Threshold-based TCP Vegas over Optical Burst Switched Networks

Basem Shihada[1], Qiong Zhang[2], Pin-Han Ho[1]
[1]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada
{bshihada,pinhan}@bbcr.uwaterloo.ca
[2]Mathematical Science and Applied Computing, Arizona State University West Campus, Phoenix, USA
qiong.zhang@asu.edu

***Abstract* - Due to the bufferless nature of Optical Burst Switched network, contentions occur even at low traffic loads, leading to burst losses. Contention resolution schemes, such as burst retransmission and deflection, can reduce burst losses, especially at low traffic loads. However, both schemes result in additional packet delay for the packets in bursts that are retransmitted or deflected. The additional packet delay affects the performance of delay-based TCP implementations that rely on packet delay to estimate available bandwidth in networks and to detect network congestion state. In this paper, we discuss the issues of TCP Vegas over OBS networks and propose a threshold-based TCP Vegas version that is suitable for the characteristics of OBS networks. The threshold-based TCP Vegas are able to distinguish whether the increases in packet delay are due to network congestion, or due to burst contentions at low traffic loads. Our simulation results show that the threshold-based TCP Vegas has higher throughput for a TCP connection compared to TCP Vegas and the loss-based TCP implementations, such as TCP Sack.**

Key words: Optical burst Switching, TCP Vegas, Deflection, Burst Retransmission

I. INTRODUCTION

Wavelength division multiplexing (WDM) is a promising transmission technology for the next-generation Internet. In WDM technology, the optical transmission spectrum is carved up into a number of non-overlapping wavelength bands, and each wavelength supports a single communication channel operating at the peak electronic rate. Currently, WDM technology enables the multiplexing of 160-320 wavelengths into a single fiber, with a transmission rate of 10-40 Gb/s per wavelength. In order to efficiently utilize the raw bandwidth in WDM networks, an all-optical transport method, which supports fast resource provisioning and asynchronous transmission, must be developed. Optical burst switching (OBS) is such a promising WDM switching technology [1].

In OBS networks, data traffic is transmitted as bursts. Each burst consists of multiple packets (IP packets for an IP over WDM network), and a burst header packet (BHP) is generated for each burst. For a burst transmission, a BHP is first sent from the source node traveling thought the core nodes to set up the path based on predefined traffic parameters. After an offset time, the data burst is then sent through the network all-optically in a one-way signalling fashion. The offset time must ensure that, at each intermediate node, the BHP is processed prior to the burst arrival. Hence, there is no need to delay burst at each intermediate node.

Burst contention occurs when more than one burst attempts to traverse the same output port or wavelength at the same time. We refer to the burst that fails to make a successful wavelength reservation due to contention at a core node as the *contending burst*. Contentions result in bursts being dropped, leading to burst losses. Since OBS core networks are bufferless, OBS networks suffer from random burst losses due to burst contentions, even at low traffic loads. There are many contention resolution schemes that can reduce random burst losses in OBS networks, such as fiber delay line buffering [2], wavelength conversion [3], deflection [4], and burst retransmission [5].

TCP has been under tremendous amount of research and enhancements over the past decade. Many TCP versions have been proposed in order to improve TCP performance over networks with different network transmission characteristics, such as long propagation delay in long distance optical circuit switched (OCS) networks [6, 7, 8]. TCP versions can generally be classified into loss-based or delay-based according to how TCP senders detect congestion state in a network. The loss-based TCP implementations, such as TCP Reno, NewReno, and Sack, detect network congestion by packet losses. The delay-based TCP implementations, such as TCP Vegas [9] and Fast TCP [6], use the packet delay to estimate the available bandwidth in networks.

When loss-based TCP implementations run over OBS-based WDM networks, packet losses in OBS networks cannot correctly indicate network congestion since random burst losses occur even at low traffic loads. These random burst losses may be mistakenly interpreted by the TCP layer as congestion in the network, leading to serious degradation of the TCP performance. Several schemes have been proposed in [10, 11] to solve the false congestion detection issue for the loss-based TCP implementations over OBS networks.

