BAIMD: A Responsive Rate Control for TCP over Optical Burst Switched (OBS) Networks

Basem Shihada[1], Pin-Han Ho[1,2], Fen Hou[2]
School of Computer Science[1], Electrical & Computer Engineering[2]
U. of Waterloo, Waterloo, Canada

Xiaohong Jiang, Susumu Horiguchi
Graduate School of Information Science
JAIST, Ishikawa, Japan

Minyi Guo
School of Computer Science & Engineering
U. Aizu, Aizu-Wakamatsu, Japan

Hussein Mouftah
School of Information Technology & Engineering
U. of Ottawa, Ottawa, Canada

***Abstract*— Additive Increase Multiplicative Decrease (AIMD) window adjustment mechanism has been embedded in TCP in order to regulate the transmission rate in modern communication networks. In recent years, the AIMD (1,0.5) traffic regulation mechanism along with possibly additional enhancements, such as false timeout detection and explicit notification, has been considered in the carriers with Optical Burst Switching (OBS) as the underlying transmission technology. This paper introduces a novel rate control mechanism based on Generalized AIMD (α,β), called Burst AIMD (BAIMD), for tuning the rate control parameters (α, β) at each sender. BAIMD is designed to improve throughput while maintaining friendliness with co-existing AIMD (1,0.5) flows, and is characterized in the following two folds: (1) no burst window is required in the TCP sender's level; (2) no explicit notifications are required. The above characteristics make the proposed scheme distinguished from all the past reported counterparts by minimizing the signalling efforts and control complexity. The simulation result shows that BAIMD can solidly outperform the past reported AIMD-based (1,0.5) rate control schemes under a wide range of traffic loads. We also suggest that BAIMD rate control mechanism may serve as a better choice than AIMD (1,0.5) in the bufferless OBS networks due to its dynamic and flexible (α,β) parameter pair.**

I. INTRODUCTION

Optical Burst Switched Network (OBS) is envisioned as one of the most promising underlying technologies for the next-generation all-optical networks. Compared with the static Optical Circuit Switched (OCS) with Wavelength Routed Network (WRNs), OBS transmission is dynamic in nature with on-demand bandwidth allocation such that a better dynamicity can be achieved [1-2]. Compared with Optical Packet Switched (OPS), OBS induces much less technical barriers in the practical implementation, and can bring together the best of the OCS and OPS networks. With OBS, data bursts of variable lengths are launched and asynchronously switched along predetermined physical routes to support specific services. The control packet is initiated at the source node corresponding to each data burst and is sent prior to the date burst to configure the switch fabrics at each intermediate OBS switch [1], [2].

OBS has been widely considered to provision UDP-based services due to its high dynamicity and unacknowledged resource reservation. However, the Internet is essentially dominated with data traffic, in which the TCP bulk data transfer mechanism with strict QoS requirements in terms of data integrity and reliability are of extremely high importance. To our best survey, only a few studies have been reported dedicatedly on TCP over OBS networks [4], [5], [6], where a relatively limited understanding on the TCP performance behaviours over the OBS networks has been gained in presence of burst drops at a wide range of traffic loads.

Most of the currently proposed TCP implementations follow the Additive Increase Multiplicative Decrease (AIMD) (1,0.5) window-based congestion control for regulating data transmission. Under such a framework, a TCP sender receives an acknowledgment (*ACK*) for each launched packet or a sequence of packets (or called a segment). The transmission rate is linearly increased by enlarging the congestion window (*cwnd*) by one packet per round trip time (*RTT*) until either packet is dropped or the round trip timer expires [3]. If the packet is dropped, the sender will receive a triple duplicate acknowledgment (TD) and halves its *cwnd*. TCP considers packet loss/delay events to be the two main indications of congestion in the network, and halving the window size can effectively resolve the congestion for bulk data transfer applications.

The abovementioned AIMD (1,0.5) congestion control mechanism is subject to great challenges when OBS is adopted in the underlying transportation infrastructure. With OBS, burst delay and drop could not only be caused due to congestion when the network load is heavy, but also due to some other reasons such as burst assembly, random contention, and retro-blocking, etc. An improper reaction on an event of burst dropping or delayed arrival of *ACK* could bring about very negative impacts on thousands of TCP senders.

