustment that is responsive to the underlying OBS characteristics and burst delivery mechanisms is critical to the success of employing the OBS infrastructure in the modern communication carrier networks with mission-critical services such as grid computing.

To our best knowledge, only limited attention has been paid to the performance of TCP over OBS. BTCP [3] is one of the TCP implementations proposed to handle the issue of detecting false timeout, where false reactions to the TCP timeouts are performed in the TCP senders at a burst loss event under various traffic loads. The authors in [3] proposed using TCP segmentation strategies, such that a TCP agent located at the ingress node sends an acknowledgement (*ack*) whenever a TCP segment is received. Also in [3], a more complicated approach has been proposed, where each core node maintains a TCP agent. Whenever, a burst dropping occurs, this agent dissembles the burst and sends back negative *ack*s to the sender. This approach introduces intolerably high complexity at the OBS core nodes. In [7], the authors proposed to persistently retransmit a lost burst. This approach mainly requires the OBS edge node to buffer a copy of the all the bursts within a certain time window until they are successfully delivered, which obviously leads to a huge additional storage requirement at the OBS edges.

The authors in [10] introduced a new TCP scheme called BAIMD for *cwnd* adjustment, where the TCP senders simply consider every burst drop as due to congestion. To mitigate the false identification of network congestion status that may irrelevantly shrink *cwnd* in the TCP senders, the TCP factors for additive increase (denoted as) α and multiplication decrease (denoted as β) are manipulated. This is also referred to as Generalized AIMD (GAIMD) [8], which has been reported to effectively improve the flexibility and efficiency in the TCP implementation in various communication environments. With BAIMD, the summarized effect of a burst drop event due to both contention and congestion is evaluated at each TCP sender by manipulating α and β.

In [11] the authors have solved the false congestion detection problem and avoided unnecessary *cwnd* reduction by introducing a novel TCP implementation, called TCP with Explicit Notification Generalized AIMD (TCP-ENG). TCP-ENG measures the link utilization on each OBS network path and adjusts TCP *cwnd* based on the link utilization and burst dropping information carried in the explicit notification messages between the edge node and the TCP source. This is considered the most straightforward approach for acquiring the congestion status of the path in the OBS network. However, the accuracy of determining the link utilization and the burst dropping probability remains a major challenge. Therefore, TCP-ENG requires additional information to accurately determine the network congestion (*i.e.* long term and short term statistical data of burst losses and round trip time variation).

Based on the amount of asynchronous bursty bandwidth demand at each ingress-egress pair, a new scheme for congestion control should be devised such that decreasing the *cwnd* only responds to the event of true network congestion. The idea has been exercised by [4] in the wireless communication scenarios, where explicit loss notification (ELN) is devised for the state leakage between the lower layer and the TCP layer, similar to the approaches proposed in [3,11]. With ELN, the cause of each burst loss is reported to the TCP sender, which could be due to congestion or other transmission conditions. TCP sender adjusts its *cwnd* according to the network status of the lower layer. To our best knowledge, a comprehensive and systematic approach that can effectively avoid the false congestion detection in IP over OBS networks has never been found.

In this paper, we solve the false congestion detection problem in TCP over OBS networks and avoid the unnecessary *cwnd* reduction by introducing a novel TCP implementation, called TCP with Explicit Notification Generalized AIMD (TCP-BCL). The proposed scheme highly depends on collecting statistical data for measuring the link utilization along each route in the OBS domain and adjusts TCP *cwnd* based on the link utilization and burst dropping information carried in the explicit notification messages. This is considered as the first study that integrates the explicit notification platform with statistical data and the GAIMD framework.

This paper is organized as follows. Section II presents the proposed scheme. Section III introduces an analytical model for the throughput performance of TCP-BCL. In Section IV, simulation is conducted to verify the proposed performance model and compare the proposed scheme with a number of recently reported counterparts, including Reno, SACK, and BAIMD in terms of throughput and fairness. Section V concludes the paper.

II. TCP WITH DYNAMIC BURST-CONTENTION NOTIFICATION

It has been shown that the previously reported TCP implementations are subject to numerous limitations and negatively affected by the OBS bufferless characteristics.
TCP-BCL attempts to improve TCP performance over OBS networks without losing TCP fairness. In terms of the design premise and novelty, the scheme takes the best of BTCP [3] and BAIMD [10], where the GAIMD window-based congestion control paradigm and the mechanisms of explicit notification between the TCP and OBS layers are both used.

A. Congestion Identification in OBS Networks

In traditional packet-switched networks, IP packets are stored and forwarded at each intermediate node by reference to the routing table. In such a switching paradigm, network congestion arises as a result of buffer overflow, and a packet-drop event can serve as a clear indication of network congestion. The situation is different in OBS networks, where each data burst cuts through the pre-configured intermediate core nodes. Therefore, it is important to quantify congestion in such an environment.

In OBS, congestion occurs at ingress and/or egress edge nodes. In general, an ingress node receives incoming packets and forms data bursts. However, packets may arrive in a bursty manner, with a much higher arrival rate than that the ingress node deal with. This could cause the node to drop bursts due to buffer overflow. Similarly, egress nodes may receive a huge number of data bursts, which are buffered for disassembly. We refer to congestion at the network edge node as edge congestion. In addition to edge congestion, path congestion must also be defined, which leads to congestion at network core.