The packet delay is known to be more accurate in congestion estimation than the packet loss events due to two reasons [6]. In high speed with large bandwidth-delay product, packet losses are very rare events. Thus, packet losses are not a proper indicator of network congestion. Furthermore, the packet delay provides multi-bit information, which will make

an equation-based rate control implementation easier to stabilize a network with a target fairness and high utilization compared to the one-bit loss information obtained by a packet loss.

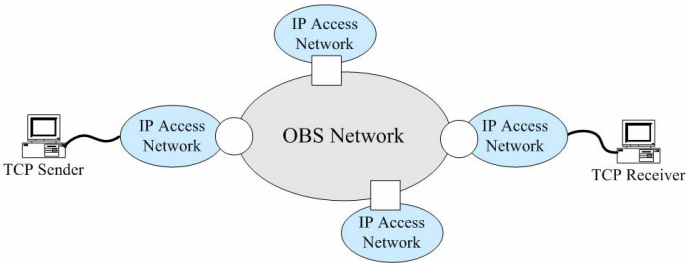

Figure 1. TCP over OBS network

The delay-based TCP Vegas implementation measures the round trip delay (*RTT*) of each packet in order to calculate the Expected throughput and the Actual throughput, and then adjusts its congestion window size (*cwnd*) based on the difference between the Expected throughput and the Actual throughput [12, 13]. By selecting proper α and β parameters, it has been shown that TCP Vegas performs well over modern high-performance links and better than TCP Reno [14, 15].

In this paper, we focus on investigating the performance of the delay-based TCP Vegas over OBS networks, and the network model is shown in Fig. 1. When TCP Vegas runs over typical barebone OBS networks, network congestion happens in two scenarios. One scenario is when packet-switched IP access networks are congested, which results in longer queuing delay. The other scenario is when OBS networks are heavily loaded, which results in high packet loss probability. Since the packet delay incurred in an OBS core network primarily consists of burst assembly and link propagation delay, the packet delay will not vary if the OBS network adopts a fixed source-routing scheme. Thus, the delay-based TCP Vegas can only estimate the congestion state in IP access networks that have electronic buffers. TCP Vegas will suffer false congestion detection when a random burst loss occurs due to contention in a lightly-loaded OBS network.

OBS networks employ contention resolution schemes in order to reduce random burst loss, thus improving the transmission reliability of OBS networks. Both deflection and burst retransmission schemes can recover burst losses, but introduce extra burst delay for bursts that experience contention. For the fiber delay line contention resolution scheme, the delay accommodated by fiber delay lines is very limited, e.g. to delay a single burst for 1 *ms* requires a fiber with length of 200 *km*. Hence, in this paper, we investigate the performance of delay-based TCP over OBS networks that employ deflection or burst retransmission scheme.

When TCP Vegas runs over OBS networks with burst retransmission or deflection scheme, TCP detects the increase in *RTT*s for packets in bursts that are deflected or retransmitted in OBS networks. However, TCP Vegas could not tell whether the increases in packet delay are due to network congestion or due to retransmission or deflection in lightly-loaded OBS networks. Hence, the delay-based TCP Vegas has a great challenge to accurately estimate the network congestion state in OBS networks.

In this paper, we propose a threshold-based TCP Vegas that is suitable for the characteristics of OBS networks with deflection or retransmission scheme. We introduce a threshold to distinguish between network congestion and contention at low traffic loads. If an OBS network is heavily loaded, random burst contention will occur frequently, which results in higher number of bursts being deflected or being retransmitted. Thus, if the number of packets that have increases in *RTT*s exceeds the threshold, then TCP detects network congestion in the OBS network. If an OBS network is lightly loaded, random burst contention will occur infrequently. Thus, if the number of packets that have increases in *RTT*s is below the threshold, then TCP considers the increases in *RTT*s are due to contentions at low traffic loads. Once TCP Vegas realizes the correct network state, it will correspondingly adjust the size of *cwnd*.