The study in [4] introduced a false timeout (TO) detection of burst losses due to contention under a wide range of traffic load. Three different implementation scenarios were proposed. The first solution is mainly based on estimating the number of TCP packets assembled in one burst without necessary knowing the burst assembly mechanism deployed at the OBS layer. Along with the *cwnd*, TCP maintains a burst congestion window burst_wd. When a TCP sender detects a timeout, it first compares its *cwnd* with the burst_wd. If the *cwnd* ≤ burst_wd and burst_wd > 3, then the TCP sender treats this timeout as a false one by halving its *cwnd* and performing fast retransmission of the missing segments. If the *cwnd* > burst_wd or burst_wd ≤ 3, the sender considers this timeout as a true timeout by following the normal TCP retransmission procedure. In the second implementation, each TCP packet is acknowledged separately by a TCP agent located at the OBS edge node. This approach can successfully assist TCP from falling into false timeout; however, it violates the end-to-end TCP semantics since the *ACK*s reach the sender before actual completion of the packet delivery. Finally, a more complicated approach has been proposed by maintaining a TCP agent at

each OBS core node. Whenever a burst is dropped at the core node, the agent disassembles the burst and sends a negative *ACK*, called *BNAK*, to each corresponding TCP sender, which is explicitly informed of the missing segments and probably the network conditions. Generally, the abovementioned solutions could introduce intolerably high signalling overhead and may not be achievable in practical Internet.

In [5], the authors proposed to persistently retransmit any dropped burst, where each OBS edge node buffers a copy of the bursts it launched within a certain time window or until the bursts are successfully delivered. The design premise of the scheme is to improve the TCP throughput by increasing the reliability of OBS layer transmission. However, some deficiencies could be inevitably introduced such as the large buffering capacity and delay. Furthermore, the scheme falls short of the capability in dealing with false timeouts in response to the associated burst retransmission delay.

The abovementioned TCP congestion control mechanisms are subject to different problems, and a TCP rate control scheme that can be responsive to the OBS domain network condition without taking much extra signalling has never been seen. In this paper, we are committed to develop a novel approach that can facilitate the user domain TCP window congestion control. In particular, the study takes advantage of the Generalized AIMD (GAIMD) rate control architecture, which is more flexible and adaptive to the network conditions without losing friendliness with co-existing TCP flows. The paper will first apply GAIMD to the IP over OBS environment and demonstrate that the proposed scheme can greatly improve TCP throughput compared with the past reported counterparts by fully exploring the characteristics of OBS transmission.

The rest of the paper is organized as follows. Section II introduces the AIMD (α,β) framework in the OBS environment, which serves as the fundamental part of the study. In Section III, the formulas for the additive and multiplicative control parameters are derived. In Section IV, the derived control parameters are examined and the resultant performance is compared with the conventional TCP. Section V concludes the paper and presents the issues for our future work.

II. GENERALIZED AIMD IN TCP/OBS

This section introduces the architecture of the proposed GAIMD scheme. We first start by investigating the concept of congestion in the OBS environment, followed by the effectiveness and justification of deploying the GAIMD rate control mechanism in the OBS networks.

A. Congestion in OBS Networks

In traditional packet-switched networks, IP packets are stored and forwarded to the next hop at each intermediate node by looking up the routing table. In such a switching paradigm, network congestion accuses as a result of buffer overflow, and a packet drop event can serve as a clear indication of congestion. Halving the *cwnd* for each packet drop event at the TCP senders can effectively resolve the network congestion.

The situation is different in the networks with OBS as the underlying transportation technology where each data burst cuts through the preconfigured intermediate core nodes. Some burst drop and TCP timeout events may not be caused by network congestion but simply random contention or extra buffering delay due to the burst assembling algorithm. In such a circumstance, halving the *cwnd* for each packet loss event at the TCP senders will certainly lead to unnecessary sending rate fluctuation and maliciously affect on the throughput. Thus, the design of GAIMD is aimed to improve the TCP throughput when OBS is taken underlying by exploring the characteristic of bufferless transmission and the high frequency of multiple TCP segment losses in a single *RTT*.

