Transport Control Protocol (TCP) over Optical Burst Switched Networks

by

Basem Shihada

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Transport Control Protocol (TCP) is the dominant protocol in modern communication networks, in which the issues of reliability, flow, and congestion control must be handled efficiently. This thesis studies the impact of the next-generation bufferless optical burst-switched (OBS) networks on the performance of TCP congestion-control implementations (*i.e.*, dropping-based, explicit-notification-based, and delay-based).

The burst contention phenomenon caused by the buffer-less nature of OBS occurs randomly and has a negative impact on dropping-based TCP since it causes a false indication of network congestion that leads to improper reaction on a burst drop event. In this thesis we study the impact of these random burst losses on dropping-based TCP throughput. We introduce a novel congestion control scheme for TCP over OBS networks, called Statistical Additive Increase Multiplicative Decrease (SAIMD). SAIMD maintains and analyzes a number of previous round trip times (RTTs) at the TCP senders in order to identify the confidence with which a packet-loss event is due to network congestion. The confidence is derived by positioning short-term RTT in the spectrum of long-term historical RTTs. The derived confidence corresponding to the packet loss is then taken in to account by the policy developed for TCP congestion-window adjustment.

For explicit-notification TCP, we propose a new TCP implementation over OBS networks, called TCP with Explicit Burst Loss Contention Notification (TCP-BCL). We examine the throughput performance of a number of representative TCP implementations over OBS networks, and analyze the TCP performance degradation due to the misinterpretation of timeout and packet-loss events. We also demonstrate that the proposed TCP-BCL scheme can counter the negative effect of OBS burst losses and is superior to conventional TCP architectures in OBS networks.

For delay-based TCP, we observe that this type of TCP implementation cannot detect network congestion when deployed over typical OBS networks since RTT fluctuations are minor. Also, delay-based TCP can suffer from falsely detecting network congestion when the underlying OBS network provides burst retransmission and/or deflection. Due to the fact that burst retransmission and deflection schemes introduce additional delays for bursts that are retransmitted or deflected, TCP cannot determine whether this sudden delay is due to network congestion or simply to burst recovery at the OBS layer. In this thesis we study the behaviour of delay-based TCP Vegas over OBS networks, and propose a version of threshold-based TCP Vegas that is suitable for the characteristics of OBS networks. The threshold-based TCP Vegas is able to distinguish increases in packet delay due to network congestion from burst contention at low traffic loads.

The evolution of OBS technology is highly coupled with its ability to support upper-layer applications. Without fully understanding the burst transmission behaviour and the associated impact on the TCP congestion-control mechanism, it will be difficult to exploit the advantages of OBS networks fully.

Acknowledgements

Like many challenges in life, working on a PhD and writing a thesis requires diligence, perseverance, patience, and discipline – characteristics that don't come easily to many of us, and I am not an exception. However, I am very fortunate to have people who encouraged me to stay on track, finish this thesis, and the entire PhD program, despite many challenges.

This thesis would never have been possible without the help of my supervisor and mentor Prof. Pin-Han Ho, who stood by me throughout the PhD program. If I acknowledge all Pin-Han's ideas, discussions, assistance, and guidance, his name will be all over this thesis. Pin-Han helped me in a great deal in developing my views, reviewing my work, and evaluating the topics critically.

I would also like to thank my PhD advisory committee members Prof. M. Tamer Özsu, Prof. James P. Black, D. R. Cheriton School of Computer Science, for the insightful discussions and guidance. Also, thanks to Prof. Sagar Naik, department of Electrical and Computer Engineering, for discussion and kindly offering to read my thesis. Also my acknowledgement is extended to Prof. Douglas Stinson for assistance, motivation, and support. Special thanks go to Prof. Black for his effort and assistance in revising the thesis and clarifying the presentation of the topics. I also would like to thank my external committee examiner Prof. Hossam Hassanein, School of Computing, Queen's University, for accepting to read and evaluate the thesis.

My acknowledgement also goes to Dr. Qiong Zhang, Arizona State University, and Prof. Jason Jue, University of Texas at Dallas, for their helpful guidance, profound feedback as well as support throughout the thesis.

On a personal level, I would like to thank my parents, Prof. Abdel-Fattah Shihada and Suhaila Abu-Saymah, my brother Ibrahim, and my sisters Laila and Manal. Although they are thousands of miles away, they have always inspired me in this endeavour. I dedicate this thesis to them.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Optical Burst Switching (OBS) is considered a promising solution for all-optical switching in Wavelength Division Multiplexing (WDM) networks. OBS has attracted researchers' attention due to its ability to achieve dynamic and on-demand bandwidth allocation, which offers improved network economics and enables control and management integration. The evolution of OBS technology is highly coupled with its ability to support upper-layer applications. Most of today's Internet applications use the Transport Control Protocol (TCP) for maintaining reliable and congestion-tolerant data transmission. Using OBS networks as an all-optical long-haul backbone switching technology has proven to affect TCP throughput negatively. Therefore, understanding the burst-transmission behaviour along with the associated impact on the TCP congestion-control algorithms can contribute to better exploitation of the advantages of OBS networks.

1.1 TCP over OBS Networks

An OBS network is basically bufferless yet best-effort in nature [3][14][91][94][97]. The bufferless and all-optical (AO) natures distinguish OBS networks from traditional packet-switched networks, which rely on electronic processing [14]. The implementation of OBS networks requires precise signalling and higher switching speed than current Optical Circuit Switched (OCS) networks [94][97]. In order to transmit a burst from the source OBS node (ingress node) to a certain destination (egress node), a corresponding control packet is created at the source OBS node and is sent prior to launching the burst [97]. The burst cuts through the network following the control packet in a one-way (*i.e.,* Tell-and-Go) transmission scheme.

Although OBS achieves better flexibility and efficiency than OCS networks, OBS suffers from a burst contention phenomenon which leads to burst drops. Burst contention occurs when more than one burst simultaneously attempts to traverse through one output port or wavelength channel [3][14]. When n bursts contend, $n-1$ of them are dropped. This is due to the one-way resource reservation and signalling protocol, where resources are reserved on

the fly (*i.e*, without acknowledgement or reservation guarantee) and the lack of burst buffering. Burst retransmission [104][105], burst deflection routing [37][68], burst buffering through Fiber Optic Delay Lines (FDL) [18], burst segmentation [83], and wavelength conversions [101] are some approaches to dealing with the problem. However, these mechanisms are not considered to be part of typical OBS networks.

TCP congestion control, which adapts the Additive Increase Multiplicative Decrease (AIMD) approach, is designed to cope with buffer-oriented Internet Protocol (IP) networks. However, OBS networks aggregate multiple TCP/IP packets in a single burst. The burst is transmitted without any buffering delay, delivery guarantee, or processing overhead. The aggregation of multiple TCP/IP packets in single burst, the bufferless burst transmission, the burst contention problem, and the proposed burst contention resolution schemes affect TCP performance. Therefore, modifying TCP congestion control is necessary to cope with the new burst-transmission characteristics.