The rest of the paper is organized as follows. Section II first describes the burst retransmission and the deflection schemes in OBS networks, and then presents the implementation and issues of TCP Vegas in IP packet-switched networks. Section III discusses the issues of the delay-based TCP Vegas over OBS networks and presents the proposed threshold-based TCP Vegas. In Section IV, we compare the performance of the threshold-based TCP Vegas with the original TCP Vegas version and the loss-based TCP versions, such as TCP Sack. Section V concludes the paper.

II. TCP VEGAS

The delay-based TCP implementations use delay measurement to estimate available bandwidth in networks, including TCP Vegas [9] and Fast TCP [6]. It has been shown in [14, 15] that TCP Vegas improves TCP throughput by achieving *37%* to *71%* higher throughput and by significantly reducing packet retransmissions compared to TCP Reno. The performance of Fast TCP has been evaluated in [6, 16]. Fast TCP can be thought of as a high-speed of TCP Vegas [6]. In this paper, we focus on the delay-based TCP Vegas.

TCP Vegas modifies TCP Reno in the congestion avoidance, slow-start, and retransmission phases [2]. We describe the modifications of TCP Vegas as follows.

1) TCP Vegas Congestion Avoidance

TCP Reno uses packet losses as a signal for network congestion and can not detect any potential congestion before packet losses occur. On the other hand, TCP Vegas uses the difference between the estimated throughput and the measured throughput as a way of estimating the congestion state of the network.

TCP Vegas first computes the *BaseRTT* as the minimum measured *RTT* that is an estimation of the propagation delay as well as the queuing delay. Then Vegas computes the *Expected* throughput according to

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

where *cwnd* is the current congestion window size.

Second, Vegas calculates the current *Actual* throughput. For each packet being sent, Vegas estimates its *RTT* before its ACK comes back. Vegas then computes *Actual* throughput using the estimated *RTT* by

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

Vegas then compares *Actual* and *Expected* and computes the *Diff* as follows:

$$Diff = Expected - Actual, \quad where\ Diff > 0. \quad (3)$$

The *Diff* is used to adjust the next *cwnd*. Vegas defines two threshold values for controlling *Diff*, α and β. If *Diff* < α, then Vegas increases the window size linearly during the next *RTT*. If *Diff* > β, Vegas decreases the congestion window size linearly during the next *RTT*. Otherwise, Vegas leaves the window size unchanged. Hence, TCP Vegas congestion avoidance mechanism aims to maintain the expected number of outstanding packets in the queues of networks between α and β. If the *Actual* throughput is much smaller than the *Expected* throughput, then it is likely that the network is congested. Thus, the TCP sender should reduce the flow rate. On the other hand, if the *Actual* throughput is too close to the *Expected* throughput, then the connection may not be utilizing the available flow rate, and hence should increase the flow rate.

2) TCP Vegas Slow Start

TCP Vegas increases the *cwnd* exponentially only every other *RTT*. In between the two consecutive *RTT*s, the *cwnd* stays fixed in order to achieve the expected and actual transmission rates. When the actual rate falls below the expected rate, TCP Vegas changes from slow-start mode to linear increase/decrease mode.

3) TCP Vegas Packet 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 uses the estimated *RTT* to decide 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, then Vegas 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 time interval since the segment was sent is larger than the timeout value. If it is, then Vegas retransmits the segment. This will catch any other segment that may have been lost previous to the retransmission without having to wait for a duplicate *ACK*. Hence, Vegas retransmission mechanism reduces the time to detect lost segments from the third duplicate *ACK* to the first or second duplicate *ACK*.

4) TCP Vegas for Rerouting and Network Congestion Problem in Packet-Switched IP Networks

In a packet-switched IP network, the route of a TCP connection may be changed by routers, which is called rerouting. Rerouting a path may increase the propagation delay of the connection. Without an explicit signal from the router, TCP Vegas will not be able to tell whether the increase in the measured *RTT* is due to network congestion or a change in the route. If the delay increase is due to rerouting, then TCP Vegas must decide an accurate *BaseRTT* that estimates the propagation delay. A modified TCP Vegas has been proposed in [17] in order to detect accurate *BaseRTT* based on measured *RTT*s. The basic idea behind this modified TCP Vegas is as follows. If the minimum *RTT* computed for a number of packets is consistently much higher than *BaseRTT*, then it is likely that the actual propagation delay is larger than the measured *BaseRTT*, and then *BaseRTT* will be increased. Once *BaseRTT* increases, the *Diff* of *Actual* and *Expected* throughput starts to reduce as the congestion window size is properly reset and linearly increased.