In OBS, congestion may occur at ingress and/or at egress node. Precisely, packets may arrive at the ingress node in a bursty manner with a much higher arrival rate than that the ingress is capable of dealing with. This could cause the ingress node to drop bursts due to buffer overflow. Similarly, the egress nodes are subject to receiving a huge number of data bursts queued for disassembly and forward their content to their destinations. We refer to the congestion happening at the network edge as *edge* congestion. In addition to the edge congestion, *core* congestion must also be defined, which leads to congestion in the OBS core switch. Burst dropping occur at the core switch is said to be due to congestion if and only if, the total number of simultaneously arriving burst (or control packets) surpass the number of available wavelengths persistently [9]. The study in [9] proposed an *overlapping degree* model to determine the number of overlapped bursts in one link. The larger the *overlapping degree*, the more likely incoming bursts will be dropped.

B. BAIMD in TCP/OBS

The proposed BAIMD scheme is based on the framework of GAIMD for the *cwnd* adjustment. Two parameters are defined: α and β, which serve as the additive incremental and reduction ratio on *cwnd* in each TCP sender as any packet is successfully delivered or dropped, respectively. Unlike TCP where α and β are constantly set to be 1 and 0.5, BAIMD has the TCP senders dynamically determine the two parameters in such a way that the *cwnd* is increased by α segments for each acknowledged packet in a round trip and is decreases by β ($\beta <$ 1) as any packet loss occurs. It is clear that BAIMD is more general than AIMD (1,0.5) with much better design granularity and flexibility. It is worth noting that the proposed BAIMD scheme follows the slow-start and congestion control mechanisms as for the conventional TCP. The only difference is that upon packet loss, the BAIMD senders adjust their *cwnd* by manipulate α and β values using the measured *RTT* to guarantee the best throughput.

With BAIMD, the sender is not explicitly notified about the used burst assembly mechanism and the reasons of the burst losses. Each sender simply treats a packet loss event as a congestion loss. Obviously, this could lead to overestimation of the network congestion by irrelevantly halving the *cwnd* for every segment drop. To compensate the overestimation, the BAIMD senders take the value of β larger than 0.5 instead of β = 0.5 in TCP. As such, the summarized effect of a burst drop event is estimated when the factors α and β are dynamically determined in the BAIMD senders using the burst-level states (i.e., the traffic load in the OBS domain), such that the best throughput can be achieved for each competing flow. For

example, if a burst is lost when the network load is low, the lost packets are considered due to random contention, and the multiplicative factor is set to be $0.5<\beta<1$. Otherwise, β is set to 0.5 when network load is heavy and the burst dropping is considered to be due to congestion.

One of the most important advantages of BAIMD is its simplicity where no burst-level window is maintained in the TCP sender and no explicit notification specific to each launched TCP segment must be performed between the OBS edge and each TCP sender. Thus, a clean separation between the control signalling between the TCP senders and OBS edge nodes can be gained. Most notably, the BAIMD senders use *RTT* of each launched segment and the number of TOs as references for sensing the network load. For example, data bursts are subject to extra buffering delay at the OBS ingress nodes in response to serious congestion. Thus, the *RTT* will substantially increase. On the other hand, if the data burst is dropped due to random contention, the *RTT* remains approximately the same. Thus, it is considered an indication of low network load.

It is clear that the determination of the β value for those BAIMD senders serves as a critical task for improving the overall throughput. In the following section, an analytical derivation for α and β that can guarantee the best performance over OBS will be presented.

III. BAIMD Rate Control Parameters

In this section, both the additive and the multiplicative factors (α, β) of the BAIMD over OBS are analytically derived. We denote $W_T^{(i)}(t)$ and $W_B^{(j)}(t)$ as the *cwnd* size of the *ith* TCP and the *jth* BAIMD senders at time *t* competing for the common network resources, respectively. Let *K* represents the total number of co-existing flows that traverse through the OBS domain at time *t*.