1.2 Problem Definition and Motivation

TCP is the most important Internet transport service that is self-regulating (*i.e.,* it adjusts the transmission rate according to the network congestion status) and maintains fairness (*i.e.,* it shares the available bandwidth among the flows fairly) for bulk data transfer. In the current Internet, the underlying optical transport layer is based on SONET and OCS, where the Quality of Service (QoS) of the provisioned services can be guaranteed [84]. This provides good support for TCP-based applications. For OBS to be considered as a next-generation optical Internet backbone, significant efforts must be addressed to devising control and signalling protocols that support upper-layer services such as TCP [29][46][58][81]. In OBS, an edge node capable of performing electronic-to-optical (E/O) and optical-to-electric (O/E) conversion, collects, sorts, and assembles incoming IP packets from the higher layers into optical data bursts, and disassembles optical data bursts into IP packets destined for it. One node may correspond to thundered of thousands of TCP senders [29], where hundreds or thousands of TCP segments are assembled and transported in a single burst. Burst losses in OBS networks maybe caused not only by the high utilization of network resources, *i.e.,* long-

term congestion, but also by random burst contention that occurs even when network resource utilization is low. A burst loss due to random contention in the OBS domain may cause the TCP senders to react improperly to the resultant TCP packet-loss event, *i.e.*, by cutting the congestion window or even entering the slow-start state. This is the *false-congestion detection phenomenon* [98]. False-congestion detection significantly downgrades TCP throughput in OBS networks and results in unnecessary packet retransmissions and unnecessary decreases in the transmission rate.

Depending on the number of multiplexed TCP flows and the burst assembly process, TCP performs differently over OBS networks. In the case where few TCP connections are multiplexed at the edge node, the probability of having multiple TCP segments from a single TCP flow assembled in a single burst is high. When a burst contention loss occurs, few TCP connections will be affected or even trigger a timeout (TO). However, in the case where millions of TCP connections are multiplexed at the edge node, the probability of multiple TCP segments from a single connection being assembled in a single burst is very low. Therefore, a single burst likely contains segments from many individual flows. For example, assume that 10,000 TCP flows are multiplexed at the edge node. Assume a single burst contains 1000 TCP segments each belonging to a different flow. When a burst is lost due to contention, the 1000 flows will eventually cut their congestion windows (*cwnd*) and each TCP implementation enters fast retransmission, thus causing a dramatic decrease in the overall network utilization, which is also called *global synchronization problem*. This congestion-detection situation is completely different from traditional IP-based networks where only one TCP flow is affected by each packet loss. We conclude that burst dropping in OBS networks wastes routing, assembly, and signalling efforts performed at both the IP-access and the OBS network and could significantly affect the TCP senders in terms of how the TCP congestion windows are adjusted. An improper strategy and reaction upon a burst delay and/or loss event will reduce overall TCP throughput significantly.

In order to cope with the TCP false-congestion detection problem, it has been shown in [104][105] that burst retransmission and burst deflection schemes can reduce burst-loss probability and hide burst-loss events from the upper TCP layer, thereby eventually reducing

the chances of TCP false congestion detection. With burst retransmission or deflection, bursts subject to contention can be retransmitted at an OBS edge, or can be deflected to an alternate route in the OBS network. Hence, burst-contention probability, which is the probability that a burst experiences random contention at low traffic load, could be much larger than the burst loss probability, which is the probability that a dropped burst is sensed at the TCP layer. Although burst deflection and/or retransmission can reduce burst-loss probability, they introduce additional delay for bursts that are retransmitted or deflected. As we will explain later, this extra delay affects the throughput of delay-based TCP (because it increases the TCP round trip time) as well as the behavior of the congestion-detection algorithm.

1.3 Thesis Objectives

OBS introduces many advantages over Wavelength Routed (WR) OCS networks. However, a few key problems have been identified and are still open, such as burst contention resolution and end-to-end delay variation. These problems make it challenging to design TCP extensions for use with OBS networks.

The primary objective of this thesis is to develop a suite of interoperable strategies that enable TCP congestion control to operate well given the intrinsic characteristics of OBS. In particular, we identify four important goals.

1. Analyze and prove the negative impacts of burst-loss events on conventional TCP congestion control, thus on TCP throughput. This is performed through examining a number of previously reported TCP congestion-control mechanisms, including Reno, New Reno, SACK, Duplicated SACK (DSACK), FACK, TCP with Eifel, and Burst TCP (BTCP), in terms of throughput under different burst dropping probabilities and traffic loads.
2. Propose and analyze a novel congestion-control mechanism for TCP over OBS networks, called Statistical Additive Increase Multiplicative Decrease (SAIMD). SAIMD maintains and analyzes a number of previous round trip times (RTTs) at the TCP senders in order to identify the confidence with which a packet-loss event is due

to network congestion. The confidence is derived by positioning short-term RTT in the spectrum of long-term historical RTTs. The derived confidence corresponding to the packet loss is then taken into account by the policy for TCP congestion-window adjustment. We show through extensive simulation that the proposed scheme can solve the false-congestion detection problem effectively and outperform conventional TCP significantly. Also, based on the proposed congestion-control algorithm, a throughput model is formulated and verified by simulation.

3. Propose and analyze a novel congestion-control scheme, called TCP with Explicit Burst Contention Loss Notification (TCP-BCL), which is the first study that integrates the explicit notification mechanism with the Generalized Additive Increase Multiplicative Decrease approach (GAIMD) over OBS networks. The basic design principle of the scheme is to tune the congestion-control parameters α and β such that the congestion window sizes in the corresponding TCP senders can be adjusted to exploit explicit notification from the OBS edge node. The performance impact on TCP of false-congestion detection is considered and investigated. An analytical model is developed for the proposed scheme and is verified through extensive simulation.
4. Propose and analyze a modified version of delay-based TCP (*e.g.*, TCP Vegas) that adopts a threshold-based mechanism for identifying network congestion status in OBS networks. Throughput models are developed for TCP Vegas and threshold-based Vegas over an OBS network with burst retransmission. Simulation is conducted to validate the models and to verify the proposed threshold-based TCP Vegas. Based on the analytical model, a threshold value that results in an optimal steady state TCP throughput is obtained and verified.

These results provide us with better knowledge and deeper understanding of TCP behaviour over both the conventional IP-based Internet and OBS backbone networks.

