

Performance Evaluation of TCP Vegas over Optical Burst Switched Networks

Basem Shihada¹, Qiong Zhang³, Pin-Han Ho^{1,2},

¹School of Computer Science, University of Waterloo, Waterloo, Canada

²Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada

bshihada.pinhan@bcr.uwaterloo.ca

³Mathematical Science and Applied Computing, Arizona State University West Campus, Phoenix, USA

qiong.zhang@asu.edu

Abstract - Burst Dropping in Optical Burst Switched (OBS) networks can occur due to random contention and congestion. Burst retransmission and deflection schemes in the OBS domain is expected to improve the throughput of TCP flows by hiding burst loss events from the upper TCP layer in order to mitigate the congestion window fluctuation of TCP senders. However, burst retransmission and deflection schemes introduce additional delays which negatively impact the performance of the delay-based TCP versions, such as TCP Vegas. Additional delays could result in delay-based TCP to falsely detect network congestion. In this paper we investigate the delay-based TCP Vegas behavior over OBS networks. We also analyze the throughput of TCP Vegas over OBS networks with burst retransmission.

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

I INTRODUCTION

TCP has been under tremendous amount of research and enhancements over the past decade. A large number of proposed solutions have suggested modifying TCP such that it adapted to the new network transmission characteristics [1-9]. However, these modifications have assumed the best-effort, buffer-oriented routing mechanism, which has been adapted in the current Internet routing architecture. TCP congestion control mechanisms can generally be classified into three categories: loss-based, delay-based, and explicit notification-based. This study targets the delay-based TCP Vegas implementation. The delay-based TCP implementations such as Vegas rely on the queuing delay to estimate the network congestion [5]. Using a proper congestion control parameters, it has been shown that TCP Vegas improves the TCP efficiency by achieving 37% to 71% higher throughput and by significantly reducing packet retransmissions compared to TCP Reno [6].

Optical Burst Switching (OBS) tends to be a bufferless network [10]. In OBS networks, data bursts are cuts-through networks all-optically following their control packet using a one-way signaling control protocol. Data bursts do not incur any buffering delay. On the other hand, OBS networks suffer from random burst contentions. Contention occurs when more than one burst attempts to traverse the same wavelength on a output port at the same time. Random contentions may happen even when OBS networks are not congested. Burst contentions

result in burst losses in OBS networks. From the TCP prospective, burst losses due to random contentions may cause TCP senders falsely identifying network congestion situation, which is referred to as TCP false congestion detection [11].

In OBS networks, it has been shown in [12], [11], and [13] that, deflection and burst retransmission schemes can significantly reduce burst losses, especially at low traffic loads. In both schemes, contended bursts are able to be retransmitted or deflected to another route, instead of being immediately dropped. Hence, burst contention probability, which is the probability of a burst being contended in OBS networks, is different from the burst loss probability.

In this paper we investigate the behavior of the delay-based TCP Vegas over bufferless OBS networks, and highlight the impacts of extra delay introduced by burst retransmission and deflection on TCP Vegas. We derive an analytical model for the throughput of TCP Vegas over barebone OBS networks and over OBS networks with burst retransmission. The analytical model is further verified by simulation.

II BACKGROUND

The delay-based TCP implementations use delay measurement to estimate available bandwidth in networks, including TCP Vegas [5] and Fast TCP [2]. It has been shown in [7, 8] 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 [2, 3]. Fast TCP can be thought of as a high-speed of TCP Vegas [5]. 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. We now describe the modifications of TCP Vegas based on TCP Reno as follows.

A 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 every *RTT* time, Vegas computes *Actual* throughput using the most recent measured *RTT* by

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

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

$$\begin{aligned} Diff &= Expected - Actual, \quad \text{where } Diff > 0 \\ &= \left(\frac{W}{BaseRTT} - \frac{W}{RTT} \right) BaseRTT \\ &= W \left(1 - \frac{BaseRTT}{RTT} \right). \end{aligned} \quad (3)$$

The *Diff* is used to adjust the next *cwnd*. Vegas defines two threshold values for controlling *Diff*, α and β . The congestion window behavior is summarized as follows:

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

If $Diff < \alpha$, then Vegas increases the window size linearly during the next *RTT*. If $Diff > \beta$, 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.