There are two possible approaches for detecting path congestion. The first approach is to delegate the congestion-detection process to the core nodes. The edge nodes receive explicit signals from the core switches indicating link congestion. A similar approach is proposed in BTCP with (BACK/BNACK) [3]. It is notable that this approach may not be very practical, since it increases the signalling and computation overhead at the core nodes. Therefore, in the following paragraphs, we introduce a novel mechanism for detecting congestion along an OBS route/path statistically, at the edge node. This approach does not introduce any additional signalling overhead at the core nodes. Path congestion is measured by how congested the route in the OBS network. This is an important index for the upper-layer TCP senders using the route to adjust their congestion windows.

In our scheme, each OBS ingress edge node maintains the long-term and sort-term statistics along each route initiated. Whenever a burst-drop event occurs, the ingress node determines if the route is in congestion by correlating the long-term and short-term statistics. Specifically, let the parameter *M* be the number of launched bursts along the route used to obtain the long-term statistics, where *M* should be sufficiently large in order to fully represent the intrinsic characteristics of the network topology, routing policy, and traffic pattern, *etc*. The outcome of the *M* burst deliveries is kept as a $1 \times M$ vector with each entry being 0 or 1, representing a burst-drop event or successful delivery, respectively. Let the parameter *N* be the number of launched bursts for evaluating the short-term burst-drop rate, which is generally small; and let the average short-term burst-drop rate of the previous *N* burst deliveries of the current burst-drop event be noted as *avg_b_N*.

The main idea of the proposed scheme is to position the value of *avg_b_N* in the spectrum of long-term burst-drop rates formed by the *M* burst deliveries. To achieve this, the outcomes of the *M* burst deliveries are divided into $\left\lfloor \frac{M}{N} \right\rfloor$ segments each containing the outcomes of consecutive *N* burst deliveries. Thus, a vector θ of a size $1 \times \left\lfloor \frac{M}{N} \right\rfloor$ is obtained, where each entry, θ_i, $i = 1$ to $\left\lfloor \frac{M}{N} \right\rfloor$, keeps the number of burst drops of the *i-th* small segment of the *M* burst deliveries. With the vector, we can obtain the average burst-drop rate for the *M* burst deliveries (*avg_b_M*) and the variance of each entry in the vector θ (*var_b_M*). If *avg_b_N* is larger than *avg_b_M*, it is possible that the route in the OBS core is subject to random burst contention, and the corresponding TCP senders should not take the current burst drop seriously in the adjustment of *cwnd*. On the other hand, with comparable *avg_b_N* and *avg_b_M*, we expect that the current burst-drop event is more likely an indication of congestion along the route. In this case, the corresponding TCP senders should cut their congestion windows in order to relieve the network congestion.

To quantify the relationship between *avg_b_N* and *avg_b_M*, OBS ingress edge firstly derives the spectrum of θ, which is a histogram of θ as shown in Figure 1; then the ingress edge position the *avg_b_N* corresponding to the current burst-drop event in the spectrum in order to identify how likely it is that the route in the OBS domain is congested. The following section explains the approach taken by the edge node to determine the burst loss as congestion or random contention loss.

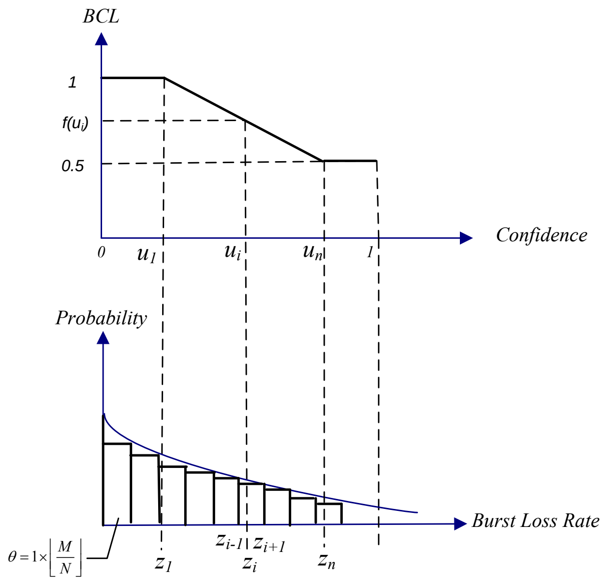

Figure 1. The relation of z_i, u_i, and BCL values in the edge node scheme

B. OBS Edge Node Congestion Detection & Signalling

With *avg_b_N* and the spectrum formed by θ, we define the confidence with which the current burst-loss event is due to network congestion by positioning *avg_b_N* in the spectrum [14]. A function $z_i = BL_{conf}(u_i)$ is defined, where u_i is the confidence level, and z_i is the burst-drop rate corresponding to a confidence level of u_i ($0 < u_i \le 1$) in the spectrum formed by θ. In other words, we have a one-to-one mapping between u_i and z_i as shown in the following expression:

$$u_i = cdf(z_i) = \sum_{j=0}^{i} pmf(z_j) \quad \text{for } 1 \le i \le \left\lfloor \frac{M}{N} \right\rfloor$$

Let the terms $cdf(z_i)$ and $pmf(z_i)$ denote the cumulative density function and probability mass function in the burst losses spectrum given the burst loss value of z_i, respectively. The dynamic determination of network condition based on the confidence level u_i by the edge node is illustrated in Figure 1.