1.4 Thesis Organization

The thesis is organized as follows. Chapter 2 provides a detailed survey of the state of the art of TCP extensions or modifications in OBS networks. Chapter 3 presents the methodology used to model and simulate TCP over OBS. It also describes the issues in the design of TCP schemes in the OBS environment, where the throughput performance under a wide range of burst-dropping probabilities and traffic loads is examined for a number of previous TCP enhancements, including Reno, New Reno, SACK, DSACK, FACK, TCP with Eifel, Burst AIMD (BAIMD), and Burst TCP (BTCP). Chapter 4 presents our congestion-control mechanism based on TCP statistical RTT, Statistical AIMD (SAIMD). SAIMD is designed to improve throughput while maintaining friendliness with co-existing TCP flows. Chapter 5 presents our TCP implementation that combines the advantages of the explicit burst-contention loss notification and the GAIMD congestion control. An analytical model is developed for the proposed scheme and is verified through extensive simulation. Chapter 6 discusses the issues of delay-based TCP (*e.g.*, TCP Vegas) over OBS networks and introduces a modified threshold-based TCP Vegas congestion-control scheme that is suitable for the characteristics of TCP over OBS networks. We show that the proposed threshold-based TCP Vegas scheme is able to distinguish whether the increase in the RTT is due to network congestion or due to burst contentions at low traffic loads. Chapter 7 concludes the thesis and presents open problems, selected directions for further research, and recommendations.

Chapter 2
State of the Art of TCP over OBS Networks

In this chapter, we provide a comprehensive survey of TCP congestion-control enhancements in OBS networks. These enhancements aim to mitigate the numerous side-effects of OBS networks such as the bufferless characteristics of burst transmission.

2.1 Introduction to TCP

The complex Internet infrastructure interconnects millions of communicating devices through wired and wireless connections. One of the key success factors of today's Internet infrastructure is the ability to maintain reliable, self-regulating, and congestion-tolerant transport to serve the end-user applications. TCP [82], originally designed for military communication by ARPANET [46][58][61], is the most pervasive protocol for the majority of current Internet-based applications. TCP is the predominant protocol in terms of the traffic volume (in bytes), consisting up to 90% of the total Internet traffic [29]. The High-Performance Computing (HPC) networks observed an average share of 83% of TCP traffic, and for the NETI@Home data, TCP flows are the majority of traffic volume.

TCP congestion-control mechanisms can be classified into three categories: (1) dropping-based (*e.g.*, TCP Reno [81] and TCP SACK [56]), (2) delay-based (*e.g.*, TCP Vegas [9], Fast TCP [36][39]), and (3) explicit-notification-based (*e.g.*, XCP [42]). These congestion-control mechanisms are basically used at the TCP sender to determine whether the network is congested and the transmission rate should be reduced accordingly, while the receiver is totally reactive to the transmission protocol.

Dropping-based TCP protocol implementations such as TCP Reno simply take a packet (or TCP segment) loss as an indication of network congestion [64]. These TCP protocol stacks follow the Additive Increase Multiplicative Decrease (AIMD) window-based congestion-control mechanism for regulating data transmission and maintaining network bandwidth usage. At each TCP sender, the AIMD scheme is comprised of four stages: (1) slow start, (2) congestion avoidance, (3) fast retransmission, and (4) fast recovery [81].

At the slow-start stage, the TCP sender has just started, or restarted the transmission due to a timeout (TO) event. The congestion window (*cwnd*) is initialized to the size of one segment for dropping-based TCP (*e.g.,* Reno, SACK). The size of the *cwnd* is increased linearly for each successfully delivered (acknowledged) segment. The linear growth continues until either it exceeds the receiver's advertised congestion window (*rwnd*) or a packet-loss event occurs, which leads the TCP sender to enter the congestion-detection stage. Note that a TCP sender senses network congestion either when a packet timeout occurs or when triple-duplicated acknowledgments (TD) are received. Either case reflects packet loss or out-of-order packet delivery. When a timeout occurs, which indicates heavy congestion, *cwnd* is set to one segment, and TCP returns to the slow-start stage, followed by the congestion-avoidance stage. In the triple duplicate case, which indicates light congestion, TCP enters the fast-retransmission stage by taking half of the sender's *cwnd* as the slow start threshold *ssthresh* and setting *cwnd* to *ssthresh* plus three segments. Then the sender attempts to retransmit the missing segments and increments *cwnd* by one segment. After receiving an acknowledgment for the second data segment, *cwnd* is set to *ssthresh*.

The delay-based TCPs, in particular TCP Vegas, estimate the available bandwidth and the congestion status by measuring the delay of each transmitted packet in terms of round trip time (RTT). The performance of Fast TCP, which enhances TCP Vegas throughput over high-speed large-bandwidth networks, has been evaluated in [36][39], as we describe in the Chapter 6.

TCP Vegas modifies TCP Reno in the slow start, congestion avoidance, and retransmission stages. TCP Vegas determines the congestion status in the network through comparing the *estimated* and the *actual* throughputs. TCP Vegas first computes the *BaseRTT*, which is the minimum measured RTT, (*i.e.,* the summation of the propagation delay and the queuing delay). The *expected* throughput can thus be derived as,

$$expected = \frac{cwnd}{BaseRTT} \tag{2.1}$$

Second, the *actual* throughput is calculated for every round as,

$$actual = \frac{cwnd}{RTT}, \tag{2.2}$$

where *RTT* is the most recent measured RTT. The sender then computes:

$$\begin{aligned} Diff &= (expected - actual)BaseRTT, \quad where\ Diff > 0 \\ &= (\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT})BaseRTT \\ &= cwnd(1 - \frac{BaseRTT}{RTT}). \end{aligned} \tag{2.3}$$

Obviously, *Diff* is non-negative, and is used to adjust the next *cwnd* size. Vegas defines two threshold values, α and β, for controlling *cwnd* size:

$$cwnd = \begin{cases} cwnd + 1 & diff < \alpha \\ cwnd & \alpha \le diff \le \beta. \\ cwnd - 1 & diff > \beta \end{cases} \tag{2.4}$$

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

TCP Vegas congestion avoidance aims to maintain the expected number of backlog packets in the network between α and β bytes. If the *actual* throughput is smaller than the *expected* throughput, then it is likely that the network is congested. Thus, the TCP sender reduces the transmission rate. However, if the *actual* throughput is close to the *expected* throughput, the TCP flow may not be utilizing the available bandwidth properly, and thus the rate should be increased.

In slow start, TCP Vegas initializes the *cwnd* to the size of two segments and increases *cwnd* exponentially every other round. TCP Vegas exits slow start when *cwnd* reaches the slow-start threshold, denoted by γ. Between the two consecutive rounds, *cwnd* is unchanged, which ensures a valid comparison between the *expected* and *actual* throughputs.

In the fast retransmission stage, when the 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 decides whether to retransmit the packet based on the following two heuristics: First when a duplicated ACK is received, Vegas checks if the difference between the current time and the timestamp recorded for the associated packet is greater than the TO-threshold value. If so, it retransmits the packet without waiting for the remaining incoming duplicate ACKs. The second heuristic is applied when an ACK is received. If it is the first or second ACK after a retransmission, Vegas then checks if the time spent since the packet was sent is larger than the TO-threshold value. If so, Vegas retransmits the packet. This retransmission approach catches any other packet that may have been lost prior to the retransmission without waiting for a duplicated ACK. Hence, the Vegas retransmission mechanism reduces the time to detect a lost packet from the third to the first or the second duplicate ACK. After packet retransmission is triggered by a duplicated ACK, *cwnd* is reduced by 25% (instead of 50% in Reno), only if the time since the last-window size reduction is more than the current RTT.