A. Burst Dropping Probability

In OBS network, burst dropping occurs due to several reasons such as, burst random contention, persistent congestion, and retro-blocking. In the literature, many burst dropping probability models have been proposed to evaluate the burst dropping for different OBS design characteristics. For Just-In-Time signalling scheme (JIT), the output port is modeled as an *M/G/k/k* queue and is solved by using Erlang's-B loss probability formula to obtain the burst dropping. In [10] two-state Markov Chain model is used to approximate the packet loss for Just-Enough-Time (JET) signalling using First Fit with Void Filling (FF-VF) as a link scheduling algorithm. In [11] a burst dropping probability model is obtained while considering different fiber delay lines (FDL) optical buffering granularities. In [5], a burst dropping probability model is derived considering burst retransmission technique.

Each of the abovementioned burst dropping probability models were designed to highlight the burst dropping behaviors regarding specific OBS transmission characteristics. However, none of them have considered the situation that a burst loss event can be caused under a number of network states. In this section, an analysis on burst dropping probability is conducted which further considers the classification of burst dropping events due to either persistent congestion or random contention.

In our analytical model an OBS switch is connected with an adjacent OBS switch with *N* output ports each consisting of *M* number of wavelength channels without wavelength conversion. For simplicity, the following discussions will be based on the case with $N = 1$, where one output port is considered. The derived model can be easily generalized.

The corresponding state transition diagram is shown in Fig. 1. The state *i* represents the case where there are *i* reserved channels in the output port. State *i* jumps to state *i*+1 if a reservation request arrive at the port, or jump to state *i*-1 if a burst finishes traversing the OBS switch. We assume that the burst arrival at the state *i* follows the Poisson distribution with arrival rate λ, and each burst is served with service rate of μ. When $i = M$, the arrived bursts must be dropped. However, in the state where $0 < i < M$, an incoming burst is subject to be dropped in the case where more than $M - i - 1$ bursts arrive in the previous *RTT*. Thus, the dropping probability for an incoming burst to be dropped in the state *M*, denoted as p_M. Let π_i be the probability that the system is in the state *i*. We write the state equations are as follows,

As $0 < i < M$,

$$\pi_i \cdot (\lambda + i\mu) - \pi_{i-1} \cdot \lambda - \pi_{i+1} \cdot (i+1)\mu = 0 \tag{1}$$

As $i = 1, M$

$$\pi_i \cdot i\mu - \pi_{i-1} \cdot \lambda = 0 \tag{2}$$

where π_M denotes to the probability of an incoming burst finds all output port channels occupied and since $\sum_{i=0}^{M} \pi_i = 1$.

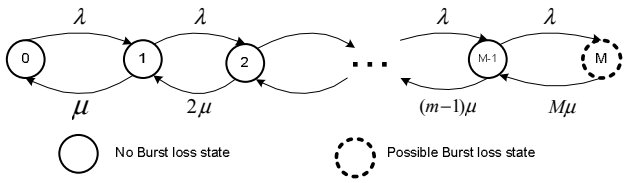

Figure 1. State transition diagram for obtaining burst dropping

From the balance equations of the Markov chain, the probability we have a burst dropping event at state *i*, where $i = 0, 1, 2, \ldots M$, can be given as follows,

$$P_i = \frac{p_0 \rho^i}{i!} \qquad \rho = \lambda / \mu, \quad i = 0,1,2...M$$
$$P_0 = \left[\sum_{k=0}^{M} \frac{\rho^k}{k!} \right]^{-1} \tag{3}$$

On the other hand, the burst dropping probability due to congestion is denote as l_{cgst} and can be expressed in Eq. (4):

$$l_{cgst} = \sum_{i=0}^{M} \pi_i \cdot p_i \quad (4)$$

B. Deriving the Multiplicative Factor (β)

In order to derive the multiplicative factor β in BAIMD, our system classifies the network condition into either one of the following two states: with low (contention) or with high (congestion) overlapping degree probability.