Based on the derived network statistics along the route, the proposed approach for distinguishing event between congestion and random-burst contention is described as follows. Let u_1 and u_n be two pre-defined confidence thresholds. The value of *avg_b_N* is evaluated every time a burst-loss occurs. If *avg_b_N* is smaller than $z_1 = BL_{conf}(u_1)$, a high confidence of random burst contention loss is indicated. In this case, the OBS edge node will send a BCL notification to the corresponding TCP senders with a value of 1. The TCP senders ignore the segment-loss event, keep *cwnd* unchanged, and retransmit the lost segments. On the other hand, when $avg\_b\_N > z_n = BL_{conf}(u_n)$ ($u_n > u_1$), it is taken as a strong indication of network congestion along the route. Hence, the OBS edge node keeps quiet. Therefore, the TCP senders cut their congestion windows by half in response to the current segment-drop event. When *avg_b_N* falls in the interval $[z_1, z_n]$, the BCL is set to $f(u_i)$ and is sent back to the TCP senders, where $f(u_i) = 1 - \frac{u_i - u_1}{2(u_n - u_1)}$. TCP senders will set their β values equal to $f(u_i)$ [14].

Note that u_1 and u_n are two parameters given in advance in order to distinguish the route status between congestion and random contention. In this study, u_1 and u_n are set to 50% and 90%, respectively. However, the values of u_1 and u_n can be set by the network administrators based on the traffic and network engineering characteristics. The policy-based adjustment scheme can be summarized in the following equation,

$$\beta = BCL = \begin{cases} f(u_i) & BL_{conf}(u_n) > avg\_b\_N > BL_{conf}(u_1) \\ 1 & BL_{conf}(u_1) \geq avg\_b\_N \end{cases}$$

The adjustment of BCL values based on the confidence level u_i is illustrated in Figure 1. The flowchart for the proposed TCP-BCL scheme is shown in Figure 2. The two parameters *M* and *N* are custom-designed, and in this study, *M* and *N* are chosen to be 500 and 10, respectively. In other words, the current network congestion status is determined by the position of the previous 10 burst deliveries in the spectrum formed by the previous 500 burst deliveries. Finding optimal values of *N* and *M* is an open problem that is left for future research. The scheme is expected to be stable since any increase of *avg_b_N* will be positively fed back by shifting the spectrum formed by θ farther to the right.

In order to enable explicit burst contention notification, an agent is placed at each OBS ingress edge node, and is responsible for determining network status (*i.e.*, link utilization) when a burst-drop event is encountered. The agent sends BCL notifications to the corresponding TCP senders if the burst-drop event is judged due to contention. In other words, TCP-BCL has the TCP senders take every segment-loss event as due to congestion when a BCL is not received. This distinguishes our scheme from BTCP in [3] since TCP-BCL only signals the dropped bursts at low link utilization, while with BTCP, the TCP agent reports every burst-drop event. Hence, the number of notifications in TCP-BCL is less than in BTCP. Furthermore, our design can significantly reduce the intra-domain signalling and core node processing overhead.

Another important feature of the proposed scheme is that it is designed for those TCP flows with high-bandwidth and long-lasting, which serve as a building block in many emerging networking scenarios such as grid computing applications. It is clear that the performance of such TCP flows is very sensitive to the false congestion identification since it could take hours for such a flow to return to the original rate with the current additive increase framework. Our scheme solves the false congestion identification problem with reasonable additional overhead incurred at the OBS edge nodes.

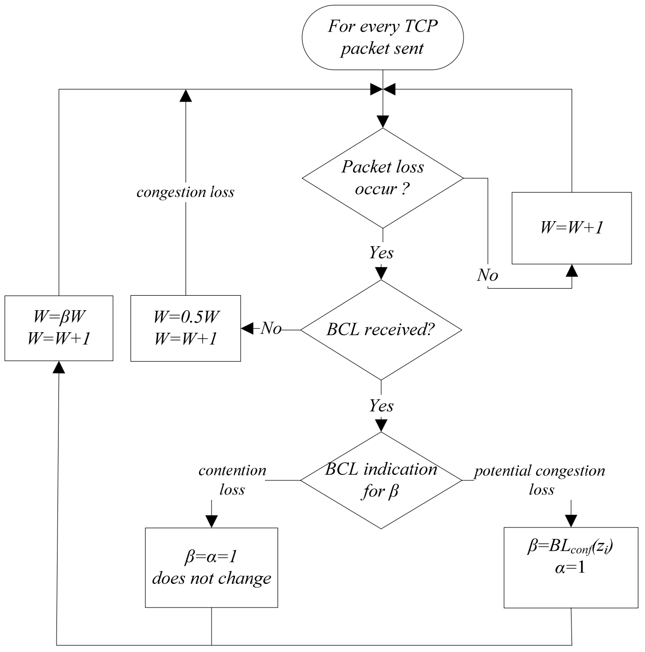

Figure 2. Flowchart for TCP-BCL congestion control scheme

III. Performance Modeling

In this section, we analyze the throughput performance of the TCP-BCL flows in an OBS network. We have selected TCP SACK for our analytical model since it has been widely deployed in the current operating systems and has shown the best throughput performance over OBS networks compared to TCP Reno and New Reno [3]. The following table lists the notations used in the analytical model.

