

A Novel Implementation of TCP Vegas for Optical Burst Switched Networks

Basem Shihada¹, Qiong Zhang², Pin-Han Ho¹, and Jason P. Jue³

¹Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada

²Fujitsu Laboratories of America, Inc., Richardson, TX, USA

³Department of Computer Science, University of Texas at Dallas, USA

Abstract – TCP performance over bufferless Optical Burst Switched (OBS) networks could be significantly degraded due to the misinterpretation of network congestion status (referred to as *false congestion detection*). It has been reported that burst retransmission in the OBS domain can improve the TCP throughput by hiding burst loss events from the upper TCP layer, which can effectively reduce the congestion window fluctuation at the expense of introducing additional delay. However, the additional delay may cause performance degradation for delay-based TCP implementations that are sensitive to packet round trip time in estimating the network congestion status. In this paper, a novel implementation of TCP Vegas that adopts a threshold-based mechanism is proposed for identifying the network congestion status in OBS networks. Analytical models are developed to evaluate the throughput of conventional TCP Vegas and threshold-based Vegas over OBS networks with burst retransmission. Simulation is conducted to validate the analytical model and to compare threshold-based Vegas with a number of legacy TCP implementations, such as TCP Sack and TCP Reno. The analytical model can be used to obtain a proper threshold value that results in an optimal steady state TCP throughput.

Index terms: Optical Burst Switching (OBS), Transport Control Protocol (TCP), TCP Vegas.

I INTRODUCTION

TCP has been subject to a tremendous amount of research effort over the past decade. A large number of reported studies have suggested modifying TCP for adapting to new network scenarios with different transmission characteristics [1-8]. Most of these modifications have been developed based on best-effort packet delivery and buffer-oriented routing mechanisms and are typically placed into three categories: loss-based TCP (such as TCP Reno [5] and TCP Sack [6]), delay-based TCP (such as TCP Vegas [7] and Fast TCP [2, 3]), and explicit notification-based TCP (such as XCP [8]). It is notable that all of these schemes are based on the assumption that the network congestion status can be effectively indicated by way of either packet loss, elongated round trip time (RTT), or the combination of both. However, this assumption is only true in buffer-oriented networks.

Optical Burst Switching (OBS) is a switching technique for optical networks that has been shown to achieve efficient and dynamic bandwidth allocation for handling the bursty and dynamic nature of the Internet traffic [9, 10]. OBS networks are also suitable for many emerging applications, such as Grid and content distribution networks [11, 12, 13]. An OBS network is bufferless in nature [9], and each data burst cuts

through the OBS domain following the corresponding control packet by way of a one-way signaling resource reservation protocol. Thus, a burst loss in the OBS domain may be caused not only by high utilization of network resources for a relatively long period (called *congestion*), but also due to *random contention*, which occurs even when the network resource utilization is quite low in a given time window. Thus, a burst loss due to random contention in the OBS domain may cause the TCP senders to react improperly in response to the resultant TCP packet loss events, which is referred to as *TCP false congestion detection* [14], and which could significantly downgrade the TCP throughput in OBS networks. This problem becomes worse when a burst contains packets from numerous TCP connections and each of the connections is regulated by individual flow control and congestion control mechanisms.

To cope with the TCP false congestion detection problem, explicit signaling from the OBS layer to the TCP layer is proposed in [14]. The limitation of this approach is that generating an explicit signal for every random contention in the OBS layer will significantly increase the network operation overhead and reduce the protocol practicality by breaking the TCP end-to-end semantics. Other approaches to solving the problem include burst retransmission and deflection schemes at the OBS layer [15,16,17]. It has been shown that these schemes can hide some burst loss events from the upper TCP layer, thereby reducing the chance of TCP false congestion detection. With burst retransmission or deflection, bursts subject to contention can be retransmitted at an OBS edge or can be deflected to an alternate route in the OBS network, respectively. Hence, the probability that a burst experiences random contention (referred to as *burst contention probability*) could be much larger than the probability that a burst is actually dropped and sensed by the TCP layer (referred to as *burst loss probability*), and the fluctuation of the TCP congestion window can be significantly suppressed.

In spite of the merits gained by employing burst retransmission, additional delay is introduced and could have a negative impact on delay-based TCP implementations such as TCP Vegas [7] and Fast TCP [2, 3]. In this paper, we propose a novel scheme, threshold-based TCP Vegas, for achieving more precise network congestion identification in OBS networks under burst retransmission. The proposed scheme adjusts the *cwnd* size based on the round trip times (RTTs) of packets received at TCP senders. If the number of RTTs that

are longer than the minimum RTTs exceeds a threshold, the scheme detects network congestion, and thus adjusts the $cwnd$ size similar to conventional TCP Vegas; otherwise, the scheme adjusts the $cwnd$ size based on the minimum RTTs. To verify the effectiveness of the proposed scheme, simulation is conducted to compare the proposed scheme with a number of conventional TCP implementations, such as TCP Sack and Reno. We show that a significant improvement is observed in terms of TCP throughput for the threshold-based TCP implementation over OBS networks with burst retransmission. Analytical models for the throughput performance of both the conventional TCP Vegas scheme and the proposed threshold-based TCP Vegas scheme are formulated, and the models are validated through extensive simulation.

The rest of the paper is organized as follows. Section II describes conventional TCP Vegas as background for the research. Section III presents the proposed threshold-based TCP Vegas. Section IV introduces the proposed analytical model for threshold-based TCP Vegas over OBS networks with burst retransmission. Section V presents the simulation results. Section IV concludes the paper.

II TCP VEGAS

TCP Vegas [7, 18, 19] measures the RTT of each packet transmission to estimate available bandwidth and congestion status in the network. TCP Vegas differs from TCP Reno in the congestion avoidance, slow-start, and retransmission phases. The modifications of TCP Vegas over TCP Reno are summarized in this section.

A TCP Vegas Congestion Avoidance

TCP Reno takes each packet loss as an indicator of network congestion and cannot detect any potential congestion before a packet loss occurs. On the other hand, TCP Vegas compares the estimated and the measured throughput in a specific time window to determine the congestion status in the network. TCP Vegas first computes $BaseRTT$ as the minimum measured RTT, which is primarily determined by the propagation delay and the queuing delay. The $Expected$ throughput can thus be derived according to the following equation:

$$Expected = \frac{cwnd}{BaseRTT}, \quad (1)$$

where $cwnd$ is the current congestion window size.

Second, $Actual$ throughput is calculated for every RTT time using the most recent measured RTT by

$$Actual = \frac{cwnd}{RTT}. \quad (2)$$

Vegas then compares $Actual$ and $Expected$ and computes the difference between the two quantities (denoted as $Diff$):

$$\begin{aligned} Diff &= Expected - Actual \\ &= \left(\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT} \right) BaseRTT \\ &= cwnd \left(1 - \frac{BaseRTT}{RTT} \right). \end{aligned} \quad (3)$$

$Diff$ is obviously non-negative, and is used to adjust the next

$cwnd$. Vegas defines two threshold values, denoted as α and β , for controlling the congestion window size, $cwnd$. The congestion window is changed as follows:

$$cwnd = \begin{cases} cwnd + 1 & diff < \alpha \\ cwnd & \alpha \leq diff \leq \beta. \\ cwnd - 1 & diff > \beta \end{cases} \quad (4)$$

If $Diff < \alpha$, Vegas increases $cwnd$ linearly in the next RTT. If $Diff > \beta$, Vegas decreases $cwnd$ linearly in the next RTT. Otherwise, Vegas leaves $cwnd$ unchanged.

The TCP Vegas congestion avoidance mechanism aims to maintain the expected number of on-fly backlog packets in the network between α and β . If the *Actual* throughput is much smaller than the *Expected* throughput, it is likely that the network is congested. Thus, the TCP sender reduces the flow rate. On the other hand, if the *Actual* throughput is close to the *Expected* throughput, the connection may not be utilizing the available flow rate, and thus the flow rate should be increased.

B TCP Vegas Slow-Start

In the slow-start phase of TCP Reno, the $cwnd$ is increased by one for each successfully acknowledged packet, effectively doubling $cwnd$ each RTT. The exponential growth of $cwnd$ continues either until it exceeds the receiver's advertised $cwnd$ or the initial threshold, or until a packet loss is reported. If the initial window threshold value is small, the exponential increase will stop early, which leads to low throughput. On the other hand, if the threshold is set too large, the $cwnd$ will grow such that the available bandwidth in the network is totally exhausted, which results in more packet losses.