In the TCP layer, when a packet loss occurs at time t and all the connected TCPs as well as BAIMD are notified through TD or TO simultaneously, the maximum link capacity has been reached at the moment. When the *jth* BAIMD sender receives a packet loss notification, its *cwnd* is reduced by a factor of β, while the *cwnd* of each competing TCP sender is reduced by a factor of 0.5. Based on this, the following two relations can thus be derived:

$$W_B^{(i)}(t^+) = \beta \; W_B^{(i)}(t), \quad (5)$$

$$W_T^{(j)}(t^+) = 0.5 \; W_T^{(j)}(t) \quad (6)$$

In the congestion state ($l_{cgst} \geq 0.6$), the BAIMD senders behave the same as TCP senders with β = 0.5. Otherwise, β follow

$$\beta = 1 - l_{cgst} \qquad where, \; 0 < l_{cgst} < 0.6 \quad (7)$$

C. Deriving the Additive Factor (α)

With the BAIMD congestion control mechanism, the relation between α and β can be obtained by Eq. (8) or (9) [7], [8] in order to ensure the friendliness of the BAIMD flows with the other co-existing TCP flows in the case of TD:

$$\alpha = \frac{3(1-\beta)}{1+\beta} \quad (8)$$

while in the case of TO, the relation between α and β becomes:

$$\alpha = \frac{4}{3}(1-\beta^2) \quad (9)$$

It is clear that in the case where β is larger than 0.5, α should be less than 1. The network is in the under-loaded region, and $W_T(t)$ of the *ith* TCP and $W_B(t)$ of the *jth* BAIMD evolve as follows,

$$W_T^{(i)}(\Delta t) = W_T^{(i)}(t) + 1 \cdot \Delta t, \quad (10)$$

$$W_B^{(j)}(\Delta t) = W_B^{(j)}(t) + \alpha \cdot \Delta t \quad (11)$$

The link capacity and the average *RTT* are the two factors that determine the *cwnd* [8]. When the *cwnd* is small due to high dropping rate, the BAIMD flow with larger β has lower throughput compared with that of the other TCP flows. In the case where few burst dropping occurs, the *cwnd* in the BAIMD senders increase by one packet in each $1/\alpha$ *RTT*. However, burst dropping may occur due to random contention, where the *cwnd* is decreased; thus, the actual increase rate of the *cwnd* will never reach α segment. For this reason, α should interact with the burst dropping condition and network load dynamically instead of merely in response to each burst dropping event. Therefore, the following rule is summarized for configuring α: If the *cwnd* is large, then α is set according to Eq. (8) or (9) depending on whether a notification of TD or TO is detected. If *cwnd* is minimal, then BAIMD behaves similar to TCP with α=1. When the *cwnd* size is in the middle, α is set dynamically to ensure a higher throughput and TCP-friendliness. The following flowchart summarizes the functionality of the proposed congestion control scheme.

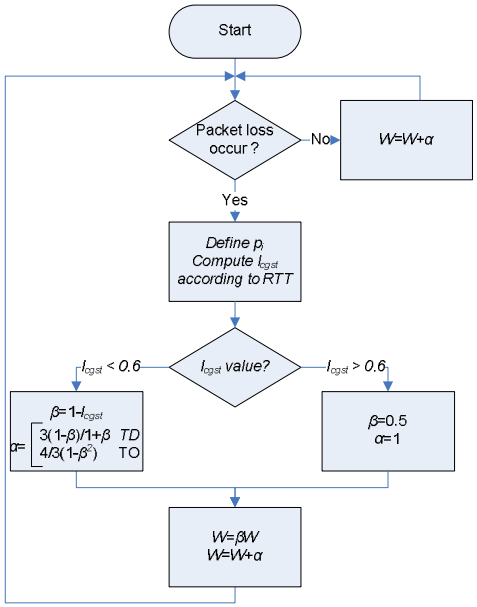

Figure 2. Flow chard demonstrating BAIMD congestion control

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. The distances shown are in km. The TCP senders and receivers are connected to the ingress and egress OBS edge nodes, which assemble the incoming TCP packets into bursts and dissemble the bursts back to TCP packets, respectively.