p	:	burst loss probability
b	:	number of packets that are acknowledged by receiving an *ack*
B	:	TCP throughput
H	:	number of packets transmitted during TO
$\overline{RTT}$	:	average round trip time
TOP	:	timeout period
TDP	:	triple duplicate period
RTO	:	retransmission timeout
Z^{TO}	:	duration of a sequence of TOs
X	:	number of successful rounds in a TDP
Y	:	number of packets sent before TD or TO expiration
W	:	current congestion window size in segments
W_m		TCP maximum window size
S	:	number of segments belonging to a single TCP flow being assembled in the current burst
Q	:	ratio between the probability of TO loss and TD loss

In our model, we define a round as when the TCP sender emits the current *cwnd* (in segments) and waits until either it receives an *ack* or the timeout (TO) expires. We also define a TD loss as a packet loss detected by triple duplicates and define a TO loss as a packet loss detected after TCP sender timeouts.

We obtain the TCP-BCL SACK throughput in OBS network for both TD and TO losses as follows:

$$B = \frac{E[Y] + Q \times E[H]}{E[TDP] + Q \times E[TOP]} \quad (1)$$

It is worth to recall that TCP-BCL uses the GAIMD congestion control mechanism. The relation between α and β for ensuring friendliness among other existed TCP flows in the case of triple-duplicate *acks* is obtained from [8] as:

$$\alpha = \frac{3(1-\beta)}{1+\beta} \quad (2)$$

while in the case of timeout,

$$\alpha = \frac{4}{3}(1-\beta^2) \quad (3)$$

In the following two sections, we will derive $E[Y]$, $E[TDP]$, $E[TOP]$, $E[H]$, and Q in the presents of TD and TO losses respectively.

A. TCP-BCL in Triple Duplicate (TD) Losses

As per the model in [12], suppose that the (c_i+1)th burst is the first burst lost in the ith TDP, TDP_i, which contains the first (a_i+1)th lost segment in the TDP_i. As shown in Figure 3, h_i additional segments will be sent in the same round after the (c_i+1)th burst is sent and lost. TCP sender retransmits all the missing segments contained in the lost burst in the next round. Therefore, in the next round, $W_{X_i} - S$ new segments will be sent, where W_{X_i} is the *cwnd* size in the X_i th round in the TDP_i. After recovering all the segments lost in the burst and the receiving *BCL*, a new round TDP_{i+1} starts with the *cwnd* being cut by a factor of β. The sending rate α is determined according to Eq. (2) and Eq. (3). The total number of segments successfully transmitted during the TDP_i is $Y_i = a_i + h_i + W_{X_i} - S$. $E[h]$ is approximately equal to $\beta E[W_X]$, since $0 \le h_i \le W_{X_i}$ and the *cwnd* is reduced by β for every TD loss. Thus, we have

$$E[Y] = E[u] + (\beta+1)E[W_X] - S \quad (4)$$

In order to derive $E[a]$, we consider a random process $\{c_i\}$, which is the average number of bursts sent in the TDP_i till the first burst loss. Assume that burst contentions in OBS networks occur independently. The probability of $c = k$ (or the case where k–1 bursts are successfully delivered before a burst loss is encountered) can be written as:

$$P[c = k] = (1-p)^{k-1} \cdot p \quad (5)$$

Given that $a_i = Sc_i$ we have,

$$E[a] = S\,E[c] = S\sum_{k=1}^{\infty} k(1-p)^{k-1}p = \frac{S}{p} \quad (6)$$

By substituting Eq. (6) into Eq. (4), we have

$$E[Y] = (\beta+1)E[W_X] + \frac{1-p}{p}S \quad (7)$$

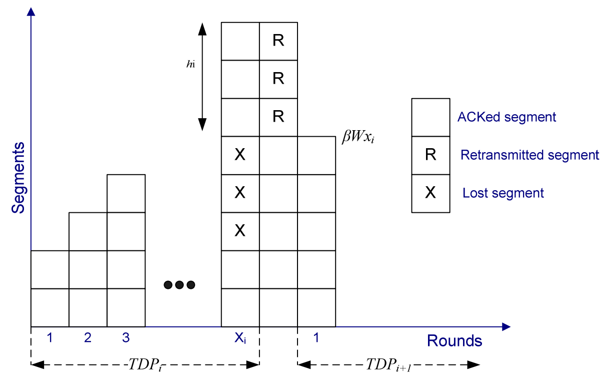

Figure 3. Evolution of TCP-BCL SACK congestion window over OBS networks

1) For high packet losses ($W_X < W_m$)