The third category is explicit notification-based TCP. In this category, TCP with Explicit Loss Notification (ELN) [1] and Explicit Congestion Notification (ECN) [25] are the two representative approaches. TCP implementations in this category rely on explicit information received from the network to identify suspicious packet losses that cause unnecessary packet retransmissions. They can distinguish among packet losses due to congestion, contention, link failure, or other reasons. Packet retransmission starts as soon as the ELN/ECN is received in the round following a packet loss.

In the following two sections, we discuss the evolution of optical switching technologies, along with a detailed illustration of OBS network architecture and the burst transmission characteristics.

2.2 The Evolution of Optical Switching Technologies

Optical backbones based on WDM are the most common carriers in modern communication networks. Several switching technologies have been proposed to take advantage of the high transmission capacity of fiber optics. Early approaches followed OCS, where point-to-point

lightpaths are established for a relatively long period of time [46]. OCS follows the store and forward approach where each optical switch performs optical-to-electrical-to-optical (O/E/O) conversions. However, the rapid increase in user demands and traffic engineering requirements have imposed several limitations on the adoption of OCS. First, the task of Routing and Wavelength Assignment (RWA), which optimizes the assignment of wavelengths to all users, is NP-Hard [4][46][80]. Second, due to its quasi-static nature, OCS cannot easily support dynamic traffic variation or frequent connection requests.

Due to the ubiquity of IP, much research has addressed the integration of IP with WDM networks. Optical Packet Switching (OPS) [46] has been proposed to support this integration, where the optical core can be taken as an extension of the IP layer. With OPS, each optical packet consists of header and data payload and is launched into the optical network. As packet arrives at an optical switch, the header is converted and read in the electronic domain to configure the switch fabric, while the data is buffered optically in a fiber delay line (FDL) [12][18][30][107] until the switch configuration is completed. OPS follows an all-optical (AO) buffer-oriented transmission approach. OPS can successfully solve the inadequacy and inefficiency of the OCS technology in terms of bandwidth provisioning, dynamics, and capacity utilization. However, the technical barrier before such highly synchronized systems can be commercialized is the high cost of all-optical buffer facilities and the dimensioning of these delay lines.

OBS has attracted the attention of researchers due to its ability to achieve more dynamic and on-demand bandwidth allocation than OCS. It also offers improved network economy and enables control and management integration [3][14][63][91]. Compared with OPS, OBS is more practical to implement, and combines the best of OCS and OPS networks.

With OBS, a data burst is formed at the edge node by assembling multiple incoming packets with same destination and/or QoS requirements. To transfer the burst to the destination, a corresponding control packet that contains both the burst size and the burst arrival time is created at the network edge node, and is sent prior to the launch of the burst [91][100]. Since the bursts cut-through each intermediate node in the network core while the control packets are subject to processing at each core node, a certain amount of offset time

must be imposed between launching control packets and the corresponding bursts. The calculation of the offset time has to consider the upper bound on the number of hops and nodal processing delay. OBS follows an AO transmission approach. Figure 2.1 highlights the evolution of the optical switching technologies.

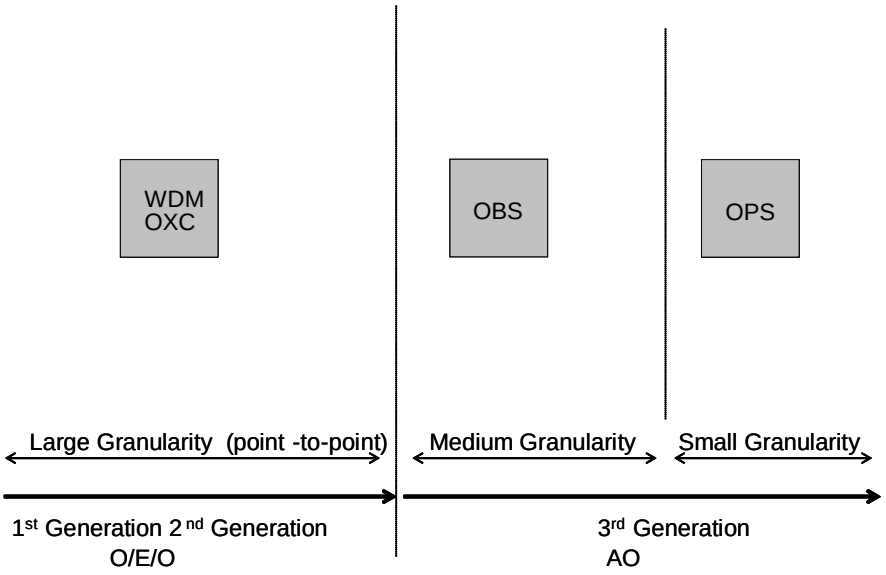

Figure 2.1 Evolution of optical switching technologies

With its out-of-band signalling mechanism, OBS provides complete separation between control and data domains that can yield better network manageability and flexibility. Since the launched bursts cut through the network core without any buffering, the bursts can be subject to a minimum amount of delay. Table 2.1 summarizes the characteristics of the three different switching schemes.

Property	Optical Circuit Switching	Optical Packet Switching	Optical Burst Switching
Bandwidth	Low	High	High
Setup latency	High	Low	Low
Switching speed	Slow	Fast	Medium
Processing overhead	Low	High	Low
Traffic adaptively	Low	High	High

Table 2.1 Comparison of optical switching technologies

2.3 Optical Burst Switching (OBS) Networks

The implementation of OBS networks requires precise signalling and higher switching speed than that required by OCS. The edge nodes are capable of performing electronic-optical (E/O) and optical-electric (O/E) conversion to collect/sort/assemble IP packets, ATM cells, SONET frames, or any other types of traffic packets from the upper layers into optical bursts, and disassemble the optical bursts at the destination edge nodes. The edge nodes are also responsible for signalling and routing in the OBS network. The core nodes (*i.e.*, optical switches), on the other hand, respond to configuration requests from the edge nodes to forward bursts in an AO manner [3][14][90][91][100]. Figure 2.2 presents the OBS network architecture. Several burst-assembly algorithms have been proposed and later discussed in Section 2.3.1.2.