Each core OBS switch is connected to ingress/egress edge node. The number of co-existing TCP and BAIMD does not impact the overall loss performance of the network. The network consists of background traffic to ensure the low and high congestion scenarios in the simulation. The mixed time/length based burst assembly algorithm is used, where the burst timeout threshold is set to 1*ms* and the maximum burst length is set to 100*KB*. The core nodes implement the *LAUC-VF* channel scheduling algorithm [1], [2], where one bi-directional control channel is allocated along each link. The number of wavelengths on each link is 8 and the transmission

rate on a wavelength is set to be 10 Gb/s. The data traffic traverses through 4 ingress-egress node pairs, where the fixed path routing algorithm is used.

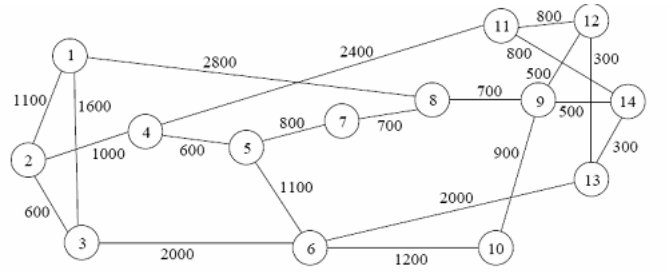

Figure 3. NSF topology

The control packet processing time is set to 1μs at both core and edge nodes. No wavelength conversion is used, and no burst retransmission facility is implemented. A File Transfer Protocol (FTP) application is adopted for generating TCP traffic with a 1*KB* average packet size. We have equipped each core node with one FDL with 3 wavelengths operating at 2, 5, and 8 *ms* delay granularities. The throughput is obtained over a 10^6s simulation period. In order to avoid the phase effect, the traffic flows are launched with a random delay between each other.

The study in [4] has shown that TCP with Selective Acknowledgment (Sack) can achieve decent throughput behaviours when deployed over OBS networks. This is due to the selective acknowledgement mechanism. Therefore, TCP Sack implementation and TCP Reno are taken in our performance evaluation.

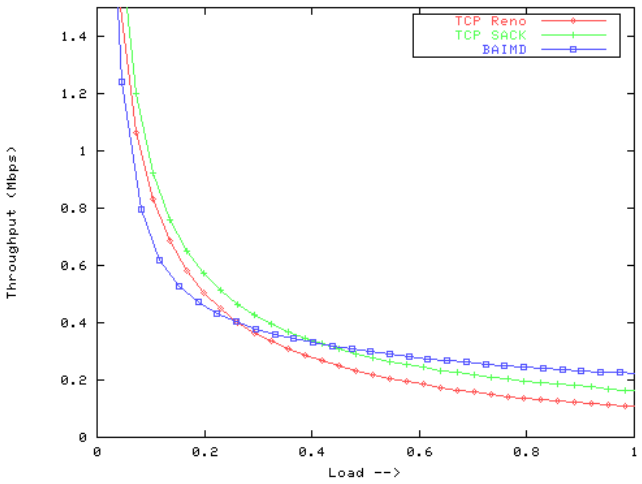

Figure 4. Throughput by TCP Reno, TCP Sack, and BAIMD under low network loads.

Fig. 4 shows the performance of the case where the flows of Sack, Reno, and BAIMD are competing in the OBS domain. It is observed that the throughput by TCP Sack and Reno is dramatically reduced even when the burst dropping rate is quite low because the AIMD (1,0.5) senders always take a burst loss event as due to congestion. It is quite clear that under such a low network traffic load, a burst loss event is mainly due to random contention instead of congestion. On the other hand, the BAIMD senders can achieve better throughput because BAIMD does not rigidly react to a burst loss as a congestion loss at this low level loads. Instead, the factor of β is manipulated in each BAIMD sender to guarantee a smoother window size cut in response to any packet loss, which summarizes the effect of both contention and congestion losses.