In the presence of a high packet loss probability, the *cwnd* will remain less than the maximum size W_m. Recall that b denotes the number of packets that are acknowledged by receiving an *ack*. During the TDP_i, the *cwnd* increases between $\beta W_{X_{i-1}}$ and W_{X_i}. Since the increase of the *cwnd* is linear with slop α/b,

$$W_{X_i} = \beta W_{X_{i-1}} + \frac{\alpha X_i}{b} \quad (8)$$

By solving for X_i, we have

$$E[X] = \frac{b(1-\beta)E[W_X]}{\alpha} \quad (9)$$

Since Y_i can be derived by summing the number of segments sent in X_i successful rounds and the additional ($W_{X_i} - S$) segments in the next round of X_i as shown in Figure 3, we have:

$$Y_i = \sum_{k=0}^{X_i/b-1}(\beta W_{X_{i-1}} + k)\frac{b}{\alpha} + W_{X_i} - S$$
$$= \frac{X_i}{2}(2\beta W_{X_{i-1}} + \frac{\alpha X_i}{b} - 1) + W_{X_i} - S$$

By substituting Eq. (8), we have

$$Y_i = \frac{X_i}{2}(\beta W_{X_{i-1}} + W_{Xi} - 1) + W_{X_i} - S$$

after substituting Eq. (9), we get

$$E[Y] = \frac{b(1-\beta^2)E[W_X]^2 - b(1-\beta)E[W_X]}{2\alpha} + E[W_X] - S \quad (10)$$

By combining Eq. (10) and Eq. (7), we have,

$$\frac{b(1-\beta^2)}{2\alpha}E[W_X]^2 + (\frac{b\beta - b}{2\alpha} - \beta)E[W_X] - \frac{S}{p} = 0$$

$E[W_X]$ can be then obtained as

$$E[W_X] = \frac{\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{b(1-\beta^2)} \quad (11)$$

By substituting Eq. (11) into Eq. (7), we obtain $E[Y]$ as,

$$E[Y] = \frac{\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{b(1-\beta)} + \frac{1-p}{p}S \quad (12)$$

Also, by substituting Eq. (11) into Eq. (9), we obtain $E[X]$ as,

$$E[X] = \frac{\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{1+\beta}$$

(13)

$E[TDP]$ is then obtained as

$$\begin{aligned} E[TDP] &= \overline{RTT}(E[X]+1) \\ &= \overline{RTT}(\frac{2\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{1+\beta} + 1) \end{aligned} \quad (14)$$

2) *For low burst losses ($W_X = W_m$)*

For a very low burst loss probability, the *cwnd* size will most likely remain at the maximum *cwnd* size, W_m, before a burst loss event occurs. From Eq. (7) we can obtain,

$$E[Y] = (\beta+1)W_m + \frac{1-p}{p}S \quad (15)$$

During each TDP, *cwnd* increases linearly from βW_m to W_m for $(W_m - \beta W_m)$ rounds and then stays at W_m for $(\alpha X_i - (W_m - \beta W_m))$ rounds, hence we can obtain the number of segments that are transmitted before a TD loss as $\frac{(W_m - \beta W_m)^2}{2} + W_m(\alpha X_i - W_m + \beta W_m) - h_i$. On the other hand, from Eq. (6), the total number of segments that are transmitted successfully before a packet loss is $\alpha S/p$. Hence we have

$$\frac{(W_m - \beta W_m)^2}{2} + W_m(\alpha X_i - W_m + \beta W_m) - h_i = \frac{S}{p} \quad (16)$$

By reversing Eq. (16), we can obtain $E[X]$ as

$$E[X] = \frac{W_m(1-\beta)}{\alpha} - \frac{W_m(1-2\beta+\beta^2)}{\alpha 2} + \frac{\beta}{\alpha} + \frac{S}{\alpha W_m p} \quad (17)$$

The duration of the *TDP* is obtained as,

$$\begin{aligned} E[TDP] &= \overline{RTT}(E[X]+1) \\ &= \overline{RTT}(\frac{W_m(1-\beta)}{\alpha} - \frac{W_m(1-2\beta+\beta^2)}{\alpha 2} + \frac{\beta}{\alpha} + \frac{S}{\alpha W_m p} + 1) \end{aligned} \quad (18)$$

B. Timeout (TO) Losses

The behavior of TCP-BCL for a TO loss is same as that of TCP SACK. Hence, the analysis of TO losses is same as the analysis in [12]. TCP-BCL transmits the same number of packets between two TOs as compared with traditional TCP. However, for TCP-BCL, the packet retransmission starts as soon as the *BCL* is received, which is the RTT after the loss round. Thus, TCP-BCL waits for *TOP= RTT/p*. From [12], we have:

$$E[H] = E[R] - 1 = \frac{p}{1-p}, \quad (19)$$

$$E[TOP] = \frac{RTT(f(p))}{p(1-p)}, \quad (20)$$

where $f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6$, and

$$Q(E[W_X]) \approx p^{\frac{W_X}{S}-1}. \quad (21)$$

C. TCP-BCL SACK over OBS Throughput Estimation

In the case of $W_X < W_m$, we can obtain the TCP-BCL throughput by substituting Eqs. (12), (14), (19), (20), and (21) into Eq. (1), which yields

$$B = \frac{\frac{\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{b(1-\beta)} + \frac{1-p}{p}S + \frac{p^{\frac{W_X}{S}}}{(1-p)}}{\overline{RTT}(\frac{2\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{1+\beta} + 1) + p^{\frac{W_X}{S}-1}\frac{RTT(f(p))}{p(1-p)})}$$