B TCP Vegas Slow Start

In TCP Reno, at the slow start phase, the *cwnd* is additively increased at each successfully transferred or acknowledged packet. The linear growth of *cwnd* continues until either it exceeds the receiver's advertised *cwnd* or exceeds the initial threshold or a packet loss is reported. If the initial window threshold value is too small, the exponential increase will stop too early, leading to low throughput. On the other hand, if the threshold window is set too large, the congestion window will grow until the available bandwidth is exceeded, resulting in more packet losses.

TCP Vegas increases the *cwnd* exponentially only every

other *RTT* during slow start phase, and exits slow start if *diff* exceeds a threshold γ . In between the two consecutive *RTTs*, the *cwnd* stays fixed in order to make a valid comparison of the *Expected* and *Actual* throughput.

C TCP Vegas Packet Retransmission

TCP Reno sets *cwnd* to one packet and enters into the slow-start phase when a timeout (TO) is detected. TCP Vegas retransmission mechanism aims to reduce both the occurrence of TOs and the time taken for fast retransmission in loss-based TCPs.

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.

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 not congested. 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 to moderate traffic loads. TCP Vegas sender then triggers time out retransmission and enters into slow start phase, which significantly reduces the TCP throughput.

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 *RTTs* for packets in bursts that are deflected or retransmitted in OBS networks, which may result in TCP Vegas reducing its *cwnd* size, leading to lower TCP throughput. But if the increases in *RTTs* are caused by burst retransmission or deflection in a lightly-loaded OBS network, TCP Vegas should not reduce its

cwnd. In the next section, we investigate the impact of burst retransmission in OBS network on the throughput of TCP Vegas and develop an analytical model for the throughput of TCP Vegas over OBS with burst retransmission.

III TCP VEGAS PERFORMANCE MODELING OVER OBS

In this section, we analyze the throughput of TCP Vegas over OBS networks. We assume that the access bandwidth of a TCP flow is high, such that all TCP packets in a single *cwnd* are assembled to a burst. Such TCP flow is referred to as TCP fast flow [14]. Since a burst loss will result in TCP packets in an entire *cwnd* being lost, TCP fast sources will never trigger triple duplicates. In the analysis, in order to make the analysis tractable, we assume that IP access networks have no congestion and the burst assembly delay is ignorable. We also assume that burst dropping probability and burst contention probability in OBS networks are given.

We first analyze the throughput of TCP Vegas over barebone OBS networks. We then analyze the throughput of TCP Vegas over OBS networks with burst retransmission. The following table lists the notations used in the analytical models.

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 round trip time without burst retransmission
RTT_r	: Expected round trip time with burst retransmission
B	: Vegas throughput
H	: Expected number of packets submitted during timeout (TO)
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 timeout expiration
W_{max}	: Maximum congestion window size in packets
W_0	: Cwnd size in Vegas stable state

In our TCP Vegas models, we define a round as when Vegas emits the current *cwnd* (in segments) and wait until either it receives an acknowledgement or the RTO expires. We define a TO loss as a TCP packet loss that triggers TCP timeout at the end of a round. Since fast TCP flows does not trigger triple

duplicates, multiple successfully sending rounds are followed by one or more loss rounds. Therefore, we obtain the throughput of TCP Vegas fast flows as follows,

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

A Fast TCP Vegas over Barebone OBS

In this section, we analyze Vegas throughput over barebone OBS. In barebone OBS, a contended burst results into an immediate burst loss. Since we assume that IP access network is not congested, the expected round trip time over barebone OBS networks, R , is very close to $BaseRTT$.