Fig. 5 shows the throughput by TCP Sack, Reno, and BAIMD under heavy network loads. In the experiment, up to 5 flows are launched for each scheme (which results in totally 15 competing flows at the core OBS node). Since the traffic load is above the congestion threshold, the BAIMD senders dynamically derive the β value according to the evaluated burst dropping probability due to congestion, which can achieve a smoother congestion window adjustment upon each burst loss event. The simulation result also shows that BAIMD can yield better throughput compared with that by TCP Reno and Sack over heavy network loads since it considers the number of sessions while computing β.

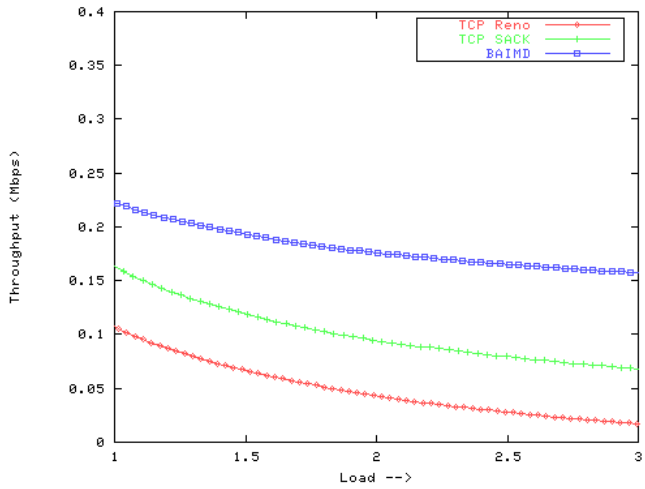

Figure 5. Throughput by TCP Reno, TCP Sack, and BAIMD under high network loads.

Fig. 6 shows the throughput by BTCP with BACK/BNAK [4], which is compared with that by BAIMD. With BTCP, a TCP agent is placed at the ingress nodes to enable the explicit loss notification facility and *ACKs* for the BTCP senders. The proposed BAIMD scheme outperforms BTCP which requires the exchange of extra notification signalling. In the experiment, up to 5 flows are launched for each scheme (which results in totally 20 competing flows at the core OBS node). The superiority of BAIMD lies in the fact that the BAIMD senders can intelligently estimate the burst loss probability due to congestion to determine the corresponding transmission parameters α and β, which is an important design in making the *cwnd* size stable to result better throughput.

In Fig. 7, we additionally consider UDP sessions which coexist with TCP Sack, Reno, and BAIMD. Since no congestion avoidance mechanism in UDP is devised, the UDP traffic does not respond to any burst or packet drop event and

yields much higher throughput. While the TCP Sack, Reno and BAIMD implementations have started to reduce their transmissions due to the increased network load, the UDP senders probe for any available bandwidth and increase their transmission rates. However, it has been seen from the simulation result that BAIMD scheme can achieve the best throughput performance in presence of UDP traffic.

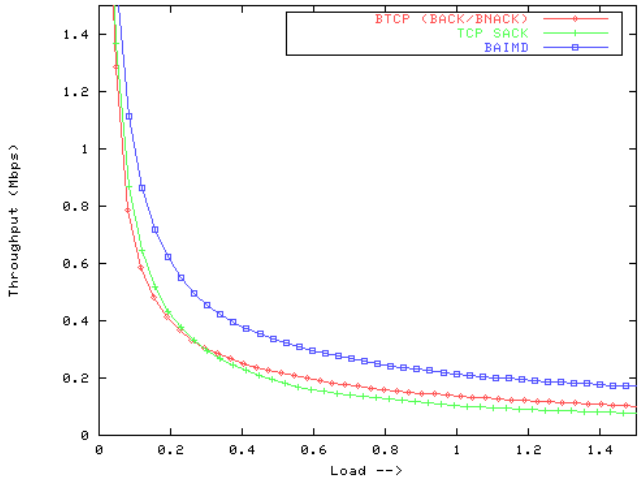

Figure 6. Throughput by BTCP with BACK/BNAK, TCP Sack, and BAIMD scheme.

Throughout the above simulation study, we verified the proposed BAIMD scheme and concluded that BAIMD can achieve significantly better throughput than that by a number of TCP implementations based on AIMD (1,0.5) mechanism in the carrier with OBS transmission. In addition, the proposed scheme can effectively maintain fairness among competing TCP flows and keep awareness on non-congestion burst dropping by manipulating the α and β values in the TCP layer.