TCP Vegas increases the $cwnd$ exponentially only every other RTT during the slow-start phase, and exits slow-start if the $cwnd$ reaches the slow-start threshold. During two consecutive RTTs, the $cwnd$ stays fixed in order to make a valid comparison between the *Expected* and *Actual* throughput.

C TCP Vegas Packet Retransmission

TCP Reno sets $cwnd$ to one packet and enters the slow-start phase when a timeout (TO) is detected. The TCP Vegas retransmission mechanism improves on TCP Reno by reducing the occurrence of TOs and reducing the time taken for fast retransmission.

When a TCP Vegas sender receives an acknowledgement (ACK), it records the clock and calculates the estimated RTT using the current time and the timestamp recorded for the associated packet. Vegas then decides whether to retransmit the packet based on the following two conditions. First, when a duplicate ACK is received, Vegas checks if the difference between the current time and the timestamp recorded for the associated packet is greater than the timeout value. If true, the Vegas sender retransmits the packet without having to wait for the remaining incoming duplicate ACKs. Second, when an ACK is received, if it is the first or the second ACK after a retransmission, Vegas again checks if the elapsed time since the packet was sent is larger than the timeout value. If it is,

Vegas retransmits the packet. This will catch any other packet that may have been lost prior to the retransmission without having to wait for a duplicate ACK. Hence, the Vegas retransmission mechanism reduces the time in detecting a lost packet from the third duplicate ACK to the first or second duplicate ACK. After the packet retransmission is triggered by a duplicate ACK, the congestion window size is reduced by 25% (instead of 50% in Reno) only if the time since the last reduction of $cwnd$ is more than the current RTT.

III THRESHOLD-BASED TCP VEGAS OVER OBS

There are several issues observed when the conventional TCP Vegas congestion control mechanism is adopted in OBS networks. With a given physical route for the source and destination OBS edges, the delay experienced in the OBS domain is primarily the sum of burst assembly delay and link propagation delay, which do not vary with the network traffic load. Hence, conventional TCP Vegas cannot effectively detect network congestion in the OBS domain. Furthermore, in the event that all the packets in a single congestion window are assembled into a single burst, the TCP Vegas sender may suffer from the false congestion detection problem if the burst is lost in the OBS domain due to random contention, which will adversely impair the TCP throughput by having the TCP sender unnecessarily performing TO retransmission and then entering slow-start.

When TCP Vegas runs over OBS networks with the burst retransmission scheme, the false congestion detection problem can be significantly mitigated. However, although able to hide a burst loss event from the TCP layer, the employment of burst retransmission in the OBS domain will certainly incur extra delay, which leads to an increase of RTTs for the packets in the retransmitted bursts as perceived by the TCP sender. The extra delay will be interpreted as some extent of network congestion in spite of a possibly lightly-loaded OBS domain in reality, causing TCP Vegas to unnecessarily decrease its $cwnd$ and resulting in a lower TCP throughput. Hence, we are motivated to further enhance conventional TCP Vegas, such that the TCP senders can tell whether the increase in RTTs is due to network congestion, or due to retransmission in a lightly-loaded OBS network. Similar situation applies to other delay-based TCP versions, such as Fast TCP [2].

We observe that if the IP access network is congested, TCP Vegas will continuously detect the increase in RTTs due to queuing delay and packet loss. In the case that the OBS network is heavily loaded, the burst contention events occur frequently. Thus, TCP Vegas will often detect the RTT increase. If both the IP access network and the OBS network are not congested, random burst contentions would occur less frequently. Hence, the TCP Vegas senders can detect network congestion (in both OBS networks and IP access networks) if RTTs are frequently increased.

Based on the above observations, we propose a threshold-based TCP Vegas scheme to assist TCP Vegas in distinguishing between network congestion and burst contention at low traffic loads. A new parameter, T , is devised, which is defined as the threshold on the number of packets

with longer RTT than the minimum measured RTT in the previous N rounds. The parameter is used to evaluate the extra delay caused by burst retransmission, and can be adaptively manipulated in order to mitigate the negative impact due to false congestion identification.

More specifically, TCP Vegas measures RTT for each launched packet and keeps track of the minimum measured $RTTs$ of previous N consecutive packets. Let $MinRTT(i)$ be the minimum measured $RTTs$ of i ($0 < i < N$) consecutive packets. In the i th round, if the measured RTT of the i th packet is larger than $MinRTT(i-1)$, it means that the i th packet was once queued in the access network or assembled in a burst that was retransmitted. A counter that keeps the number of packets whose $RTTs$ are larger than their $MinRTT(i-1)$ will then be increased by one. If the number of TCP packets whose $RTTs$ are larger than their $MinRTT(i-1)$ is under the threshold T , the TCP sender will stay with the *Actual* throughput calculated based on $MinRTT(i-1)$, even if the measured $RTTs$ are increased. If the number of TCP packets whose $RTTs$ are larger than their $MinRTT(i-1)$ exceeds the threshold T , it means that the network is congested. Hence, threshold-based Vegas recognizes the network congestion and calculates the *Actual* throughput as usual. Fig. 1 summarizes the proposed threshold-based TCP Vegas scheme.

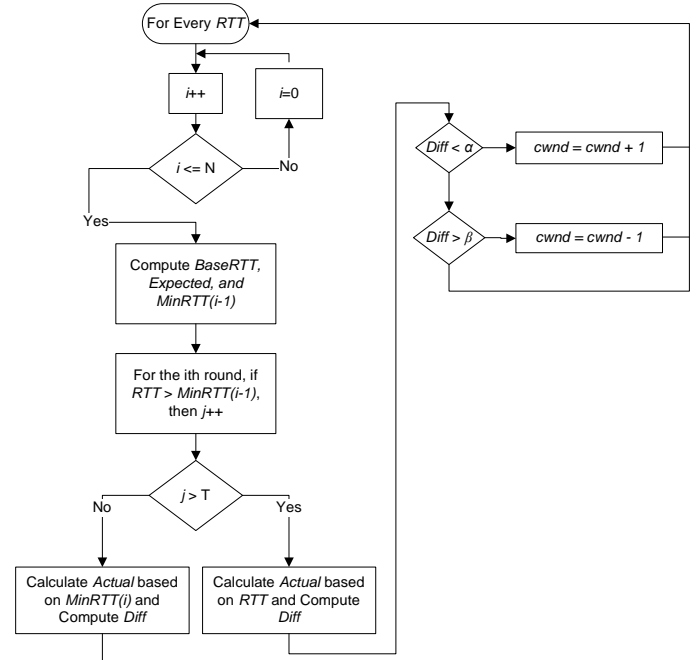


Fig. 1. The flowchart for the threshold-based TCP Vegas congestion control algorithm.

The basic idea behind threshold-based TCP Vegas is to reduce the sensitivity of TCP Vegas to the increases of $RTTs$ caused by burst retransmission in the OBS domain. Instead of changing $cwnd$, the sensitivity of TCP Vegas can also be

reduced by decreasing α or increasing β . However, changing α and β in this way makes it difficult for TCP Vegas to estimate the available bandwidth in the networks.

In threshold-based TCP Vegas, both N and T should be chosen to be much larger than the number of packets from a single TCP Vegas connection that are assembled into a burst, such that TCP Vegas is able to detect the frequency of retransmission in the OBS domain based on the record of a number of bursts. In general, the packets from a common TCP connection assembled in the same burst have the same measured RTT. By analyzing the variation pattern of historical RTTs, TCP Vegas can obtain the number of packets from a TCP Vegas connection that are assembled into a burst. T and N can also affect the TCP performance. When T is chosen closer to N, there would be fewer remaining packets in the N consecutive packets to react to the detected congestion, which results in an ineffective response to the network congestion. Hence, we take $N=iT$, where $i > 1$. In Section V, the TCP throughput performance will be evaluated according to different values of T and N.

The threshold-based TCP Vegas is particularly suitable for TCP flows that need to transfer a large amount of data for a relatively long period of time. These TCP flows are becoming important for many emerging applications that require high bandwidth, such as Grid, storage area networks, and content distribution networks. Depending on the number of TCP packets assembled in the contended burst, these flows may either trigger a TO or cut the *cwnd* to half as a response to the receiving of TDs. The time required for increasing the *cwnd* (most likely through an additive increase) to its previous size could be very long. In this case, burst losses in a non-congestion state can significantly downgrade the throughput of those high bandwidth applications.