In case where the delay is caused by network congestion, the increased *BaseRTT* is an inaccurate estimation of the propagation delay of the path, which creates a temporary increase in congestion level in the network. The RED gateways in the network will start to drop packets. Then, most connections will detect congestion and reduce their congestion window size. The congestion level will come down, which allows the connections to estimate a correct *BaseRTT*. Hence, if the connections are in persistent congestion, increasing *BaseRTT* will force the connection to break the persistent congestion state and update to a correct *BaseRTT*.

The above-mentioned modification can successfully assist TCP Vegas to correctly react to rerouting and network congestion in a packet-switched IP network. However, if TCP Vegas runs over IP over OBS-based WDM network, TCP Vegas still has the issue of false congestion detection due to contentions in a lightly-loaded OBS network. In the next section, we discuss the issues of TCP Vegas over OBS networks and propose a modified TCP Vegas version that is suitable for the characteristics of OBS networks.

III. TCP VEGAS OVER OBS NETWORK

In a typical barebone OBS network, if the OBS network adopts a fixed source-routing scheme, the packet delay experienced in the OBS network is primarily the sum of burst assembly delay and link propagation delay, which does not vary when the traffic load in the OBS network changes. Hence, the delay-based TCP Vegas cannot effectively detect network congestion in OBS networks. Furthermore, burst losses occur due to contentions even if the OBS network is lightly loaded. If all packets in the *cwnd* of TCP Vegas are assembled into a single burst, TCP Vegas suffers false congestion detection if the burst is contended and dropped at low traffic loads. TCP Vegas sender then triggers time out retransmission and enters into slow start phase, which significantly reduces the TCP throughput.

Both deflection and burst retransmission contention resolution schemes can significantly reduce burst losses, especially at low traffic loads. Hence, when TCP Vegas runs over OBS networks with burst retransmission or deflection scheme, the likelihood of false congestion detection in TCP Vegas will be reduced. However, both schemes suffer an extra delay for the contending bursts that successfully reach the destination node. When TCP Vegas runs over OBS networks with burst retransmission or deflection scheme, TCP detects the increases in *RTT*s for packets in bursts that are deflected or retransmitted in OBS networks, which may result in TCP Vegas reducing its *cwnd* size. If more packets from a TCP connection are assembled into a burst, the size of *cwnd* may be further reduced, leading to lower TCP throughput. But if the increases in *RTT*s are caused by burst retransmission or deflection in a lightly-loaded OBS network, TCP Vegas should not reduce its *cwnd*. Hence, we need to have a modified TCP Vegas that are able to tell whether the increases in *RTT*s are due to network congestion, or due to retransmission or deflection in a lightly-loaded OBS networks.

We observe that if the IP access network is congested, or TCP Vegas will continuously detect the increases in *RTT*s. If the OBS network is heavily loaded, random burst contentions frequently occur, then TCP Vegas will often detect the increases in *RTT*s. If the OBS network is lightly loaded, random burst contentions less frequently occur, and at the same time, if the packet-switched IP access network is not congested, TCP Vegas will less often detect the increases in *RTT*s. Hence, TCP Vegas can detect network congestion (in OBS networks and/or in IP access network) if *RTT*s are frequently increased.