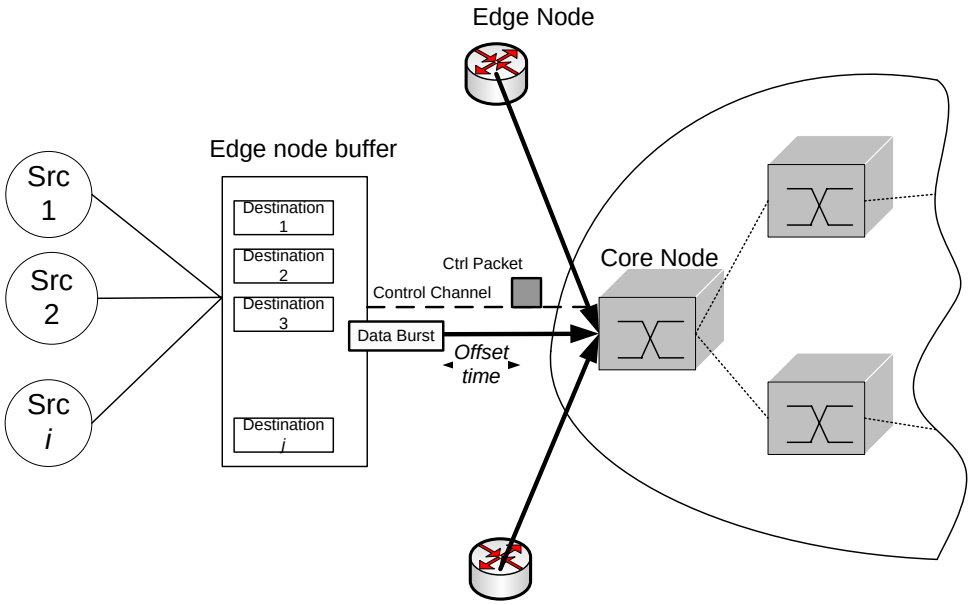

Figure 2.2 Optical burst switching network architecture

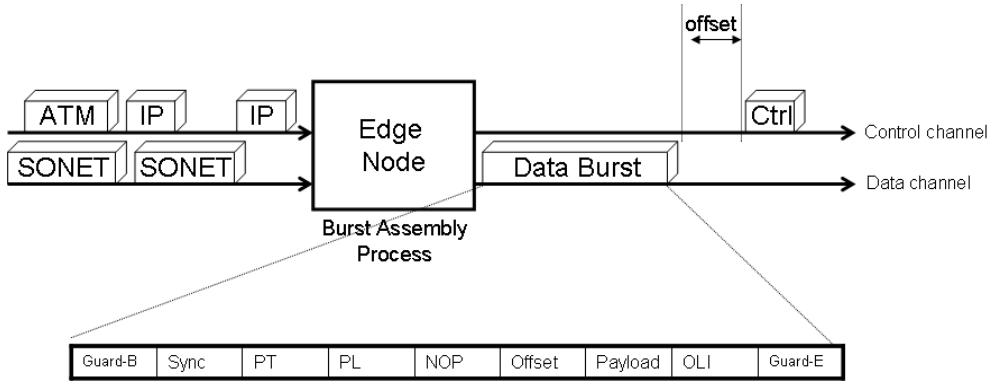

Figure 2.3 OBS edge router architecture

A burst contains several other fields when assembled at the edge node. Figure 2.3 illustrates the burst assembly process as well as the burst format. A burst consists of Guard Preamble (Guard-B) and Guard Postamble (Guard-E) fields that prevent the uncertainty of data burst arrival time and data burst duration due to clock drifts between different core nodes. The payload consists of the assembled data packets. Payload Type (PT), Payload Length (PL), and Number of Packets (NOP) represent the type of data, data length, and number of packets assembled in the burst, respectively. The offset of padding indicates the first byte of padding. The Optical Layer Information (OLI) includes some information for performance monitoring and forward error correction obtained from the communication channels [91].

A typical OBS core node (switch) consists of two layers. The upper layer is responsible for processing control packets and configuring the switching fabric. In this layer, control packets are processed, switching resources are reserved, and freed after each burst exits the switch. The switch matrix control unit, the port forwarding table, and the link scheduling module are also maintained in the upper layer. The lower layer is responsible for AO burst transport functionality. The lower layer consists of optical ports, wavelengths, and optical-to-optical connections. Figure 2.4 illustrates the generic OBS core switch architecture.

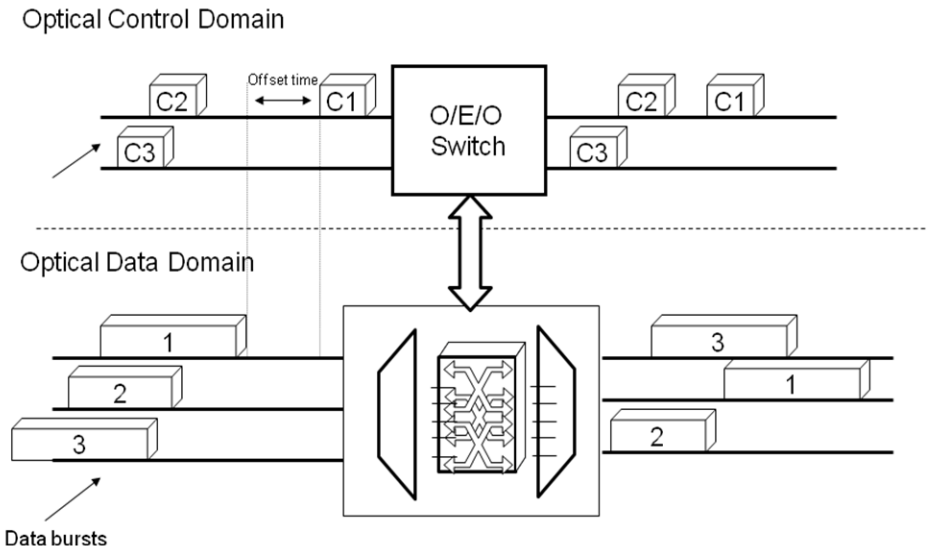

Figure 2.4 OBS core switch architecture

In both Figure 2.3 and Figure 2.4, the offset-time delay between the control header and the data burst is shown. As indicated earlier, the offset time is set to ensure that the switch fabric is fully configured before the arrival of the burst. Determining the proper offset time is fundamental to the functionality of OBS [50][62][91]. Several approaches were proposed to elongate the offset time to achieve specific QoS differentiation purposes. A fixed offset-time algorithm proposed in [91] suggested calculating the offset time based on the summation of the total nodal processing time plus the switch-fabric configuration time at all the intermediate core nodes. In order to implement this algorithm, the edge node maintains the total number of hops in the route (path). Finally, wavelength-routed OBS offset time (WR-OBS-Offset), presented in [50], obtains the offset time from a centralized resource scheduler [100].

In the literature, two resource-reservation protocols for OBS have been investigated widely: explicit [69][97] and implicit [94] approaches. The Just-In-Time (JIT) resource-reservation protocol introduced in [100] follows an explicit reservation approach. Upon receiving the control packet at the first core node, an acknowledgement packet is generated and returned to the edge node. Once the edge node receives this packet, the data burst is launched after a delay estimated by the response time for the control packet. The control

message continues traveling on hop-by-hop basis, followed by the data burst, until the destination is reached. Once the burst leaves the edge node, a tear-down packet is sent back along the same route to release reserved resources.

The Just-Enough-Time (JET) resource-reservation protocol introduced in [97] uses delayed reservation with implicit resource release. The intermediate core nodes traversed by a data burst can automatically release the reserved switching fabric shortly after the burst leaves the switch, because each intermediate node is informed of the departure time of the data burst when the control packet sets up the path. Therefore, each intermediate node does not need an explicit notification to release the resources [20].