IV TCP VEGAS PERFORMANCE MODELING OVER OBS

In this section, we analyze the throughput of the proposed threshold-based TCP Vegas scheme over OBS networks with burst retransmission. We assume that the access bandwidth of a TCP flow is high such that all TCP packets in a single congestion window are assembled into the same burst. Such a TCP flow is referred to as a TCP fast flow [20]. TCP fast sources will not trigger triple duplicates since packets in an entire congestion window are lost due to a burst loss. We assume that the burst assembly delay is fixed, and that the occurrences of burst losses and contentions are independent based on a given and fixed burst dropping probability and burst contention probability. Burst dropping probability, defined as the probability that a burst is dropped, is usually much lower than burst contention probability, defined as the probability that a burst experiences contention [15]. This assumption can be justified as some of the contended bursts can successfully reach their destinations after being retransmitted. Thus, the burst dropping probability is always less than or equal to the burst contention probability.

We first analyze the throughput of TCP Vegas over barebone OBS networks based on the analytical model proposed in [17], followed by the analysis of TCP Vegas over

OBS networks with burst retransmission. Last, we analyze the throughput of the proposed threshold-based TCP Vegas over OBS networks with burst retransmission by integrating the two previously presented analytical models. The following table lists the notations adopted in the modeling processes.

p	: Burst dropping probability
p_c	: Burst contention probability
p_{nc}	: Probability of no burst contention
p_{sr}	: Probability of a burst contended but successfully retransmitted through burst retransmission
α, β	: Vegas throughput thresholds in packets
$BaseRTT$: Minimum measured RTT
R	: Expected without burst retransmission
RTT_r	: Expected RTT with burst retransmission
B	: Vegas throughput
H	: Expected number of packets submitted during a TO period
RTO	: TCP retransmission timeout
TOP	: Duration of a sequence of TOs
X	: Number of consecutive successful rounds
A	: Duration of a sequence of consecutive successful rounds
Y	: Number of packets sent before a TO period
W_{max}	: Maximum <i>cwnd</i> in packets
W_0	: <i>cwnd</i> in Vegas stable state

In the analysis, a TCP *round* refers to the period during which all packets in the congestion window are sent and the first ACK for one of the packets in the congestion window is received. With fast TCP flows, TCP duplicate ACKs will never be triggered. Hence, multiple successful sending rounds will be followed by one or more lossy rounds. Therefore, we obtain the throughput of TCP Vegas fast flows as follows:

$$B = \frac{E[Y] + E[H]}{E[A] + E[TOP]}. \quad (5)$$

A TCP Vegas over Barebone OBS

In this section, TCP Vegas throughput over a barebone OBS is analyzed. In barebone OBS, random burst contention results in an immediate burst loss, since no burst retransmission is implemented. With the assumption that the IP access network is not congested, *RTT* would be very close to *BaseRTT*.

Fig. 2 shows the evolution of *cwnd* for TCP Vegas over a barebone OBS network. The evolution of *cwnd* can be partitioned into the following periods: (1) The slow-start period is the duration from A to B in Fig. 2, in which *cwnd* starts at 2 and doubles every other round until it reaches the expected slow-start threshold. (2) The transition period is from B to C. Since *RTT* is close to *BaseRTT*, *Diff* calculated based on Eq. (3) will be a very small value less than α . Hence, in the transition period, *cwnd* increases by 1 every RTT, until TCP Vegas reaches the stable state, $\alpha \leq Diff \leq \beta$. (3) The loss-free period includes a series of consecutive successful rounds from C to G, during which Vegas is in the stable state (*cwnd*

remains unchanged) and no burst loss occurs. (4) The TO period includes one or more consecutive timeout events. The duration from slow-start to the first encountered TO period (or the beginning of slow-start of the next sequence of rounds) is referred to as slow-start-to-slow-start (SS2SS) period [21]. We assume that burst dropping probability is low, such that there are no burst losses during the slow-start period and the transition period.

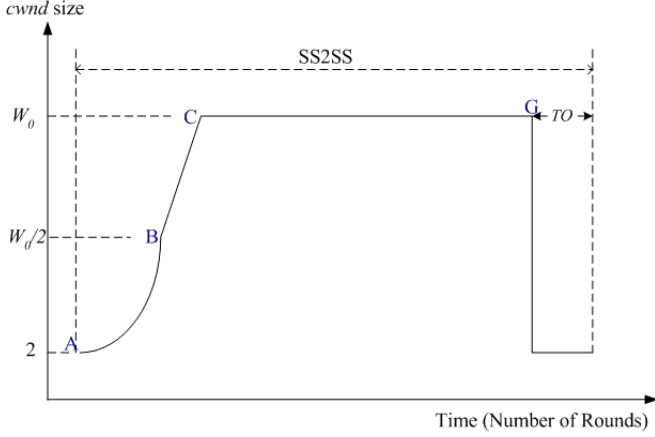


Fig. 2. Evolution of $cwnd$ for TCP Vegas over a barebone OBS.

In Fig.2, W_0 is the congestion window size when Vegas is in stable state, i.e. $\alpha \leq Diff \leq \beta$. Since TCP Vegas adjusts the backlog packets in the network between α and β , the expected value of $Diff$ is estimated as $\frac{\alpha + \beta}{2}$ based on [21]. By applying Eq. (3), we have

$$Diff = W_0 \left(1 - \frac{BaseRTT}{R}\right) = \frac{\alpha + \beta}{2}. \quad (6)$$

By rearranging Eq. (6), we can obtain W_0 as:

$$W_0 = \min\left(\frac{\alpha + \beta}{2} \times \frac{R}{R - BaseRTT}, W_{max}\right). \quad (7)$$

We now calculate the TCP Vegas throughput by computing the expected number of packets transmitted in the period from A to G. In the slow-start period from A to B, $cwnd$ doubles every other round until it reaches the slow-start threshold, $W_0/2$, since the expected $cwnd$ when the TO occurs is W_0 [21]. Thus, based on Eq. (17) in [21], the number of packets sent in the period from A to B, Y_{AB} , is

$$Y_{AB} = 2 \sum_{i=0}^{\log \frac{W_0}{4}} 2^i = 2^{\log W_0} - 4, \quad (8)$$

and the duration of the slow-start phase is

$$A_{AB} = 2 \left(\frac{\log W_0}{4}\right) R = 2(\log W_0 - 2)R, \quad (9)$$

where R is the expected RTT .

In the transition period from B to C, let the average number of packets transmitted be Y_{BC} and the duration of that period be A_{BC} . Based on Eq. (18) in [21], Y_{BC} can be obtained by

aggregating the number of packets sent when $cwnd$ increases from $W_0/2$ to W_0 as follows:

$$Y_{BC} = \sum_{i=W_0/2}^{W_0-1} i = \frac{3W_0^2}{8} - \frac{W_0}{4}. \quad (10)$$

Since $cwnd$ increases by 1 for each RTT during the transition period, the expected duration of this period is,

$$A_{BC} = \left(\frac{W_0}{2}\right)R. \quad (11)$$

We then calculate the average number of packets transmitted in the loss-free period from C to G (denoted by Y_{CG}) and the duration of that period (denoted as A_{CG}). We assume a Bernoulli burst loss model. Hence, the number of rounds during the loss-free period, $S_{lossfree}$, is

$$S_{lossfree} = \sum_{k=0}^{\infty} k(1-p)^k p = (1-p)/p. \quad (12)$$

Then the duration of the loss-free period is

$$A_{CG} = R S_{lossfree}. \quad (13)$$

Since $cwnd$ is equal to W_0 during the loss-free period from C to G, the number of packets sent in this period is:

$$Y_{CG} = W_0 S_{lossfree}. \quad (14)$$

In the TO period, since the timeout process of TCP Vegas is similar to TCP Reno, based on Eqs. (14) and (16) in [20], we have the mean duration for successive TOs as

$$E[TO] = RTO \frac{f(p)}{1-p}, \quad (15)$$

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

$$E[H] = \frac{p}{1-p}. \quad (16)$$

We then have the average number of packets sent during the consecutive successful rounds: $E[Y] = Y_{AB} + Y_{BC} + Y_{CG}$, and the duration of successful rounds: $E[A] = A_{AB} + A_{BC} + A_{CG}$. By substituting Eqs. (7) - (16) into Eq. (5), we obtain TCP Vegas throughput over a barebone OBS network as follows:

$$B_{barebone} = \frac{Y_{AB} + Y_{BC} + Y_{CG} + E[H]}{A_{AB} + A_{BC} + A_{CG} + TO}. \quad (17)$$

B TCP Vegas over OBS with Burst Retransmission

We now analyze TCP Vegas throughput over an OBS network with burst retransmission, which is expected to greatly improve the loss probability in the OBS domain at the expense of introducing longer RTT as perceived by the TCP layer.