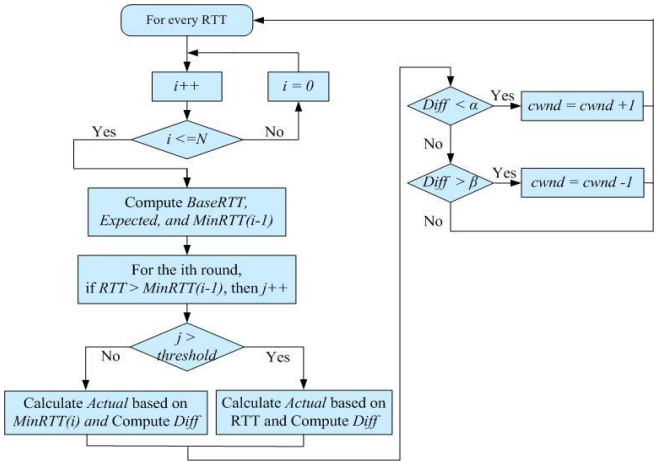

Figure 2. Flow chart for the threshold-based TCP Vegas

Based on the above observations, we propose a threshold-based TCP Vegas. We introduce a threshold T to assist TCP Vegas to distinguish between network congestion and contentions at low traffic loads. TCP Vegas measures RTT for each packet sent and keeps track of the minimum measured *RTT*s of N consecutive packets. Let $MinRTT(i)$ be the minimum measured *RTT*s of i $(0 < i < N)$ consecutive packets. For the ith round, if the measured *RTT* of the ith packets is larger than $MinRTT(i-1)$, it means that the ith packet is in the burst that is deflected or retransmitted or the ith packet is queued in access networks. A counter that measures the number of packets whose *RTTs* are larger than their $MinRTT(i-1)$ will then be increased by 1. If the number of TCP packets whose *RTTs* are larger than their $MinRTT(i-1)$ is under the threshold T, then the TCP Vegas connection will stay with the *Actual* throughput calculated based on their $MinRTT(i-1)$, even if the measured *RTT*s are increased. If the number of TCP packets whose *RTT*s are larger than their $MinRTT(i-1)$ exceeds the threshold T, which means that bursts are often deflected or retransmitted in the OBS network. Hence the TCP Vegas recognizes the network congestion and calculates the *Actual* throughput as usual. Fig. 2 summarizes the proposed threshold-based TCP Vegas.

The basic idea behind the threshold-based TCP Vegas is to reduce the sensitivity of TCP Vegas to the increases in packet delay caused by retransmission and deflection in order to more accurately react to the congestion state in OBS networks. To decrease the parameter α, or to increase the parameter β may also reduce the sensitivity of TCP Vegas by keeping the *cwnd* unchanged. However, it makes TCP Vegas difficult to estimate the available bandwidth in networks.

Note that the number of packets that are sent consecutively, N, and the threshold, T, should be chosen much larger than the number of packets from a TCP Vegas connection that are assembled into a burst, so that TCP Vegas is able to detect the frequency of deflection and retransmission in OBS networks based on a number of bursts. Usually, the packets from a TCP connection assembled in the same burst have the same measured RTT. By analyzing the variation pattern of packet RTTs, TCP Vegas can obtain the number of packets from a TCP Vegas connection that are assembled into a burst. Also, the relationship between the threshold T and the number of consecutive packets, N, affects the TCP performance. When T is closer to N, once a congestion state is detected, there are less number of remaining packets in N to react to the detected congestion, which results in TCP ineffectively reacting to network congestion. Hence, we choose N to be $N = iT$, where $i > 1$. In Section V, we evaluate different values of T and N.

IV. Numerical Results

To verify the proposed scheme, simulation is conducted using Network Simulator *NS-2*, where the NSF network topology shown in Fig. 3 is adopted as the OBS network topology. The distances shown are in km. On each link, there is a bi-directional control channel and a fiber link used for data burst transfer. Each fiber link consists of 8 wavelengths operating at *10* Gb/s transmission rate. Each OBS core node is equipped with fiber delay lines operating at *2*, *5*, and *8 ms* delay granularities. The mixed time/length based burst assembly algorithm is adopted, where the burst timeout threshold is set

to *5 ms* and the maximum burst length is set to *500KB*. The core nodes implement the *LAUC-VF* channel scheduling algorithm, where the burst offset time is set to *2μs*. The burst retransmission and deflection schemes are implemented in the OBS network. In the burst retransmission scheme, bursts can be retransmitted once and are dropped after the second contention. In the deflection scheme, bursts can be deflected once.