(22)

In the case of $W_X = W_m$, TCP-BCL throughput can be obtained by substituting Eqs. (15), (18), (19), (20), and (21) into Eq. (1), which yields

$$B = \frac{(\beta+1)W_m + \frac{(1-p)S}{p} + \frac{p^{\frac{W_m}{S}+1}}{1-p}}{\overline{RTT}(\frac{W_m(1-\beta)}{\alpha} - \frac{W_m(1-2\beta+\beta^2)}{2\alpha} + \frac{\beta}{\alpha} + \frac{S}{\alpha W_m p} + 1) + p^{\frac{W_m}{S}-1}\frac{RTT(f(p))}{p(1-p)})}$$

(23)

IV. NUMERICAL RESULTS

In this section, we evaluate the throughput performance of the proposed TCP-BCL implementation in IP over OBS environment. We compare the TCP-BCL throughout with the BTCP and BAIMD as well as the TCPs that were originally designed for general packet-switched networks, including TCP Reno and SACK. The network simulator *NS-2* is used and the NSF network topology is implemented as shown in Figure 4. The distances shown are in *km*.

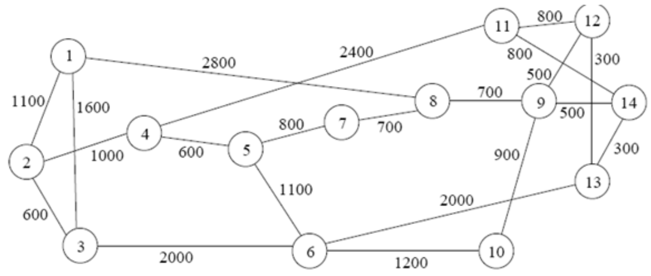

Figure 4. NSF OBS simulation topology

The bursts are generated at the ingress edges traverse through nodes with a minimum of four hops before reaching their destinations. Depending on the number of competing TCP flows, the burst dropping probability varies between 10^{-5} and 10^{-1}. At each ingress edge, the mixed time/length based burst assembly algorithm is used, where the burst timeout threshold is set to 100 *ms* and the maximum burst length is 50*KB*. The core nodes implement the *LAUC-VF* channel scheduling algorithm. One fiber link consist of 8 wavelengths operating at 10Gbps transmission rate is used for data burst transfer between two adjacent nodes with 10*ms* propagation delay. One bi-directional control channel is allocated along each link.

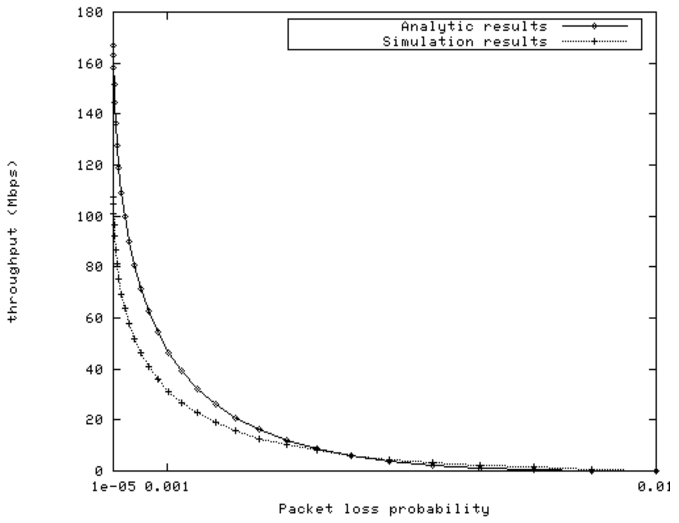

Figure 5. TCP-BCL throughput over OBS networks.

Control packet processing time is set to 1μs at both core and edge nodes. The burst offset time is set to 6μs, which is sufficient for bursts to traverse through a minimum of 4 hops. A File Transfer Protocol (FTP) application is used for generating TCP traffic with a 1*KB* average packet size. The TCP throughput is obtained over a simulation period of 10^3 seconds. We examine the TCP throughput over barebone OBS and OBS with burst retransmission. Note that burst retransmission has been identified as an effective approach for enhancing the overall network throughput by recovering some contended bursts in the OBS layer while keeping the upper TCP layer unaware [7]. If a burst has a contention in the second retransmission attempt, the burst will be dropped.

A. TCP-BCL Numerical Results

Figure 5 compares the analytical results obtained by Eq. (22) and Eq. (23) and the simulation results obtained for TCP-BCL fast flows [12]. It is notable that the simulation results match the analytical results. Figure 6 shows the throughput performance by TCP-BCL, TCP Reno and SACK under barebone OBS and OBS with burst retransmission. It is clear that TCP-BCL significantly outperforms the two conventional TCP schemes.