We assume that retransmitted bursts that successfully reach their destination experience an average round trip delay, RTT_r . We further identify two types of successful rounds as follows: (1) the rounds that experience contention but in which bursts are successfully retransmitted, and (2) the rounds that do not experience burst contention. The probability of a successful round that experiences contention but in which bursts are successfully retransmitted is:

$$p_{sr} = \frac{p_c - p}{1-p}. \quad (18)$$

The probability of a successful round that does not experience burst contention can be calculated as

$$p_{nc} = \frac{1 - p_c}{1 - p} \quad (19)$$

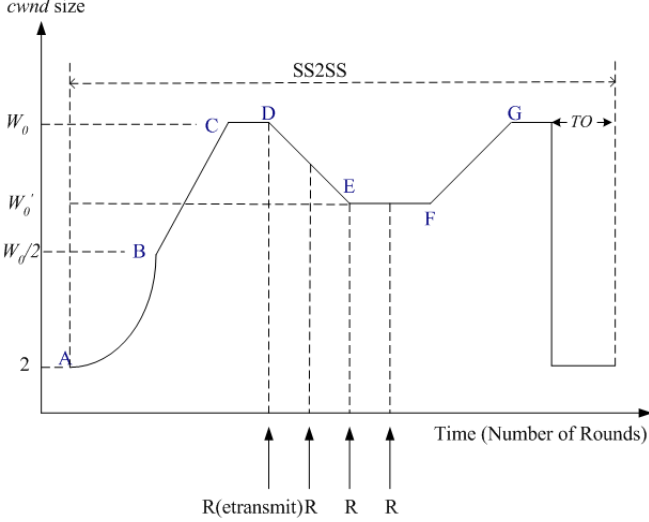


Fig. 3. Evolution of $cwnd$ for TCP Vegas over an OBS networks with burst retransmission.

Fig. 3 shows the evolution of $cwnd$ for TCP Vegas over an OBS network with burst retransmission. The SS2SS period is partitioned into 4 periods: (1) the slow-start period from A to B, (2) the transition period from B to C, (3) the loss-free period from C to G, where rounds may suffer longer RTT due to burst retransmission, and (4) the TO period. In Fig. 3, the slow-start period, the transition period, and the TO period are similar to those in Fig. 2. In this section, we focus on the analysis of the loss-free period from C to G.

Since TCP Vegas reaches stable state at point C, the value of the $Diff$ calculated at point C is

$$Diff_C = W_0 \left(1 - \frac{BaseRTT}{R}\right) \quad (20)$$

During the period from C to G, consider a round with the current $cwnd$ equal to W_0 (see the point D in Fig. 3). If a burst is successfully retransmitted in the OBS domain during this round, all the packets sent in this round will experience a longer delay, RTT_r . Hence, the $Diff$ calculated at point D is

$$Diff_D = W_0 \left(1 - \frac{BaseRTT}{RTT_r}\right) \quad (21)$$

If $\alpha \leq Diff_D \leq \beta$, TCP Vegas will leave $cwnd$ unchanged from C to G. Hence, by rearranging Eq. (21), we can obtain the range of RTT_r that keeps $cwnd$ constant, that is,

$$\frac{BaseRTT}{1 - (\alpha/W_0)} \leq RTT_r \leq \frac{BaseRTT}{1 - (\beta/W_0)}$$

If $Diff_D > \beta$, $cwnd$ is decreased by 1. If multiple consecutive rounds are retransmitted in the OBS domain, $Diff$ will keep decreasing for each of the consecutive rounds due to the decrease of $cwnd$ until $Diff$ reaches β (see the duration from D

to E in Fig. 3). After $Diff$ reaches β , $cwnd$ remains unchanged for the future consecutive rounds retransmitted in the OBS domain (referred to the duration from E to F in Fig. 3). Let W_0' be the lower bound of $cwnd$ from C to G. We have $Diff$ at point E as:

$$Diff_E = W_0' \left(1 - \frac{BaseRTT}{RTT_r}\right) \leq \beta$$

Then we have

$$W_0' = \left\lceil \frac{\beta RTT_r}{RTT_r - BaseRTT} \right\rceil \quad (22)$$

Note that, for a round with a longer delay RTT_r , when $Diff_D < \alpha$, $cwnd$ will remain unchanged, because if $W_0 = W_{max}$, then $cwnd$ has already reached the maximum window size. If $W_0 < W_{max}$, then $Diff_C \geq \alpha$ and $R < RTT_r$. Thus, we have $Diff_D \geq Diff_C \geq \alpha$ from Eqs. (20) and (21).

For any non-retransmitted round during the loss-free period, if $cwnd < W_0$, then $cwnd$ increases by 1 because $cwnd < W_0$ and $Diff = cwnd \left(1 - \frac{BaseRTT}{R}\right) < \alpha$. For example, during

the period from F to G in Fig. 3, $cwnd$ increases by 1 for each non-retransmitted round until $cwnd$ reaches W_0 .

Thus, we can conclude that $cwnd$ ranges between W_0 and W_0' . For each round, if $W_0' \leq cwnd < W_0$, $cwnd$ either increases by 1 if the round does not experience contention, or decreases by 1 if the round is retransmitted in the OBS domain. If $cwnd = W_0'$, $cwnd$ remains unchanged if the round is retransmitted in OBS domain, and increases by 1 if the round does not experience contention. If $cwnd = W_0$, $cwnd$ decreases by 1 if the round is retransmitted in the OBS domain, and remains unchanged if the round does not experience contention.

Let the probability for a successful retransmission be denoted as p_{sr} , and the probability that a round does not experience contention be denoted as p_{nc} . We model the state of $cwnd$ as a Markov chain shown in Fig. 4. Let π_i be the probability of $cwnd = W_0 - i$. We solve π_i by a local balance equation as follows.

$$p_{nc}\pi_1 = p_{sr}\pi_0 \Rightarrow \pi_1 = \frac{p_{sr}}{p_{nc}}\pi_0$$

$$p_{nc}\pi_2 = p_{sr}\pi_1 \Rightarrow \pi_2 = \frac{p_{sr}}{p_{nc}}\pi_1 = \left(\frac{p_{sr}}{p_{nc}}\right)^2\pi_0$$

...

$$p_{nc}\pi_{(W_0 - W_0' - 1)} = p_{sr}\pi_{(W_0 - W_0')} \Rightarrow \pi_{(W_0 - W_0')} = \left(\frac{p_{sr}}{p_{nc}}\right)^{(W_0 - W_0')} \pi_0$$

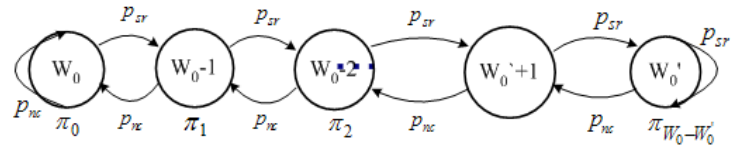


Fig. 4. Markov Chain for the state of $cwnd$.

Since $\sum_{i=0}^{w_0-w_0'} \left(\frac{p_{sr}}{p_{nc}}\right)^i \pi_0 = 1$, we obtain

$$\pi_0 = \frac{1}{\sum_{i=0}^{w_0-w_0'} \left(\frac{p_{sr}}{p_{nc}}\right)^i} = \begin{cases} \frac{1}{W_0 - W_0' + 1}, & \text{if } p_{sr} = p_{nc}, \\ \frac{1 - \left(\frac{p_{sr}}{p_{nc}}\right)^{W_0 - W_0' + 1}}{1 - \frac{p_{sr}}{p_{nc}}}, & \text{if } p_{sr} \neq p_{nc}. \end{cases} \quad (23)$$

Then, π_i can be obtained using the local balance equations. The expected congestion window size from C to G can be obtained as

$$\begin{aligned} E[W_{CG}] &= W_0 \pi_0 + (W_0 - 1) \pi_1 + \dots + W_0' \pi_{(W_0 - W_0')} \\ &= \sum_{i=0}^{W_0 - W_0'} [(W_0 - i) \pi_i]. \end{aligned}$$

In case $BaseRTT \left/ \left(1 - \frac{\alpha}{W_0}\right) \leq RTT_r \leq BaseRTT \left/ \left(1 - \frac{\beta}{W_0}\right)\right.$, TCP Vegas keeps the $cwnd$ unchanged from C to G and then $E[W_{CG}] = W_0$.