Channel scheduling is another important mechanism for enabling OBS transmission and improving link throughput [92]. Several approaches were reported in the literature. The Latest Available Unscheduled Channel with Void Filling (LAUC-VF) simply maintains single scheduling horizon for each channel [3]. Only channels whose horizons precede burst arrival can be considered available. Link void is defined as the unused (wasted) gaps between two successive scheduled bursts. The Min-Starting Void (SV) algorithm [92] was proposed as an enhancement to LAUC-VF, which schedules a data burst in $O(\log n)$ time, where n is the total number of void intervals. Furthermore, the SV algorithm minimizes the new voids generated from the time gap between the end of a new reservation and an existing reservation.

2.3.1 Characteristics of OBS Networks

The bufferless nature of OBS networks results in two major transmission characteristics that distinguish them from others, which are (1) random burst contention losses and (2) assembly and resource-reservation delay.

2.3.1.1 Random Burst Contention Losses

Contention occurs when two or more packets try to access the same output port simultaneously. In IP-based networks with electronic processing, contended packets can be stored temporarily in the electronic memory buffers of the intermediate IP routers instead of being dropped immediately. Except for extreme cases such as buffer overflow, the packet is

delivered hop by hop until it reaches the destination. This is not the case in OBS networks. As a compromise between OCS and OPS networks, OBS achieves some amount of statistical multiplexing by using JIT or JET resource reservation schemes, while buffering the data bursts only at the edge nodes. However, due to the one-way signalling mechanism, the lack of global scheduling, and the lack of optical/electric buffering, burst contention results in immediate dropping even when the network is lightly loaded [14]. These are referred to as random burst contention losses and they impose a significant impact on the upper-layer protocols (*e.g.*, TCP), especially for those that take packet losses as the only indication of network congestion. In contrast to random burst contention losses, the network could be subject to persistent network congestion, where high utilization lasts for a long time and more burst contention occurs. It is hard to distinguish between random burst contention losses and persistent congestion losses.

2.3.1.2 Assembly (Burstification) and Resource Reservation Delay

As mentioned earlier, four categories of burst assembly algorithms have been proposed [3][14][91]. The first are time-based, in which a timer is set after the beginning of each new assembly cycle. All packets with a common destination edge node arriving within a specific period of time are assembled into a single burst. The second set of algorithms is length-based, where a burst-length threshold is set. With these algorithms, the burst-length threshold serves as the minimum burst length before the burst leaves the edge node. The third set of algorithms combines the time and the burst-length algorithms, where the bursts are assembled and sent either when the burst-length exceeds the threshold or the timer expires. The last set of algorithms are adaptive, where a dynamic adaptive threshold on the burst length is set to optimize the overall performance for certain QoS-sensitive traffic [14][35] [102][103]. Traffic traversing an OBS network must experience burstification and burst resource reservation delays. These delays affect delay-sensitive data and delay-sensitive transport protocols such as TCP Vegas.

2.3.2 Taxonomy of OBS Networks

OBS networks can be classified based on the resource-reservation and signalling mechanisms as implicit [97] or explicit [94] OBS. They can also be classified based on the burst assembly mechanisms as time-based, burst-length-based, mixed, and optimized-burst-size-based OBS networks [3][14][91]. Since burst-contention and resolution schemes play a vital role in the deployment of TCP over OBS, we classify OBS networks according to whether a burst-contention-resolution mechanism is employed in the OBS network, as either *barebone* OBS or OBS with *Burst-Contention-Resolution (BCR)*.

2.3.2.1 Barebone OBS

In barebone OBS, a burst-control packet is sent to reserve the intermediate switching fabric. After a predefined offset time, the data burst cuts through on a certain path until it reaches the destination. As mentioned earlier, two major resource-reservation protocols in OBS networks are Just-In-Time (JIT) and Just-Enough-Time (JET). Since neither scheme guarantees channel availability, two or more data bursts may attempt to access a common output port simultaneously. One of the bursts must be dropped while the other is delivered. The dropped bursts are lost since barebone OBS does not employ any buffering, burst retransmission, burst deflection, or burst-contention-resolution mechanisms.

2.3.2.2 OBS with Burst Contention Resolution (BCR)

Burst contention resolution could involve retransmission at the edges [104][105], burst deflection [37][68], optical buffering [18], burst segmentation [83], or wavelength conversion [101] at the core switches.

With burst retransmission, deflection, or buffering, the data bursts subject to contention in the OBS core can be retransmitted from the edge node, deflected to an alternate route (*i.e.*, an alternate output port), or buffered using FDL at the core node, respectively. Note that burst contention constitutes a significant portion of the total burst losses in OBS networks. With burst segmentation, the contended bursts can be segmented so that the overlapped segments are discarded at the core node, while the rest of the bursts can be transmitted successfully. With wavelength conversion, optical data flows on a certain wavelength can be converted to

another, which can effectively reduce the resource segmentation and result in a better chance of successfully forming a data path.

These burst-contention-resolution schemes have been proven to increase the transmission reliability and throughput effectively at the expense of introducing extra overhead, complexity, and cost, as discussed in detail later.

2.4 Issues of TCP over OBS

It has been shown that the burstification process at the OBS edge nodes has significant impact on the performance of TCP Reno [19]. Based on the access-network bandwidth and the assembly-algorithm thresholds, TCP flows are classified into three categories: fast, medium, and slow. In a fast flow, all transmitted segments of the congestion window are assembled into a single burst. On the other hand, medium and slow flows have some proportion of the congestion window segments assembled in a single burst. For medium to slow flows, a performance penalty is introduced on the TCP sending rate due to the assembly delay, but aggregating multiple TCP segments due to burst assembly increases TCP throughput performance [19][99]. When a burst is lost, a fast TCP flows detects the segment loss through a TO, and returns to slow start since the TCP segments of the entire *cwnd* are lost, which indicates severe network congestion [98]. However, the medium and slow TCP flows enter fast retransmission since only some of the *cwnd* segments are lost [98].

2.4.1.1 Multiple TCP-Segment Loss

In OBS networks, TCP performance is significantly affected by random burst losses [19][98] [99], which cause unnecessary congestion window cuts. Hundreds or thousands of TCP segments could be assembled and transported together in a single burst. At the occurrence of burst loss and depending on the TCP flow speed [19], TCP suffers from False TO (FTO) [98] or performs unnecessary fast retransmission [19][98] as a response on burst losses due to random contention.

2.4.1.2 Packet Reordering

In a multi-buffer burstification, where the bursitification scheme maintains several assembly

buffers for each destination, out-of-order burst delivery may occur when segments of a TCP session are assembled in two or more bursts that are sent in a different order from that of the arrival of the TCP segments [8]. The frequency of out-of-order delivery depends on the time and burst-length thresholds, the traffic shape, and the QoS criteria defined in the burst-assembly algorithm. Optical buffering in the form of FDL, burst retransmission, and burst deflection can also cause out-of-order delivery. Buffered, retransmitted, and deflected bursts are subject to extra delay compared to subsequent bursts.