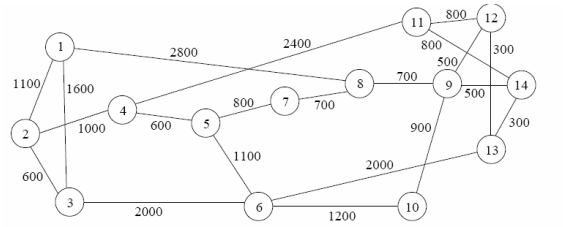

Figure 3. NSF network topology

The TCP senders and receivers are attached to OBS edge nodes. In the simulation, the packet delay variation is caused by the burst retransmission or burst deflection. A File Transfer Protocol (FTP) application is used for generating TCP traffic with average packet size of *1KB*. The TCP packets traverse through a minimum of four hops before reaching their destination. The TCP throughput is obtained over a simulation period of 10^6 seconds. The average number of packets assembled in one burst ranges from 100 to 500 packets. In the following figures, the TCP Vegas with $T \neq 0, N \neq 0$ denotes the threshold-based TCP Vegas. Also, in the figures, the throughput is measured for a single TCP flow in the network, and the burst contention probability is the probability that a burst experiences contention, but is not necessarily dropped.

Fig. 4 compares the throughput of loss-based TCP Sack, the original Vegas version, and the threshold-based Vegas over the barebone OBS network. In the threshold-based TCP Vegas, the threshold T is chosen to be *100, 200, 300*, and *400* while the number of consecutive TCP packets N is chosen to be *4T*. The threshold-based TCP Vegas with different N and T and original TCP Vegas perform very similar such that their throughput plots are overlapped. This is due to the fact that the round trip time does not vary significantly. We can also see that TCP Vegas versions perform much better than the loss-based TCP versions in the barebone OBS network.

Fig. 5 and Fig. 6 compares the throughput of loss-based TCP Sack, the original Vegas version, and the threshold-based Vegas over the OBS network with burst retransmission and burst deflection, respectively. We observe that the threshold-based TCP Vegas performs much better than the original TCP Vegas and TCP Sack when the OBS network applies either burst retransmission or burst deflection. For example, in Fig.5, when the burst contention probability is 10^{-4}, TCP Vegas with the threshold *400* improves the throughput by *73%* compared to the original TCP Vegas. We can also see that, with higher threshold, the threshold-based TCP Vegas performs better. When the burst contention probability is 10^{-4}, TCP Vegas with the threshold *400* improves the throughput by *82%* compared to TCP Vegas with threshold *200*. This is because that the TCP Vegas with threshold *400* more accurately detects the congestion state in the OBS network than the TCP Vegas with threshold *200* and delays triggering Vegas congestion avoidance mechanism.

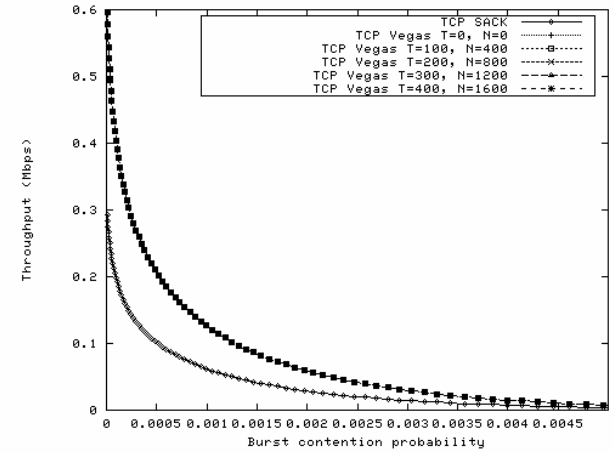

Figure 4. Throughput of TCP Sack, Vegas, and threshold-based Vegas vs. burst contention probability in barebone OBS.