During the period from C to G, the number of rounds is $S_{lossfree}$, and the expected $cwnd$ is $E[W_{CG}]$. Hence, the number of packets sent during the period from C to G is:

$$Y_{CG}' = E[W_{CG}] S_{lossfree}. \quad (24)$$

The duration between C and G includes the rounds that are retransmitted and with delay RTT_r , and the rounds that are not subject to contention and with the delay R . Hence, we have

$$A_{CG}' = (p_{nc}R + p_{sr}RTT_r) S_{lossfree}. \quad (25)$$

From Eqs. (8), (10), and (24), we obtain $E[Y]$ as

$$E[Y] = Y_{AB} + Y_{BC} + Y_{CG}', \quad (26)$$

and from Eqs. (9), (11), and (25), we obtain $E[A]$ as

$$E[A] = A_{AB} + A_{BC} + A_{CG}'. \quad (27)$$

We then obtain TCP Vegas throughput over OBS networks with burst retransmission by substituting Eqs. (15), (16), (26), and (27) into Eq. (5):

$$B_{ret} = \frac{Y_{AB} + Y_{BC} + Y_{CG}' + E[H]}{A_{AB} + A_{BC} + A_{CG}' + TOP}. \quad (28)$$

C Threshold-based TCP Vegas over OBS with Burst Retransmission

Based on the two analytical models, we can analyze the throughput of threshold-based Vegas flows over OBS with burst retransmission.

Fig. 5 illustrates the evolution of $cwnd$ during the time when N consecutive packets are sent using the threshold-based TCP Vegas. Before the number of TCP packets whose RTT s are larger than the current $MinRTT$ is under the threshold T , TCP Vegas stays with $MinRTT$ for calculating $Diff$. Hence, the $cwnd$ evolution of the threshold-based TCP is similar to that of TCP Vegas over barebone OBS, where $R = MinRTT$. Thus, the

throughput before reaching T , denoted as $B'_{barebone}$, can be calculated based on Eq. (17), where

$$W_0 = \min\left(\frac{\alpha + \beta}{2} \times \frac{R}{MinRTT - BaseRTT}, W_{max}\right).$$

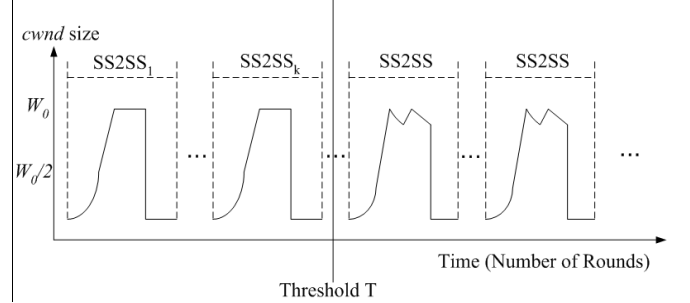


Fig. 5. Evolution of $cwnd$ for TCP threshold-based Vegas over an OBS network with burst retransmission.

After the number of TCP packets whose RTT s are larger than the current $MinRTT$ reaches T (after $SS2SS_k$ in Fig. 5), threshold-based TCP Vegas will have a $cwnd$ evolution similar to that of TCP Vegas over OBS networks with burst retransmission. Hence, the throughput after reaching T B'_{ret} , can be calculated based on Eq. (28).

Let S_1 be the expected number of $SS2SS$ periods before reaching T and let S_2 be the expected number of $SS2SS$ periods after reaching T . We know that the expected number of consecutive successful rounds in a $SS2SS$ period is $E[X] = (A_{AB} + A_{BC}) / R + S_{lossfree}$. Hence, the total number of rounds before reaching T is $(S_1 E[X])$, and the number of rounds that experience retransmission and extra delay is $(S_1 E[X] p_{sr})$. On the other hand, when the threshold T is reached, the number of rounds that are retransmitted is approximately (T/W_0) since the expected $cwnd$ when the TO occurs is W_0 [21]. We have $S_1 E[X] p_{sr} = T/W_0$, and then

$$S_1 = \left\lceil \frac{T}{W_0 E[X] p_{sr}} \right\rceil. \quad (29)$$

In each $SS2SS$ period before reaching T , we can obtain the number of packets sent as $(Y_{AB} + Y_{BC} + Y_{CG}')$ from Fig. 2. Then, the total number of packets sent before reaching the threshold is $S_1(Y_{AB} + Y_{BC} + Y_{CG}')$. Since the total number packets measured are within N consecutive packets, the total number of packets sent after reaching the threshold T is $[N - S_1(Y_{AB} + Y_{BC} + Y_{CG}')] p_{sr}$, where $N > S_1(Y_{AB} + Y_{BC} + Y_{CG}')$. Since the number of packets sent is $(Y_{AB} + Y_{BC} + Y_{CG}')$ in each $SS2SS$ period after reaching T , we can obtain the number of $SS2SS$ periods sent after reaching T as

$$S_2 = \begin{cases} \left\lceil \frac{N - S_1(Y_{AB} + Y_{BC} + Y_{CG}')} {Y_{AB} + Y_{BC} + Y_{CG}'} \right\rceil, & N > S_1(Y_{AB} + Y_{BC} + Y_{CG}'), \\ 0, & \text{otherwise.} \end{cases} \quad (30)$$

Therefore, we can obtain the throughput during the time when N consecutive packets are sent:

$$B_{\text{threshold-based}} = \frac{S_1}{S_1 + S_2} B'_{\text{barebone}} + \frac{S_2}{S_1 + S_2} B'_{\text{ret}}. \quad (31)$$

We assume that the number of consecutive packets N is large enough such that the throughput of the N consecutive packets can represent the throughput of an entire TCP session. Thus, the throughput during the time when N consecutive packets are sent can be an approximate of the throughput of an entire TCP session.

V NUMERICAL RESULTS

In this section we present the numerical results obtained from both the developed analytical models and a simulation using NS-2. In order to obtain precise protocol performance results, we limit the network complexity as well as the parameter space to a multi-hop network as shown in Fig. 6, where the distances between the nodes are in kilometers. The edge nodes, E_1 and E_2 , are connected to the network core nodes with a 10 ms propagation delay link. Each link consists of one bi-directional control channel used for the control signalling and a single fiber link used for data burst transfer. Each data link consists of 8 wavelengths each operating at a transmission rate 10 Gbps. The mixed time/length based burst assembly algorithm is adopted, where the burst timeout threshold is set to 500 ms, and the maximum burst length is set to 50 kB. The control header processing time is set to be 1 μ s. The core nodes implement the *LAUC-VF* channel scheduling algorithm [10]. In the burst retransmission mechanism, bursts subject to any contention are allowed to be retransmitted only once in order to have the best chance of meeting the timeout threshold. In our initial experiments, we assume that the TCP flows under investigation do not significantly affect the overall load in the OBS network; thus, we treat the burst contention probability p_c as an input parameter and vary its value in the range $[10^{-5}, 10^{-2}]$. The average *RTT* in the simulation is approximately 110 ms.

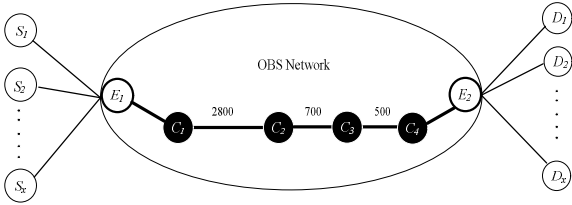


Fig. 6. The network topology adopted in the simulation.

In the simulation, TCP senders and receivers are attached to the OBS edge nodes E_1 and E_2 respectively. In order to simulate TCP fast flows, a high access bandwidth is allocated to each flow. The packet delay in the access network is set to be constant such that the effect of the burst retransmission delay is the only reason for longer round trip time of each packet. The FTP application is initiated to generate TCP segments with an average size of 1kB. In all experiments, the maximum window size of TCP is 10^4 segments. We select $\alpha=600$ and $\beta=800$ for both TCP Vegas and the threshold-based Vegas. The TCP throughput is obtained over a simulation

period of 10^4 seconds. Results in the following sections are based on simulation unless otherwise stated.

A Threshold-based TCP Vegas over OBS

In the following experimental studies, we compare the throughput of conventional TCP Vegas, threshold-based Vegas, and loss-based TCP Sack (TCP Sack performs better than other existing loss-based TCP implementations in OBS networks [14]).