The out-of-order burst delivery eventually causes the TCP sender to fall into "packet retransmission ambiguity," since the TCP receiver will issue TD ACKs assuming that there exists a sequence of missing packets. Depending on the number of packets assembled in a single burst, TD ACKs can take place in a single round. The frequency of the false TD ACKs problem is expected to affect the sender dramatically, particularly in the presence of a large number of TCP packets assembled in a single burst. The higher number of false TD ACKs causes the sender's *cwnd* to remain much smaller, which significantly throttles the throughput.

2.4.1.3 Slow Convergence

In OCS networks, the lightpaths are static in nature, last for a long period of time, span a long distance, and serve in backbone networks. In such long-haul networks, TCP suffers severely from very long convergence times. The TCP slow-convergence problem worsens in the presence of a high diversity of RTTs for bursts due to buffering and signalling delays [108]. Similar to conventional OCS networks, TCP over OBS with linear increase of *cwnd* for each successful segment-delivery round takes a significant amount of time to utilize the available bandwidth, while the multiplicative decrease for each packet loss is a critical response to random burst-contention losses and persistent network congestion. Note that it is necessary to maintain a large *cwnd* to achieve high throughput, which can only happen when an extremely low packet-loss rate is seen, such as 10^{-8}. This is unlikely in the presence of random burst-contention losses.

2.5 Taxonomy of TCP Solutions over OBS

In bufferless OBS networks, TCP senders must not blindly consider data loss to be an indication of network congestion; instead, TCP senders should attempt to collect further information that distinguishes between random burst contention and persistent network congestion. This category of TCP implementations may require extra signalling efforts between the TCP senders and the OBS network [98], and/or within the OBS network [21] [104][105].

TCP solutions over OBS networks fall into three categories [70]: (1) link-layer solutions (or solutions maintained at the OBS network), (2) modifications or enhancements based on explicit notifications from the OBS layer, and (3) newly designed TCP mechanisms that do not require explicit notifications .

In the first category, the solutions require an implementation of additional supporting functions at either the OBS edge or core nodes. Two problems arise in this category. First, these schemes break the end-to-end semantics of TCP, and second, they introduce additional control overhead, which may impair the applicability of TCP congestion-control schemes. The solutions in the second category overcome these problems by exchanging extra signalling with intermediate network devices or OBS nodes. However, the price is paid in the effectiveness of the scheme, since it is difficult for the intermediate network devices to provide extra signalling for each TCP flow. Finally, the third category of solutions includes totally new transport protocols that cope with the underlying burst transport behaviours without exchanging explicit information. We briefly describe the three categories.

2.5.1 Link-Layer Solutions

There have been several solutions proposed to increase the burst-transmission reliability in the OBS network in order to improve TCP throughput performance. These schemes typically adopt mechanisms such as adaptive burstification [14], forward error correction (FEC), burst-contention resolution schemes, automatic repeat request (ARQ) retransmissions [104][105], and others. The main advantage of employing link-layer solutions is hiding the (non-congestion) burst loss from the TCP senders, so that the layered structure of network protocols is followed. To achieve link-layer transmission reliability, burst retransmission,

burst deflection, and burst buffering using FDLs are among the widely recognized approaches. These approaches contribute to the reduction of TCP *cwnd* fluctuation and consequently improve TCP throughput at the expense of introducing additional delay and control overhead.

2.5.2 Congestion Detection with Explicit Notifications

TCP senders can be informed of the channel conditions and the actual cause of the packet-loss events, *e.g.*, congestion, contention, packet corruption, or other possible hardware component failure, through explicit notifications. The explicit notifications assist TCP senders to retransmit the lost segments without affecting *cwnd* for losses due to any reason other than network congestion. This category comprises the TCP implementations with explicit congestion notifications (ECN) or loss notifications (ELN) such as BTCP (Burst ACK/Burst Negative ACK) [98] and TCP with burst contention loss notification (TCP-BCL) [72][75].

2.5.3 Congestion Detection without Explicit Notifications

In TCP congestion detection where no explicit notification is deployed, TCP maintains and analyzes a number of previous RTTs at the TCP senders in order to determine if a packet-loss event is due to persistent congestion or random-burst contention. Such sender-side congestion-control schemes include Burst TCP with Burst Length Estimation (BTCP with BLE) [98], and our work on Burst AIMD (BAIMD) [73], Statistical AIMD (SAIMD) [71][74], and Threshold-based TCP Vegas [76][77][78]. TCP Selective Acknowledgment (SACK), Duplicated SACK (DSACK), and TCP Forward acknowledgement (FACK), also work over OBS networks without explicit signalling [98][99]. These schemes were not originally proposed for OBS networks. However, their throughput performance is examined in Chapter 3 to gain a deeper understanding of their behaviour over OBS networks. Since SAIMD and the threshold-based TCP Vegas were originally proposed for OBS networks, they are discussed in dedicated chapters.

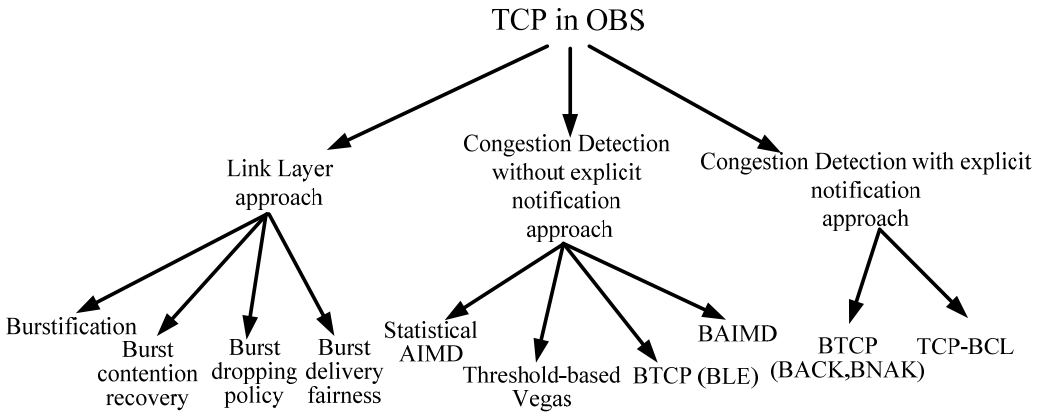

Figure 2.5 Taxonomy of TCP solutions over OBS networks

In the case of fast TCP flows, a contended burst that is delivered successfully is subject to additional delay due to the contention-resolution mechanism. The additional delay is inevitably added to the RTT for all the TCP segments assembled in that burst. The sender detects a sudden increase in the RTT of the segments, and interprets the delay as an indication of network congestion. This fatally impairs the TCP throughput due to unnecessary TO retransmissions followed by slow start at the TCP senders. Threshold-based TCP Vegas, a sender-side TCP implementation that does not require any explicit notifications, was proposed to cope with these problems. Figure 2.5 illustrates TCP solutions appropriate or designed for OBS networks.