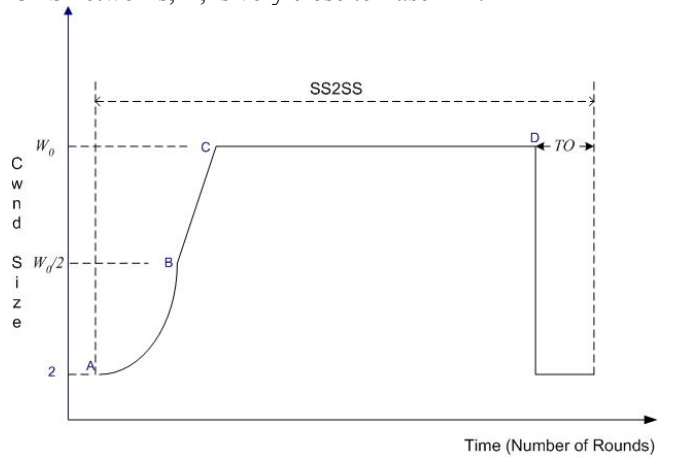


Fig. 1. Evolution of TCP windows size for TCP Vegas over barebone OBS networks.

Fig. 1 shows the evolution of TCP windows size for TCP Vegas over barebone OBS networks. The duration from A to B is the slow start phase in which *cwnd* starts at two and doubles every other round until it reaches the expected slow start threshold, $\frac{W_0}{2}$. The duration from B to C is a transition period, where the *Diff* is calculated based on Eq. (3). Since R is very close to $BaseRTT$, *Diff* will be a very small value less than α . Hence, during B to C, the window size increases by one each round until $\alpha \leq Diff \leq \beta$. When $\alpha \leq Diff \leq \beta$, the window size will remain unchanged in future rounds since both W and R will not change in subsequent rounds, thereby we call Vegas reaches stable state. The window size will remain constant until a timeout occurs. The duration from slow start to timeout that is the slow start of another sequence of rounds is called slow-start-to-slow-start (SS2SS) period.

The value of the window size during C to D is W_0 and can be calculated as follows:

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

By reversing Eq. (6), we can obtain the range of W_0 to be $\frac{\alpha R}{R - BaseRTT} \leq W_0 \leq \frac{\beta R}{R - BaseRTT}$. Based on [15], the expected value of $Diff$ is estimated as $\frac{\alpha + \beta}{2}$, which yields

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

When the timer expires for a packet, TCP Vegas remains for RTO idling, and then sets the $cwnd$ to two segments and returns to the slow-start. RTO equals as two times the smoothed RTT average plus four times the RTT variance. During the TO, no packets are transmitted, after a TO event Vegas emits only one segment. Considering Fig. 1, we can obtain the throughput by computing the expected number of packets transmitted in the period from A to D.

In the slow-start period illustrated in Fig. 1 between points A and B. The window doubles every other round until it reaches the slow-start threshold which is on average equal to $W_0/2$, since the expected window size when time outs occur is W_0 . Hence, at the slow-start phase, the window starts at two segments and doubles every other RTT until it reaches the slow start threshold, thus as per [15] we obtain Y_{AB} as follows,

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

where the duration of the slow-start phase is,

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

We then calculate the average number of packets transmitted during B to C denoted by Y_{BC} and the duration of that period denoted as A_{BC} . Y_{BC} can be obtained by summing the expected number of packets transmitted from $W_0/2$ until 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 $cwnd$ size increases by one for each round during B to D, the duration of this period is,

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

Based on Eq. (18) in [14], we have $E[X] = \frac{1-p}{p}$. The number of rounds during C and D can be calculated as (the number of successful rounds – the number of rounds during A and B – the number of rounds during B and C). Hence, the duration between C and D can be obtained as

$$A_{CD} = \left(\frac{1-p}{p} - 2(\log W_0 - 2) - \frac{W_0}{2}\right)R. \quad (12)$$

Since during C and D, the $cwnd$ size is equal to W_0 , hence the number of packets sent during C and D is

$$Y_{CD} = W_0\left(\frac{1-p}{p} - 2(\log W_0 - 2) - \frac{W_0}{2}\right) \quad (13)$$