Fig. 7 shows the throughput of TCP Vegas, threshold-based Vegas, and Sack over a barebone OBS network. Note that the burst contention probability is equal to the burst loss probability in the barebone OBS network. Threshold-based Vegas has the threshold T from 100 to 400, while the number of consecutive TCP packets N is chosen to be $4T$. We see that threshold-based TCP Vegas with different N and T values and conventional TCP Vegas with $N = 0$ and $T = 0$ perform very similar, such that their throughput plots are overlapped. This result is due to the fact that the round trip time does not vary significantly in the barebone OBS network. We can also see that TCP Vegas versions perform much better than loss-based TCP Sack in the barebone OBS network.

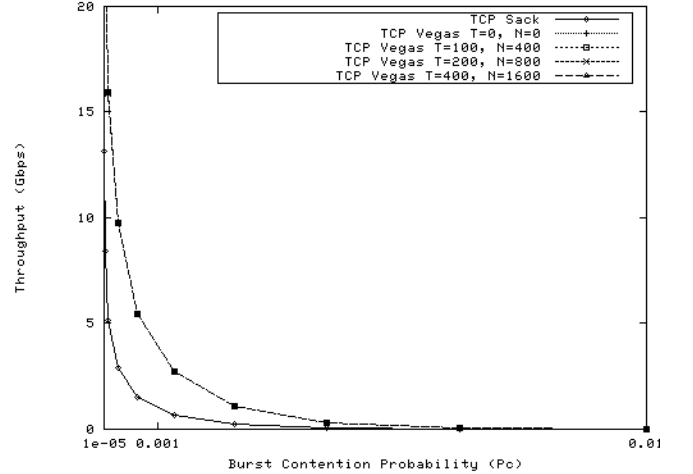


Fig. 7. Throughput comparison of Vegas, threshold-based Vegas ($N = 4T$), and Sack over a barebone OBS.

Fig. 8 compares the throughput of TCP Vegas, threshold-based Vegas, and TCP Sack over an OBS network with burst retransmission. Note that, with one attempt of retransmission, the burst loss probability in the simulation is much smaller than the burst contention probability. We observe that threshold-based Vegas performs better than conventional Vegas and TCP Sack. For example, when the burst contention probability is 10^{-4} , TCP Vegas with $T = 200$ improves the throughput by 51% compared to conventional Vegas. We can also see that, with higher threshold, threshold-based Vegas performs better. When the burst contention probability is 10^{-4} , threshold-based Vegas with $T = 400$ improves the throughput by 71% compared to the case with $T = 200$. This occurs because threshold-based Vegas with $T = 400$ more accurately

detects the congestion state in the OBS network and delays the triggering of the congestion avoidance mechanism.

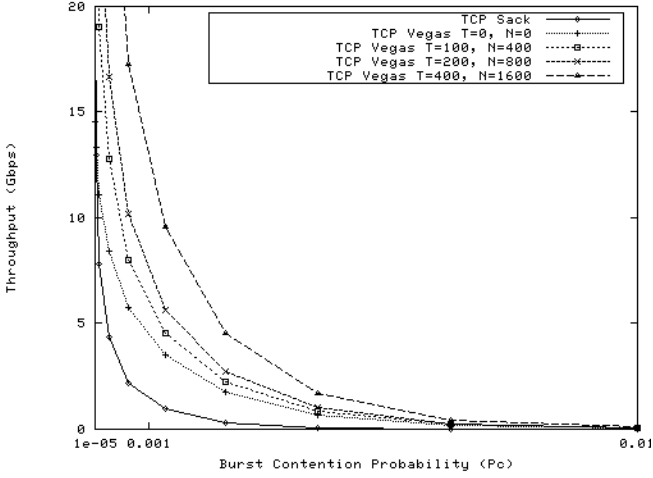


Fig. 8. Throughput comparison of TCP Vegas, threshold-based Vegas ($N = 4T$), and Sack over an OBS network with burst retransmission.

In Fig. 9, we examine the effect of T and N in threshold-based Vegas. We observe that, with a fixed T value, varying N values does not result in a major throughput change. For example, with a fixed $T = 200$, the throughput of $N = 6400$ and $N = 12800$ are very close. We can also see that the throughput is primarily affected by the value of T . For instance, the throughput of $T = 6400$ increases 92% compared to the throughput of $T = 800$ when $N = 12800$.

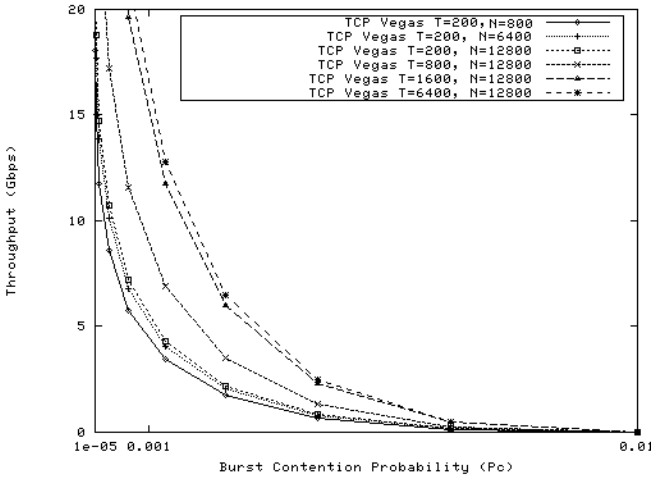


Fig. 9. Throughput comparison of the threshold-based Vegas with fixed N or fixed T values.

B Threshold-based TCP Vegas Fairness Evaluation

In this section, we examine the fairness of TCP Vegas, Threshold-based Vegas, Sack, and Reno. For this purpose, we use the *Jain fairness index* which is defined as $(\sum_{i=1}^n B_i)^2 / n \sum_{i=1}^n B_i^2$, where n is the number of competing

flows and B_i is the throughput of the i th flow. The fairness index ranges from $1/n$ to 1 [22]. If all flows have the same throughput, then the fairness index is 1. In our simulation, we generate three flows of each TCP version. The competing flows are sharing the same source-destination pairs. We then calculate the fairness index for each TCP version after obtaining the throughput of each flow.

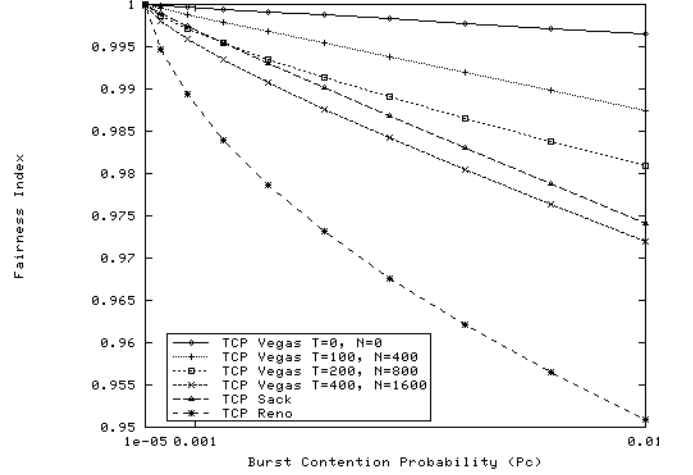


Fig. 10. Fairness Index of Vegas, threshold-based Vegas ($N = 4T$), Sack, and Reno.

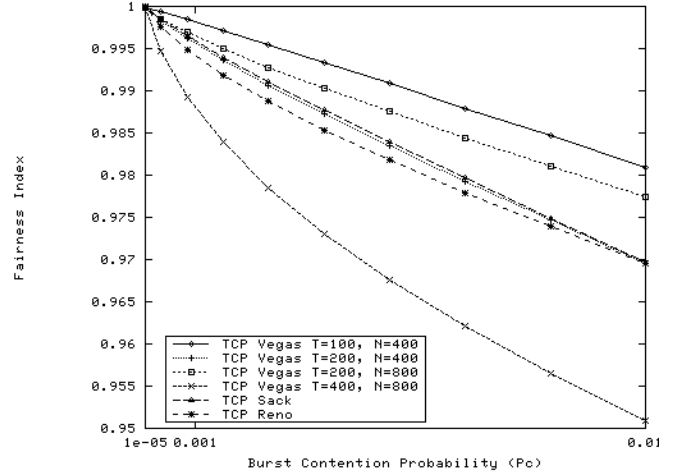


Fig. 11. Fairness Index of Vegas, threshold-based Vegas with fixed N or fixed T , Sack, and Reno.