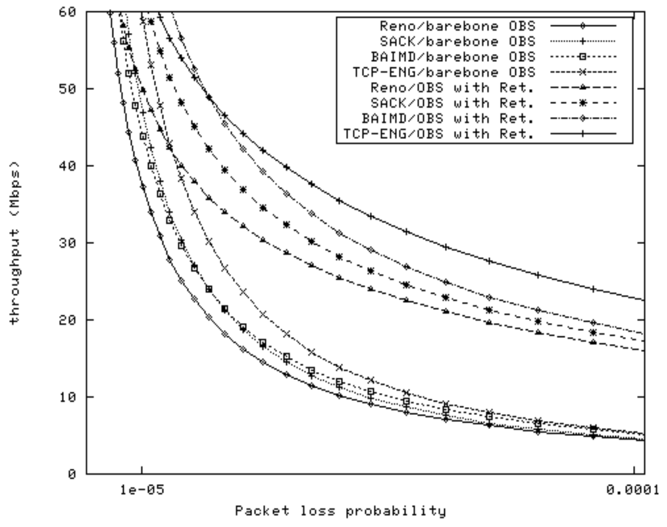

Figure 6. TCP Reno, SACK vs. TCP-BCL throughput over barebone OBS and OBS with burst retransmission

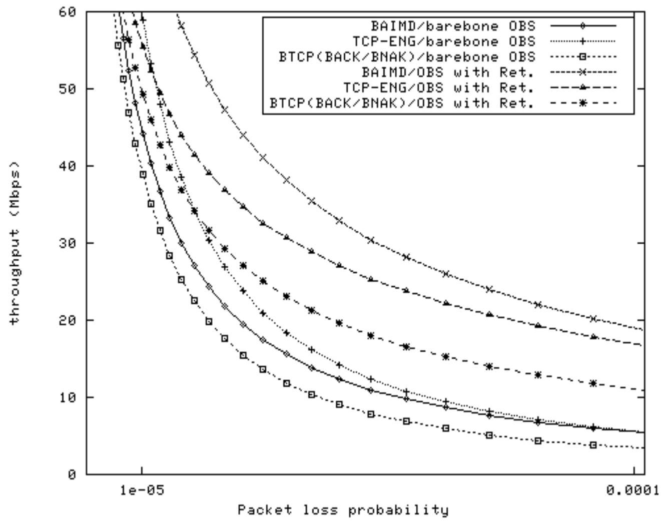

Figure 7. BTCP, BAIMD, and TCP-BCL throughput over barebone OBS and OBS with burst retransmission.

In Figure 7, TCP-BCL is compared with BAIMD and BTCP. We observe that TCP-BCL still achieves higher throughput than the other two by combining the best features of BTCP and BAIMD. The explicit notifications make the TCP-BCL senders be informed of a specific OBS domain channel status, where the two control parameters can be adjusted with the best relevance in response to a burst dropping event.

Through the above simulation studies, we verified the proposed TCP-BCL scheme and concluded that TCP-BCL can solidly solve the false congestion detection problem and achieve better throughput than that by all the other TCP implementations based on AIMD mechanism in the carrier with OBS transmission. The overhead incurred in the proposed scheme is the explicit notification which is required when a burst loss is determined as due to random contention. With the aid of explicit notifications and allocation of the TCP agent at the OBS edges, the TCP-BCL scheme can solidly outperform BAIMD and BTCP. In addition, we observed that the proposed scheme can effectively maintain awareness of non-congestion burst drops by manipulating the α and β values in the TCP senders, which can right serve as a candidate in the TCP implementation for the future Internet with OBS taken as the underlying infrastructure technology.

B. TCP-BCL Fairness

In this section, the fairness among TCP-BCL, BAIMD, SACK, and Reno is examined. For this purpose, we use Jain's fairness index. If all flows have the same throughput, then the fairness index is 1 [8]. The competing flows share the same source-destination pairs.

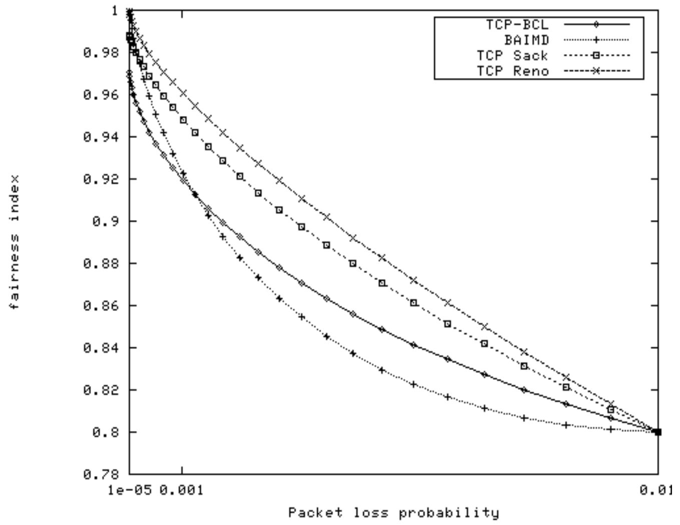

Figure 8. Fairness index of TCP-BCL, BAIMD, SACK, and Reno in barebone OBS

Figure 8 shows the fairness index of flows by TCP-BCL, BAIMD, and TCP Reno. We can see TCP-BCL has a much better fairness index than BAIMD, TCP Reno, and SACK. This is due to the fact that the congestion control mechanism taken by the BAIMD, TCP Reno, and SACK does not relay on the explicit signalling between the TCP senders and the OBS edge nodes, which causes underestimation of network congestion. Thus, it results in selecting a larger multiplicative values β while keeping the additive value α at 1.