Since the timeout process of TCP Vegas is similar to TCP Reno, hence, based on Equations (14) and (16) in [14], we have the mean duration for successive timeouts as

$$E[TO] = RTO \frac{f(p)}{1-p}, \quad (14)$$

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

$$E[H] = \frac{p}{1-p}. \quad (15)$$

We then have the average number of packets sent during the successful rounds, $E[Y] = Y_{AB} + Y_{BC} + Y_{CD}$, and the duration of successful rounds, $E[A] = A_{AB} + A_{BC} + A_{CD}$. By substituting Equations (7) through (15) in Eq. (5), we obtain Vegas throughput over barebone OBS networks as follows,

$$B_1 = \frac{Y_{AB} + Y_{BC} + Y_{CD} + E[H]}{A_{AB} + A_{BC} + A_{CD} + TO}. \quad (16)$$

B Fast TCP Vegas over OBS with Burst Retransmission

In this section, we analyze Vegas throughput over OBS networks with burst retransmission. Burst retransmissions improve burst loss probability by retransmitting contented bursts. However, it causes longer round trip time for the retransmitted bursts. We assume that retransmitted bursts that successfully reach their destinations experience an average round trip delay, RTT_r .

We further identify two types of successful rounds as follows: the rounds in which a burst experience contention but successfully retransmitted and the rounds that do not experience burst contention. We obtain the probability of a successful round in which a burst experiences contention but is successfully retransmitted as:

$$p_{sr} = \frac{p_c - p}{1-p}. \quad (17)$$

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

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

Fig. 2 shows the evolution of TCP windows size for TCP Vegas over OBS networks with burst retransmission. In this paper, we assume that burst contentions will not happen between point A and point C. At point C, the $Diff$ is calculated as

$$Diff_C = W_0\left(1 - \frac{BaseRTT}{R}\right) \geq \alpha. \quad (19)$$

During C to G, consider a round that experiences contention and has the current $cwnd$ size equal to W_0 (see point D in Fig. 2). The $Diff$ at point D is calculated as

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

If $\alpha \leq Diff_D \leq \beta$, then TCP Vegas will remain $cwnd$ unchanged during C to G even multiple burst contentions happen. Hence, by reversing Eq. (22), we can obtain the range of RTT_r , that remains $cwnd$ constant, that is,

$$\frac{BaseRTT}{1 - \frac{\alpha}{W_0}} \leq RTT_r \leq \frac{BaseRTT}{1 - \frac{\beta}{W_0}}.$$

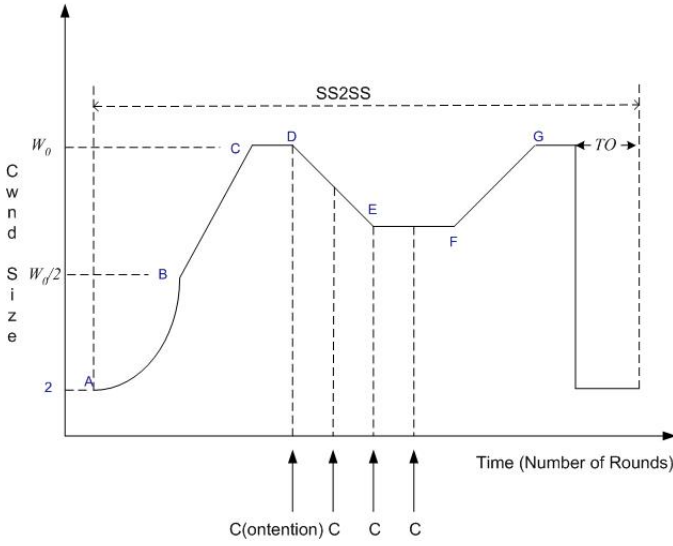


Fig. 2. Evolution of TCP windows size for TCP Vegas over OBS networks with burst retransmission.

The case when $Diff_D < \alpha$ will never happen for rounds that experience contention and have the current $cwnd$ size, W , equal to W_0 . This is because that $RTT_r \geq R$, which results in $Diff_D \geq Diff_C \geq \alpha$. Therefore, during a sequence of successful rounds, the $cwnd$ size will never exceed W_0 when $W = W_0$.