Fig. 10 and Fig. 11 compare the fairness index of TCP Vegas, threshold-based Vegas, Sack, and Reno. It is notable that the fairness index of threshold-based Vegas with larger T value results in lower fairness among the other co-existing flows. This result is due to the fact that the window size reduction in threshold-based Vegas is substantially delayed when RTT increases. Thus, threshold-based Vegas is given the chance to occupy more link bandwidth at the expense of other TCP streams. The fairness index shows that the throughput of threshold-based Vegas is close to that of Sack and Reno. Also, the throughput is close among competing threshold-based Vegas flows while varying T and N values. Thus, we can see

that threshold-based Vegas can maintain a friendly condition along with other TCP versions.

C Analytical Results

In this section, we verify the analytical models proposed in Section IV. The analytical results yield the TCP throughput under different burst contention probabilities p_c .

Fig. 12 compares the throughput of conventional TCP Vegas obtained from Eq. (17) to the throughput obtained from simulation with different burst contention probability p_c . The average RTT delay was 110 ms. We can see that the simulation results and the analytical results are very close.

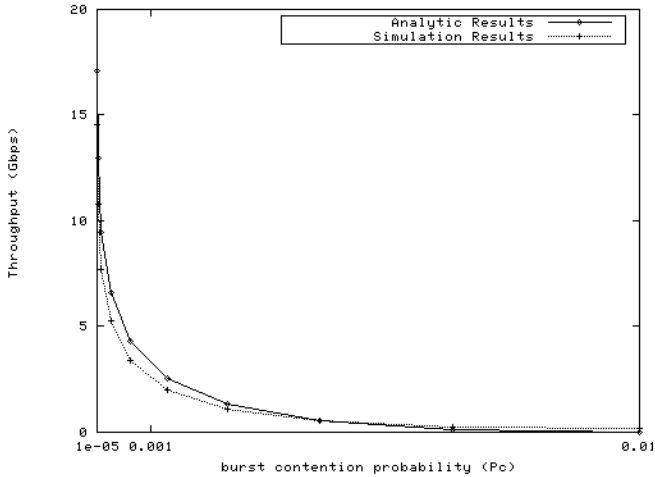


Fig. 12. TCP Vegas throughput over a barebone OBS network.

Fig. 13 compares the analytical results obtained by Eq. (28) with simulation results. In the experiment, the OBS edge node enables the burst retransmission mechanism. Bursts are retransmitted at most once. The average RTT delay in the simulation was 110ms. Bursts that are retransmitted experience an average of 100 ms extra delay. We see that the simulation results match the analytical results very well.

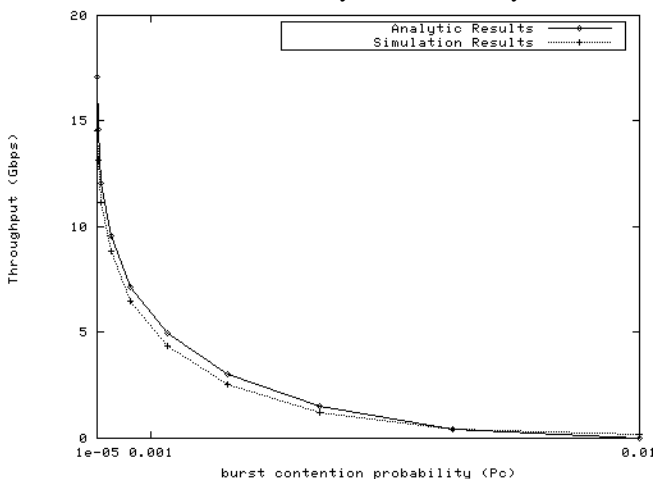


Fig. 13. TCP Vegas throughput over an OBS network with burst retransmission.

Fig. 14 compares the analytical results obtained by Eq. (31) and the simulation results obtained for the threshold-based Vegas fast flow. In this experiment we compare, $T = 100$ and $N = 400$, $T=200$, $N=800$, and $T=400$, $N=1600$. We can see that the simulation results match the analytical results.

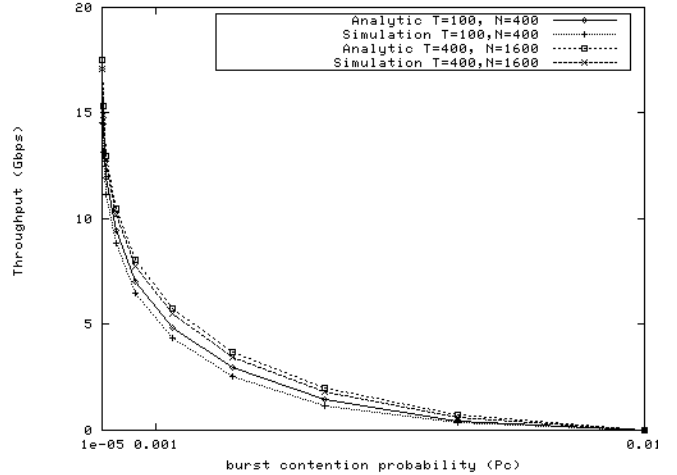


Fig. 14. Threshold-based Vegas throughput over an OBS network with burst retransmission.

The analytical model can be used to evaluate the effect of contention probability and threshold parameters on TCP throughput. The model can also be used to determine the steady state operating point of the network when combined with an analytical model that evaluates burst contention probability, which we present in the following section.

D Threshold-based TCP Vegas Steady State over OBS

In previous sections, we assume that the burst contention probability does not change when the TCP throughput varies. However, if there are a finite number of TCP flows in the network and most senders are transferring large files using TCP, the sending rate of a TCP sender is tightly coupled with packet loss within the network [23], [24]. A high packet loss rate will cause a TCP sender to slow down, thereby reducing the network load and decreasing the subsequent packet loss rate in the network. Hence, in this section, we apply the analytical model to analyze the steady-state of the total input load and the burst contention probability for threshold-based Vegas over OBS networks. We verify the existence of the steady state through simulation. We also aim to obtain the proper N and T values which maximizes the input load and correspondingly maximize the TCP throughput.

The fixed-point method based on the TCP feedback mechanism is adopted from [23, 24]. For each input load, we can calculate the burst contention probability in an OBS network using the OBS contention probability model proposed in [15, 16], shown as a solid line labelled “OBS” in Fig. 15. On the other hand, for each burst contention probability, we can determine the corresponding TCP throughput using Eq. (31) and then determine the total network input load, shown

by dashed lines in Fig. 15. The intersection of a dashed line and the solid line is a point at which the solution of the OBS contention probability equation matches the solution of the TCP throughput equation, and is the point where the system could converge.

We now describe the details of the analysis for OBS contention probability. Let the throughput of a TCP flow from source s to destination d be $S_{s,d}$, and the size of a TCP packet be pkt . The input load from a TCP flow is

$$\rho_{s,d} = S_{s,d} / pkt.$$

Using the input load, we can then obtain the burst contention probability in the OBS network based on the analytical model proposed in [15, 16]. Let p_c be the burst contention probability along the path from s to d . Since all the dropped bursts are able to retransmit only once, the total load including the retransmitted traffic is,

$$\mathfrak{R}_{s,d} = \rho_{s,d}(1 + p_c).$$

Then the total load on link l_{ij} is

$$\mathfrak{R}_{ij} = \sum_{(\forall s,d,l_{ij} \in route(s,d))} \mathfrak{R}_{s,d}.$$

The burst arrival on each link is assumed to be a Poisson distribution [25]. By using the Erlang-B formula, the burst contention probability on link l_{ij} can be obtained as:

$$p_{ij} = ErlangB(\mathfrak{R}_{ij}, m),$$

where m is the number of wavelengths on each link. Hence, assuming that links are independent of one another, the burst contention probability along the path from s to d , p_c' is

$$p_c' = 1 - \prod_{(\forall s,d,l_{ij} \in route(s,d))} (1 - p_{ij}). \quad (32)$$

The converged p_c and p_c' will be the burst contention probability for the given TCP throughput. Note that the actual burst loss probability is much smaller than the burst contention probability due to burst retransmission.