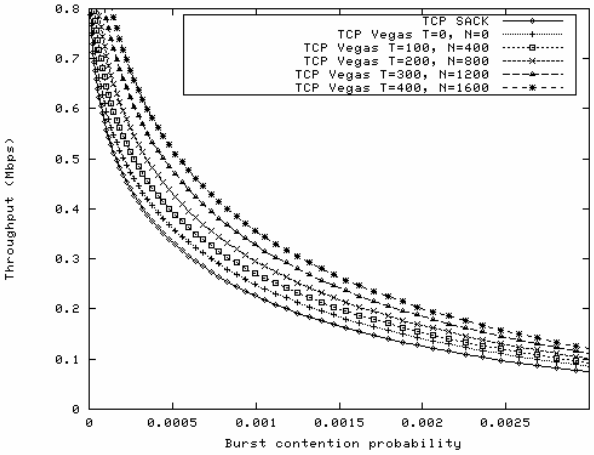

Figure 5. Throughput of TCP Sack, Vegas, and threshold-based Vegas vs. burst contention probability in OBS with burst retransmission.

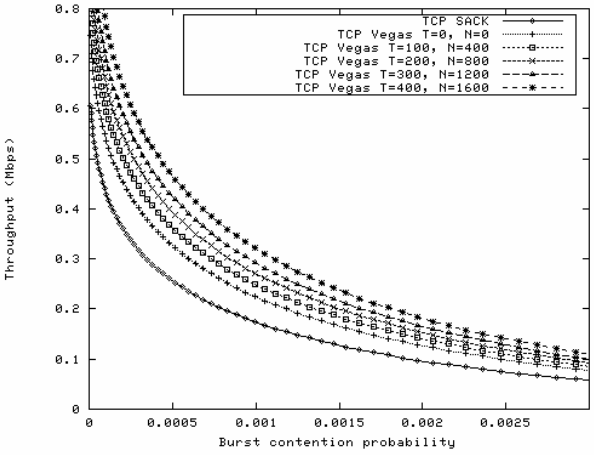

Figure 6. Throughput of TCP Sack, Vegas, and threshold-based Vegas vs. burst contention probability in OBS with burst deflection.

Fig. 7 and Fig. 8 compare the throughput of TCP Sack and the threshold-based Vegas over the OBS network with burst retransmission and deflection, respectively. In these experiments, we show the effect of varying the values of *T* and *N* in the threshold-based Vegas. We observe that, with a fixed *T* value, varying *N* values does not result into a major throughput change. For example, the throughput of $N = 400$ and $N = 800$ are very close when $T = 200$. We can also see that Vegas throughput is mainly affected by varying the values of *T*. For instance, when $N = 400$, the throughout of $T = 200$ increases *86%* compared to the throughput of $T = 100$.

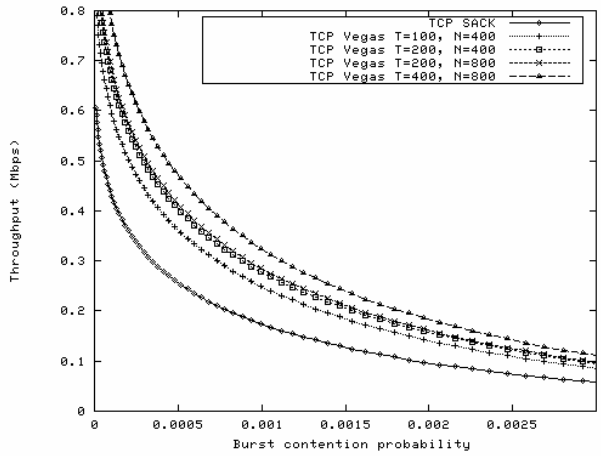

Figure 7. Throughput of TCP Sack and threshold-based Vegas vs. burst contention probability in OBS with burst retransmission.