If $Diff_D > \beta$, then the $cwnd$ decreases size by 1. If multiple consecutive rounds experiences contention, then $Diff$ will keep decreasing for each of the consecutive rounds due to the decreasing of the $cwnd$ size, W , until $Diff$ reaches β , then the $cwnd$ will remain unchanged for future consecutive rounds that experience contention (see the duration from E to F in Fig. 2). Let W_0' be the lower bound of the $cwnd$ size during C to G. We can calculate the $Diff$ at E where the $cwnd$ remains unchanged for consecutive contented rounds as

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

Then we have

$$W_0' = \left\lceil \frac{\beta RTT_r}{RTT_r - BaseRTT} \right\rceil. \quad (21)$$

During D to G, the $Diff$ of any non-contented round is calculated as

$$Diff_F = W \left(1 - \frac{BaseRTT}{R}\right), \text{ where } W < W_0.$$

Note that $Diff_F$ is same as the $Diff$ calculated during B to C. Hence, any non-contented round during D and G will result in the $cwnd$ increasing by one unless the $cwnd$ size is equal to W_0 (see the duration from F to G in Fig. 2).

We conclude that the $cwnd$ size, W , changes between W_0 to W_0' . For each round, if $W_0' \leq W < W_0$, W either increases by one if the round does not experience contention, or decreases by one if the round experiences contention. If $W = W_0'$, W remains unchanged if the round experiences contention, and increases by one if the round does not experience contention. If $W = W_0$, W decreases by one if the round experiences contention, and remain unchanged if the round does not experience contention.

From the above observation, we model the state of W as a Markov Chain as shown in Fig. 3.

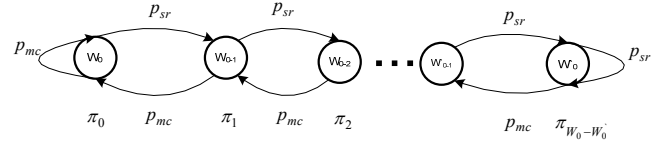


Fig. 3. Markov Chain for the state of W .

Let π_i be the probability of $W = W_0 - i$. We can solve π_i by local balance as follows:

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

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

...

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

Since $\sum_{i=0}^{W_0-W_0'} \left(\frac{P_{sr}}{P_{nc}}\right)^i \pi_0 = 1$, we obtain

$$\pi_0 = \frac{1}{\sum_{i=0}^{W_0-W_0'} \left(\frac{P_{sr}}{P_{nc}}\right)^i} = \begin{cases} \frac{1}{W_0 - W_0' + 1}, & P_{sr} = P_{nc} \\ \frac{1 - \left(\frac{P_{sr}}{P_{nc}}\right)^{W_0 - W_0' + 1}}{1 - \frac{P_{sr}}{P_{nc}}}, & P_{sr} \neq P_{nc}. \end{cases} \quad (22)$$

Then, we can obtain the expected $cwnd$ size during C to G as

$$E[W_{CG}] = W_0\pi_0 + (W_0 - 1)\pi_1 + \dots + W_0\pi_{(W_0 - W_0)}$$

$$= \sum_{i=0}^{W_0 - W_0'} [(W_0 - i)\pi_i].$$

For the case when $\frac{BaseRTT}{1 - \frac{\alpha}{W_0}} \leq RTT_r \leq \frac{BaseRTT}{1 - \frac{\beta}{W_0}}$, TCP Vegas

remains *cwnd* unchanged during C to G. Hence, $E[W_{CG}] = W_0$. Then, we have

$$Y_{CG} = E[W_{CG}] \left(\frac{1-p}{p} - 2(\log W_0 - 2) - \frac{W_0}{2} \right). \quad (23)$$