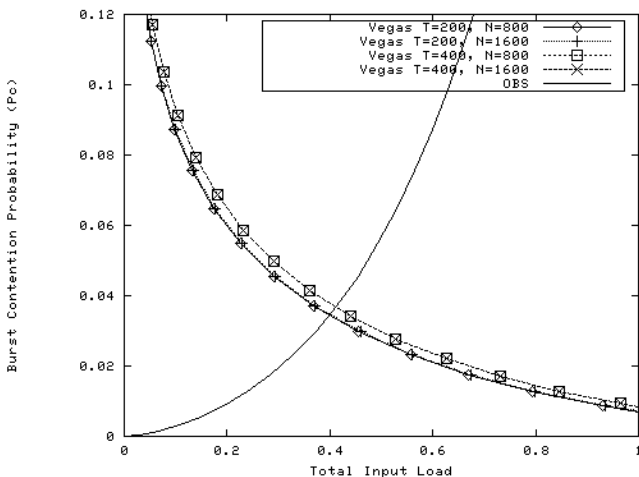


Fig. 15. Analytical results of the steady-state throughput of threshold-based Vegas with $T = 200$ and $T = 400$.

Fig. 15 shows the analytical results of the steady state threshold-based Vegas throughput over OBS with burst retransmission. From the figure, we can see that, when T is 200 and N varies from 800 to 1600, the steady state throughputs of threshold-based Vegas are very close. This is also true when T is 400 and N varies. These results validate the findings in Section V. A, where we concluded that the threshold T has a dominant effect on the throughput.

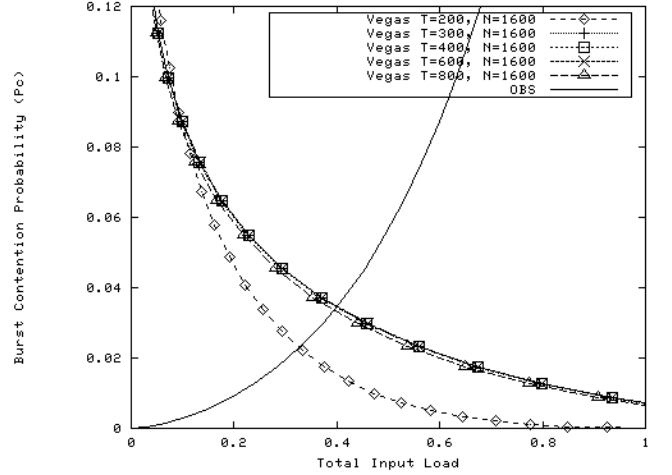


Fig. 16. Analytical results of the steady-state throughput of threshold-based Vegas with $N = 1600$.

Fig. 16 plots analytical results of the steady state threshold-based Vegas throughput with fixed $N = 1600$. From this experiment, we observe that the threshold-based Vegas flows with $T \geq 300$ do not have a significant throughput increase. For the flows where $T \geq 300$, the steady state throughput remains close to the flows with $T = 800$. Therefore, we conclude that $T = 300$ and $N = 1600$ can be considered as proper parameters for obtaining the best threshold-based throughput.

From both Fig. 15 and 16 we observe that different values of T and N affect the overall network load. This effect is reflected on the overall throughput gain for each TCP flow.

Table 1 shows the simulation results which confirm that the different configurations of TCP Vegas converge the steady state.

	P_c	Input load (simulation)	Input load (Analytical)
$T=200, N=800$	0.036147	0.400214	0.4
$T=200, N=1600$	0.036147	0.400214	0.4
$T=400, N=800$	0.040100	0.428198	0.4
$T=400, N=1600$	0.040100	0.428198	0.4

Table 1 Steady state total input load and P_c for each pair of T and N .

VI CONCLUSIONS

In this paper, we have investigated the issue of false congestion identification when delay-based TCP Vegas is implemented over OBS networks. A threshold-based TCP Vegas mechanism

was proposed for effective detection of network congestion by manipulating a threshold parameter when TCP runs over OBS networks with burst retransmission. Our simulation results showed that the proposed scheme can significantly outperform the conventional TCP Vegas and TCP Sack. Simulation results showed that the threshold T has a more significant impact on the TCP throughput than N . We have also analyzed the throughput performance of a fast threshold-based Vegas source over an OBS network with burst retransmission. Based on the analytical model, we have obtained the steady state TCP throughput and have observed the proper threshold value that results in an optimal throughput. The analytical model provides insights regarding the effect of burst contentions and losses in the OBS domain on the TCP throughput.

REFERENCES

- [1] S. Floyd, "Quick-Start for TCP and IP," *Internet draft, draft-amit-quick-start-02.txt*, 2002.
- [2] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *Proceedings, IEEE Infocomm*, Hong Kong, China, March 2004.
- [3] S. Hegde, *et al.*, "FAST TCP in High-speed Networks: An Experimental Study," *Proceedings, GridNets*, San Jose, CA, October 2004.
- [4] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-distance Networks", *Proceedings, IEEE Infocom*, Hong Kong, China, March 2004.
- [5] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *RFC 2001*, 1997.
- [6] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," *RFC 2018*, 1996.
- [7] L. Brakmo and L. Peterson, "TCP Vegas: End-to-end Congestion Avoidance on A Global Internet," *Journal on Selected Area in Communications*, vol. 13, no. 8, pp. 1465-1480, October 1995.
- [8] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," *Proceedings, ACM SIGCOMM*, Pittsburgh, PA, August 2002.
- [9] C. Qiao and M. Yoo, "Optical Burst Switching (OBS) - A New Paradigm for An Optical Internet," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69-84, January 1999.
- [10] Y. Xiong, M. Vanderhoute, and H. C. Cankaya, "Control Architecture in Optical Burst-switched WDM Networks," *IEEE Journal on Select. Areas in Communications.*, vol. 18, pp. 1838-1851, Oct. 2000.
- [11] N. M. Garcia, *et al.*, "Performance of Optical Burst Switched Networks for Grid Applications," *Proceedings, International Conference on Networking and Services (ICNS '07)*, Athens, Greece, June 2007.
- [12] G. Zervas, R. Nejabati, and D. Simeonidou, "Grid-empowered Optical Burst Switched Network: Architecture, Protocols," *Proceedings, International Conference on Networking and Services (ICNS '07)*, Athens, Greece, June 2007.
- [13] D. Simeonidou, *et al.*, "An Optical Network Infrastructure Suitable for Global Grid Computing," *Proceedings, TERENA Networking Conference*, Rhodes, Greece, June 2004.
- [14] X. Yu and C. Qiao and Y. Liu , "TCP Implementations and False Time Out Detection in OBS Networks," *Proceedings, IEEE Infocomm*, Hong Kong, China, March 2004.
- [15] Q. Zhang, V. Vokkarane, Y. Wang, and J. P. Jue, "Analysis of TCP over Optical Burst-Switched Networks with Burst Retransmission," *Proceedings, IEEE GLOBECOM*, St. Louis, MO, November 2005.
- [16] Q. Zhang, V. Vokkarane, Y. Wang, and J. P. Jue, "Evaluation of Burst Retransmission in Optical Burst-Switched Networks," *Proceedings, 2nd International Conference on Broadband Networks (BROADNETS)*, Boston, MA, Oct. 2005.
- [17] C. Hsu, T. Liu, and N. Huang, "Performance Analysis of Deflection Routing in Optical Burst-switched Networks," *Proceedings, IEEE Infocom*, New York, NY, June 2002.
- [18] J. Mo, R. La, V. Anantharam, and J. Walrand, "Analysis and Comparison of TCP Reno and Vegas," *Proceedings, IEEE Infocomm*, New York, NY, March 1999.
- [19] E. Weigle and W. Feng, "A Case for TCP Vegas in High-performance Computational Grids," *Proceedings, 10th IEEE International Symposium High Performance Distributed Computing*, San Francisco, CA, August 2001.
- [20] A. Detti and M. Listanti, "Impact of Segments Aggregation on TCP Reno Flows in Optical Burst Switching Networks," *Proceedings, IEEE Infocom*, New York, NY, June 2002.
- [21] C. Samios and M. K. Vernon, "Modeling the Throughput of TCP Vegas," *Proceedings, ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, San Diego, CA, June 2003.
- [22] 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.
- [23] C. Cameron, *et al.*, "TCP over OBS - Fixed-point Load and Loss," *Optics Express*, vol. 13, pp. 9167-9174, 2005.
- [24] M. Zukerman, E. Wong, Z. Rosberge, G. M. Lee, and H. L. Vu, "On Teletraffic Applications to OBS", *IEEE Communication Letters*, vol. 8, no. 1, 2004.
- [25] Q. Zhang, V. Vokkarane, J. Jue, and B. Chen, "Absolute QoS Differentiation in Optical Burst-Switched Networks," *IEEE Journal on Selected Areas in Communications, Optical Communications and Networking Series*, vol. 22, no. 9, pp. 1781-1795, Nov. 2004.