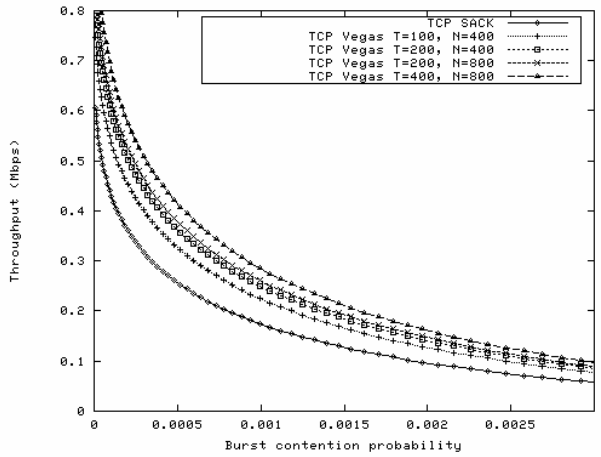

Figure 8. Throughput of TCP Sack and threshold-based Vegas vs. burst contention probability in OBS with burst deflection.

V. CONCLUSIONS

In this paper, we discussed the issues of TCP Vegas over OBS networks due to the misinterpretation of the additional delay introduced by the burst retransmission and burst deflection schemes. Based on the TCP packet delay variation pattern in an OBS network with burst retransmission or deflection scheme, we proposed a threshold-based TCP Vegas that is able to distinguish whether the increases in TCP packet *RTT*s are due to network congestion or due to retransmission or deflection in a lightly-loaded OBS networks. Our simulation results showed that the threshold-based TCP Vegas can more accurately detect the congestion state in OBS networks with burst retransmission or deflection, thus improves the throughput compared to the original TCP Vegas version and the loss-based TCP Sack.

Our proposed threshold-based TCP Vegas has a single threshold to measure the congestion state in the OBS network. Our future work can be extended to multiple thresholds, which have more granualarity for the congestion state.

REFERENCES

[1] 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.

[2] I. Chlamtac, A. Fumagalli, L. G. Kazovsky, and et al., "CORD: Contention resolution by delay lines," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 1014–1029, June 1996.

[3] S.J.B. Yoo, "Wavelength conversion technologies for WDM network applications," *IEEE/OSA Journal of Lightwave Technology*, vol. 14, pp. 955–966, June 1996.

[4] C. Hsu, T. Liu, and N. Huang, "Performance analysis of deflection routing in optical burst-switched networks," *Proceedings, IEEE Infocom*, vol. 1, pp. 66–73, 2002.

[5] 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) 2005*, Boston, MA, Oct. 2005.

[6] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *Proceedings, IEEE Infocom*, 2004.

[7] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks", *Proceedings, IEEE Infocom*, Hong Kong, China, 2004.

[8] S. Floyd, "Quick-Start for TCP and IP," *Internet draft, draft-amit-quick-start-02.txt*, 2002.

[9] L. Brakmo and L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global Internet," *Journal on Selected Area*, vol. 13, no. 8, pp. 1465-1480, 1995.

[10] X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false time out detection in OBS networks," *Proceedings, IEEE Infocom*, March 2004.

[11] Q. Zhang, V. Vokkarane, Y. Wang, and J. P. Jue, "Analysis of TCP over Optical Burst-Switched Networks with Burst Retransmission," *Proceedings, IEEE Global Communication Conference (GLOBECOM)*, St. Louis, 2005.

[12] L. Brakmo and L. Peterson, "TCP Vegas: end-to-end congestion avoidance on a global Internet," *Journal on Selected Area*, vol. 13, no. 8, pp. 1465-1480, October 1995.

[13] J. Mo, R. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," *Proceedings, IEEE Infocomm*, pp. 1556-63, March 1999.

[14] E. Weigle and W. Feng, "A case for TCP Vegas in high-performance computational grids," *Proceedings, 10th IEEE International Symposium High Performance Distributed Computing*, pp. 158–167, 2001.

[15] J. Ahn, P. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: emulation and experiment," *Computer Communication Review*, vol. 25, pp 185-95, 1995.

[16] S. Hegde, D. Lapsey, and et. al., "FAST TCP in high-speed networks: An experimental study," *Workshop on Networks for Grid Applications*, October 2004.

[17] R. La, J. Walrand and V. Anantharam, "Issues in TCP Vegas," *UCB/ERL Memorandum*, No. M99/3, Electronics Research Laboratory, University of California, Berkeley, 1999.