From Equations (8), (10), and (23), we obtain $E[Y]$ as

$$E[Y] = Y_{AB} + Y_{BC} + Y_{CG}$$

$$= (2^{\log W_0} - 4) + \left(\frac{3W_0^2}{8} - \frac{W_0}{4} \right) \quad (24)$$

$$+ E[W_{CG}] \left(\frac{1-p}{p} - 2(\log W_0 - 2) - \frac{W_0}{2} \right).$$

The expected duration between C to G can be calculated as

$$A_{CG} = p_{nc}E[X]R + p_{sr}E[X]RTT_r, \quad (25)$$

where $E[X] = \frac{1-p}{p}$. We then have

$$E[A] = A_{AB} + A_{BC} + A_{CG}. \quad (26)$$

We obtain TCP Vegas throughput over OBS networks with burst retransmission by substituting Equations (14), (15), (24), and (26) in Eq. (5):

$$B_2 = \frac{Y_{AB} + Y_{BC} + Y_{CG} + E[H]}{A_{AB} + A_{BC} + A_{CG} + TOP}. \quad (27)$$

IV NUMERICAL RESULTS

In this section we present the numerical results obtained from both the developed analytical model and simulation by using a NS-2 simulation. In our simulation, the NSF network topology shown in Fig. 4 is adopted as the OBS backbone, where the distance (in km) of each link is shown as the attached number. 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 fiber link consists of 8 wavelengths 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 5 ms, and the maximum burst length is set to 400KB. The control header processing time is set to be 1μsec. The core nodes implement the *LAUC-VF* channel scheduling algorithm. The burst retransmission mechanism is implemented at the OBS edge nodes. The contented bursts are allowed to be retransmitted only once in order not to exceed the time-out threshold.

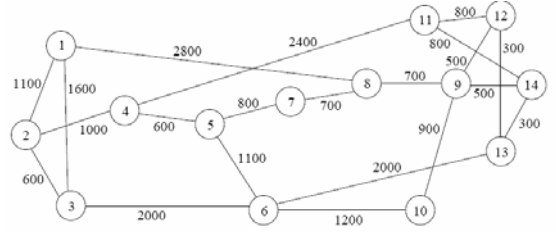


Fig. 4. NSF network topology adopted in the simulation.

The TCP senders and receivers are attached to the OBS edge nodes. In the simulation, the packet delay in the access network is set to be constant, such that the effect of the burst retransmission delay is the only source of enlarging the TCP round trip time of each packet. A File Transfer Protocol (FTP) application is initiated to generate TCP segments with an average size of 1KB. TCP Vegas congestion avoidance parameters are set to $\alpha = 2$ and $\beta = 5$ for all experiments.

We first fill the network with moderate load of traffic between each pair of nodes. Each TCP segment is launched at edge node 1 and is assembled into the very next burst starting at node 1 for edge node 14. In order to simulate the random burst contention phenomena, different random burst contention probabilities are inserted at the burst scheduling phase. In order to simulate the fast TCP flows, a high access bandwidth is allocated to each flow. TCP throughput is obtained over a simulation period of 10^6 seconds.

A Vegas Throughput in Barebone OBS Networks

This experiment studies TCP Vegas throughput in the existence of different burst contention probabilities. The average *RTT* in the simulation is approximately 100ms. The random burst contention probability p_c ranges in $[10^{-5}, 10^{-2}]$. For barebone OBS networks, burst contention probability is equal to burst loss probability. Fig. 5 compares the results of TCP Vegas throughput from the model obtained from Eq. (16) and the simulation under different p_c . We can see that the simulation results and the analytical results are very close.