The fairness index has also been examined in Figure 9 while enabling the burst retransmission mechanism. The simulation results have shown that the throughput of TCP-BCL is still close to SACK and Reno, which verifies that TCP-BCL maintains a friendly relation with the other TCPs.

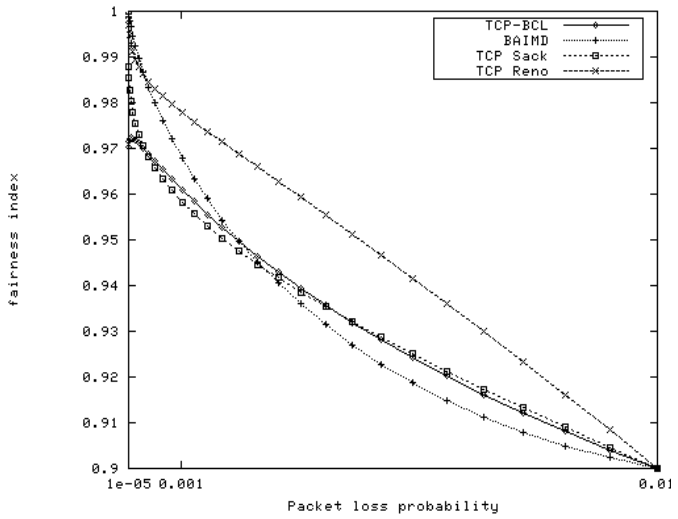

Figure 9. Fairness index of TCP-BCL, BAIMD, SACK, and Reno in OBS with burst retransmission

V. Conclusions

In this paper, we proposed a new TCP implementation in IP over OBS networks, called TCP with Explicit Burst-Contention Loss Notification (TCP-BCL), which aims to take the best of two recently proposed TCP enhancements: BAIMD and the BTCP. Through the simulation and the analytical modeling we demonstrated that the proposed TCP-BCL scheme can countermeasure the negative effect of the OBS bufferless characteristic and is expected to serve as a better candidate in the bufferless OBS networks compared with the conventional AIMD architecture. We have shown the superiority of the proposed scheme in terms of throughput, link utilization, and fairness.

Acknowledgement

This study is sponsored by Bell University Laboratory (BUL), Bell Canada.

References

[1] W. Stevens, "TCP/IP Illustrated, Volume1, Addison Wesley Longman, 1994.

[2] W. Noureddine and F. Tobagi, "The transmission control protocol, an introduction to TCP and a research survey", Technical Report, July 2002.

[3] X. Yu, C. Qiao, and Y. Liu, "TCP implementation and false time out detection in OBS networks," *Proceedings of IEEE Infocom*, pp. 774-784, 2004.

[4] H. Balakrishanan, R. Katz, "Explicit loss notification and wireless web performance," *Proceedings of IEEE Globecom, Internet MiniConference,* 1998.

[5] Y. Chen, C. Qiao, and X. Yu, "An optical burst switching: a new area in optical networking research," *IEEE Network*, vol. 18, no. 5, pp. 16-23, 2004.

[6] X. Cao, J. Li, Y. Chen, and C. Qiao, "Assembling TCP/IP packets in optical burst switched networks", *Proceedings of IEEE Globecom*, 2002.

[7] Q. Zhang, V. Vokkarane, Y. Wang, and J. Jue, "TCP over optical burst-switched networks with optical burst retransmission," *IEEE Journal on Selected Areas in Communications – Optical Communication and Networking Series*, 2005.

[8] L. Cai, X. Shen, J. Pan, and J. W. Mark, "Performance analysis of TCP-friendly AIMD algorithms for multimedia applications", *IEEE Transaction on Multimedia*, vol. 7, no. 2, pp. 339-355, 2005.

[9] S. Bhandarkar and N. Reddy, "Improving the robustness of TCP to non-congestion events", internet draft draft-ietf-tcpm-tcp-dcr-01.txt, 2004

[10] B. Shihada, P-H. Ho, F. Hou, and et al,"BAIMD: a responsive rate control for TCP over optical burst switched (OBS) networks," *Proceeding of IEEE (ICC)*, Istanbul, Turkey, 2006.

[11] B. Shihada, P-H. Ho, and Q. Zhang, "TCP-ENG: Dynamic Explicit Congestion Notification for TCP over OBS Networks", 16th IEEE International Conference on Computer Communications and Networks (ICCCN 07), Hawaii, USA, August, 2007.

[12] A. Detti and M. Listanti," Impact of segment aggregation on TCP Reno flows in optical burst switching networks, " in the proceedings of IEEE Infocom, 2002.

[13] X. Yu, C. Qiao, Y. Liu, and D. Towsley, "Performance evaluation of TCP implementations in OBS networks," Technical Report, 2003-13, the State University of New York at Buffalo, 2003.

[14] J. A. Gubner, "Probability and random processes for electrical and computer engineers", Cambridge University Press, pp 240-262, 2006.