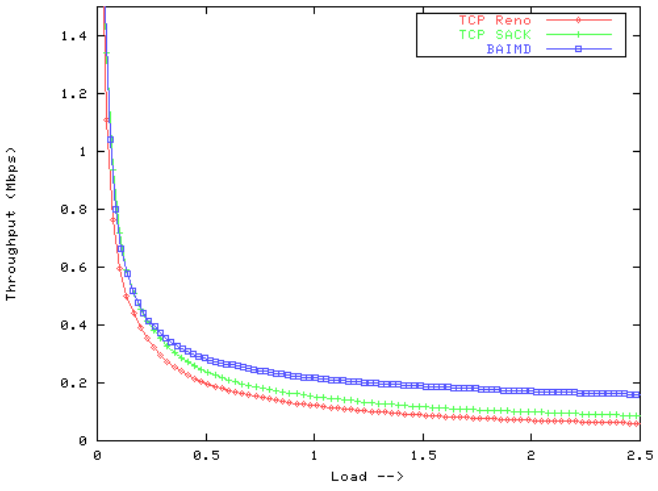

Figure 7. Throughput by the TCP implementations in presence of UDP traffic.

V. CONCLUTION

The paper has tackled the problems of regulating the TCP traffic in the carrier networks supported by Optical Burst Switching (OBS). A novel congestion control scheme called Burst AIMD (BAIMD) is proposed, which is aimed to resolve the vicious effect of non-congestion burst drops in the OBS domain which unnecessarily halve the TCP senders' congestion windows by using the conventional AIMD scheme. The proposed scheme is characterized by the fact that no extra signalling is required between the BAIMD senders and the OBS edge nodes. Furthermore, it has demonstrated a clean separation between the transmission layer and the underlying OBS layers. Analysis on burst dropping probability is conducted to evaluate the rate control ratios α and β for achieving the best throughput without losing TCP friendliness. The proposed BAIMD scheme is further verified by extensive simulation and compared with a number of past reported counterparts, such as TCP Sack, Reno, and BTCP, under a wide range of different circumstances in terms of burst loss probability, the number of competing flows, and the number of co-existing non-TCP traffic. We proved that the proposed BAIMD scheme can yield solid advantages against the schemes based on the traditional AIMD framework in every aspect of interest by the TCP scheme designers, which can serve as a better candidate in the bufferless OBS networks.

REFERENCES

[1] Y Chen, C. Qiao, X. Yu, "An optical burst switching: a new area in optical networking research," IEEE Network, vol. 18, pp. 16-23, 2004.

[2] C. Qiao, M. Yoo "Optical burst switching (OBS), a new paradigm for an optical internet" Journal of High Speed Networks, vol. 8, pp. 69-84, 1999.

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

[4] X. Yu, C. Qiao, and Y. Liu, "TCP implementation and false time out detection in OBS networks," IEEE Infocom, pp. 774-784, Hong Kong, China, 2004.

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

[6] X. Cao, J. Li, Y. Chen, and C. Qiao, "Assembling TCP/IP packets in optical burst switched networks", IEEE Globecom, vol. 3, pp. 17-21, Taiwan, 2002.

[7] Y. Yang, S. Lam, "Generalized AIMD congestion control," *University of Texas, Technical Report TR-2000*, available at http://www.cs.utexas.edu/users/lam/NRL/TechReports/

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

[9] J. Li, C. Qiao, J. Xu, and D. Xu, "Maximizing throughput for optical burst switching networks", IEEE Infocom, pp. 1853 – 1863, Hong Kong, China, 2004.

[10] A. Kaheel, H. Alnuweiri, F. Gebali, "Analytical Evaluation of blocking probability in optical burst switching networks", IEEE ICC, vol. 3, pp. 20-24, France, 2004.

[11] X. Zhu, J. Kahn, "Queuing models of optical delay lines in synchronous and asynchronous optical packet-switching networks", Optical. Engineering, vol. 42, no. 6, pp. 1741-1748, 2003.