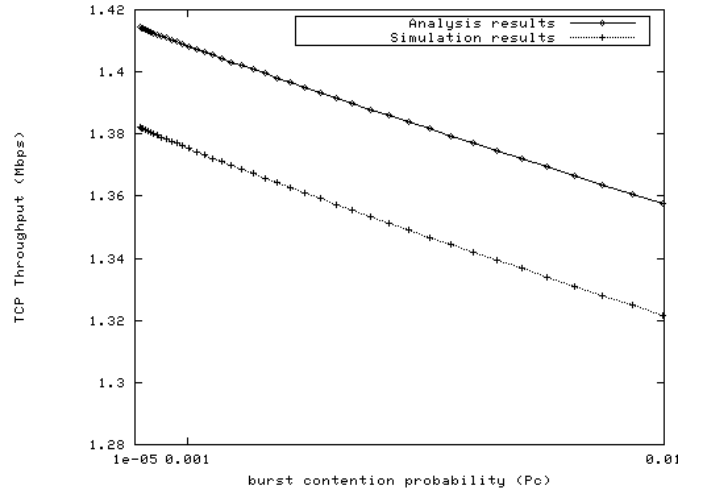


Fig. 5. Vegas throughput while varying the burst contention probability under barebone OBS

B Vegas Throughput in OBS with Burst Retransmission

In this 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. Fig. 6 compares the analytical results obtained by Eq. (27) and the Vegas fast flow obtained through the simulation. We see that the simulation results matches the analytical results.

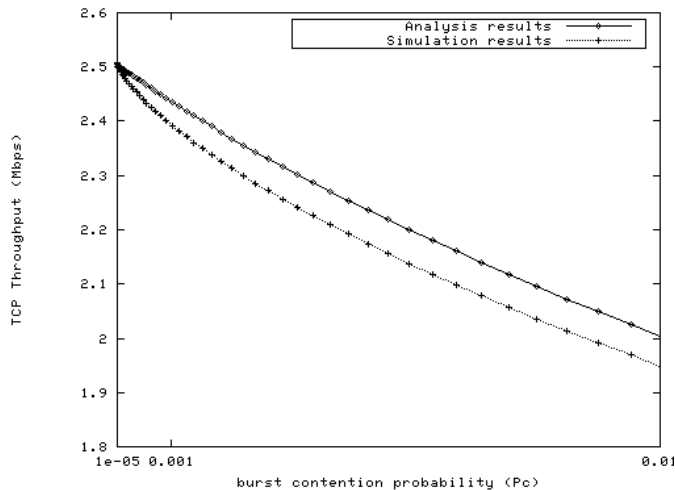


Fig. 6. Vegas throughput while varying the burst contention probability in the case of OBS with burst retransmission.

V CONCLUSION

TCP Vegas serves as one of the well known delay-based TCP implementations. This paper analyzes the throughput performance of TCP Vegas over OBS networks, where we have developed an analytical model to obtain fast TCP Vegas throughput as a function of burst contention probability and burst retransmission delay. The model insights on Vegas throughput performance under various burst transmission conditions. Our models provide two closed form throughput estimates. The first form is obtained for barebone OBS networks, while the second is obtained for OBS networks with burst retransmission. We found that the throughput performance obtained through the analytical models closely matches the results obtained from the simulation, which well validated the developed model.

The study has also shown that the extra delay introduced by burst retransmission scheme impacts the throughput of TCP Vegas. Therefore, our future work aims to develop a mechanism that assists TCP Vegas to better react to the delay variation. We conclude that the contribution of this paper comes from the capability of understanding the throughput behaviour of TCP Vegas while the underlying OBS transmission is considered.

ACKNOWLEDGEMENT

This work is funded by Bell Canada through Bell University Laboratories (BUL).

REFERENCES

- [1] S. Floyd, "Quick-Start for TCP and IP," *Internet draft, draft-arnit-quick-start-02.txt*, 2002.
- [2] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *Proceedings, IEEE Infocom*, 2004.
- [3] S. Hegde, D. Lapsley, and et. al., "FAST TCP in high-speed networks: An experimental study," *Workshop on Networks for Grid Applications*, 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, 2004.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [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] 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.
- [13] 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.
- [14] A. Detti and M. Listanti, "Impact of Segments Aggregation on TCP Reno Flows in Optical Burst Switching Networks," *Proceedings, IEEE Infocomm*, 2002. C.
- [15] Samios and M. K. Vernon, "Modeling the Throughput of TCP Vegas," *Proceedings, 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems SIGMETRICS '03*, vol. 31, no. 1, 2003.