2.6 Overview of Existing Solutions

This section surveys dropping-based (*e.g.*, Reno), delay-based (*e.g.*, Vegas), and explicit–notification-based (*e.g.*, ELN/ECN) TCP implementations in the presence of the link-layer solutions and the new TCP congestion-control schemes proposed for OBS networks.

2.6.1 Link-Layer Solutions

2.6.1.1 Solution Based on Burstification Processes

It has been shown that the delay caused by burst assembly increases TCP RTT which potentially reduces TCP throughput and leads to the slow convergence problem [14][19][99].

In [19], the authors examined the performance of TCP Reno in the presence of various burst-assembly algorithms at OBS edge nodes. The authors showed that for medium to slow TCP flows, significant performance degradation is induced because the assembly delay becomes very large with respect to the inter-arrival times of the TCP segments. On the other hand, the aggregation of multiple TCP segments in fast TCP flows is less influenced by the assembly delay.

Cao *et al.* [14] proposed an Adaptive-Assembly-Period (AAP) algorithm and investigated the impact of the algorithm parameters (burst assembly time, burst size, and the adaptive-assembly queue-length thresholds) on the performance of both TCP and UDP traffic over OBS networks. They showed that the adaptive-assembly algorithm helps TCP to achieve the best throughput performance.

The simulation-based study in [32] took TCP Reno as a dominant TCP, and examined various burst-assembly delays and burst sizes. The study concluded that TCP Reno fails to deal with burst losses when each burst contains a large number of TCP segments assembled from a single TCP flow.

2.6.1.2 Burst Contention Recovery

As one of the link-layer solutions, BCR can be implemented in the OBS layer in order to reduce random burst-contention losses, thereby improving the transmission reliability of OBS networks. BCR schemes include burst retransmission [104][105], burst deflection routing [37][68][105], FDLs [18], and burst segmentation [83]. Among the three schemes, FDLs achieve BCR by providing limited buffering time for each contended burst. In general, 1 *ms* of buffering time requires a fiber with 200 *km* of length. The buffered bursts suffer from signal dispersion and require signal amplification, which complicates the core node architecture and functionality. On the other hand, burst retransmission and deflection also introduce significant delay.

With burst retransmission, an OBS edge node stores a copy of each launched burst for possible retransmission. As the control packet traverses the core nodes, an intermediate node that fails to reserve the resource notifies the OBS edge. Upon receiving the failure notification, the edge node retransmits the stored copy of the contending burst, preceded by

the corresponding control packet. The retransmitted burst experiences extra retransmission delay, between the initial control packet transmission and the last notification received for the corresponding burst. If the network is lightly loaded, the retransmission scheme has a good chance of successfully delivering the contended bursts without involving the TCP retransmission mechanisms. The studies in [104] and [105] showed that the retransmission scheme can reduce the burst-loss probability compared with barebone OBS at low traffic loads. The authors also showed that retransmitting lost bursts from edge nodes prevents dropping-based TCP from falling into false congestion detection. However, when the network is heavily loaded, the retransmitted bursts may still be dropped and finally lead to a timeout at the TCP senders. Therefore, the maximum number of retransmission attempts for a single burst and the timeout threshold at the senders must be defined precisely and subject to further research.

With burst deflection, a burst is first routed through its primary path. In the event of burst contention, the burst is dynamically rerouted and redirected to an alternate path segment starting at the core node where the burst encounters contention. Since the primary path is usually the shortest path, data bursts following an alternate path segment suffer from longer propagation delay [37][68]. The study in [104] investigated burst retransmission along with deflection routing, and showed that burst deflection can reduce burst-loss probability significantly at low traffic loads. As with burst retransmission, additional delay caused by burst deflection affects delay-based TCPs. If the network is heavily congested, deflection may worsen the congestion situation by consuming more resources at other network switches, which leads to fatal impairment of TCP throughput.

2.6.1.3 TCP with Burst Acknowledgement

In [21], a TCP throughput model is introduced that incorporates the burst acknowledgment mechanism. The authors propose an error-recovery mechanism for electronic buffering at the edge nodes in order to improve throughput. However, this mechanism consumes extra memory space at the edge node for buffering a copy of all bursts for a certain time. Since the amount of additional memory can be large, the scheme is subject to practical implementation problems. For example, let the edge node switching capacity be c, the number of IP packets

be k, the average assembling granularity be M, and the burst dropping probability be p. The required extra memory space for buffering the bursts is $c \times (RTT_{burst} + p \times (RTT_{burst} + k \times M))$, where RTT_{burst} includes the burst assembly time and the burst offset time. Furthermore, the scheme cannot prevent TCP senders from TO or TD indications since *cwnd* can be decreased in response to the longer buffering delays.

2.6.1.4 TCP Decoupling

As we mentioned earlier, explicit congestion notifications enable network devices to signal congestion explicitly to the TCP sender, and hence allow the sender to decouple its congestion control from segment-loss recovery. In [95], a TCP decoupling approach is introduced. In this approach, the edge node controls the burst-contention probability at the OBS network bottleneck link by taking advantage of the TCP self-clocking property: a new segment is injected into the network only after an old one has left. The authors proposed using a TCP management packet, called TCP decoupling packet, to control the burst transmission rate. The rate is controlled through the arrival of the TCP decoupling packets. In this scheme, a virtual circuit (VC) is established for each source-destination pair in the OBS network. The VC is then controlled by the TCP congestion control located at the OBS edge node such that the sending rate never exceeds the link capacity. The OBS edge node uses the TCP ACK packets to control the timing of the burst transmission.

Given the simulation parameters provided in [95], the authors demonstrated an improvement to TCP throughput by avoiding unnecessary burst losses. The overall link utilization increased from 50% to 62% and the packet dropping probability decreased from 50% to 30%. However, this approach requires maintaining a record of the launched bursts, burst launch time, and the corresponding TCP segments for each source and destination pair. This approach complicates the OBS edge node architecture and functionality by manipulating the TCP packets in the OBS network.

2.6.1.5 Fixed-Point Feedback

The fixed-point approach that incorporates TCP's feedback mechanism aims to compute the expected steady state TCP input rate under a certain network load. In OBS, the fixed-point

approach consists of four phases. First, the TCP input rate is determined. Second, the corresponding burst distribution is computed. Third, the resultant OBS burst loss is obtained. Finally, the expected TCP sending rate is calculated based on the burst loss obtained in the third phase. In [13], such a fixed-point method was adopted to compute TCP steady state throughput in OBS networks. The method works only for barebone OBS by taking advantage of a two-dimensional Markov chain [31] for modelling a single OBS link, where burst contention is the only source of burst loss. Through successively applying the TCP throughput function and the resultant OBS burst loss, a fixed-point rate of TCP over OBS can be reached. This method can compute the expected steady-state TCP input rate in the OBS network at the expense of the edge node exchanging explicit notifications between the edge and the core node, based on information such as the burst losses, RTT variation, the number of output wavelengths at each output port, *etc*. Currently, the fixed-point feedback mechanism is used for theoretical analysis rather than practical implementation at the TCP layer.

2.6.1.6 Retransmission-Count Based Dropping Policy (RCDP)

In [68] and [69] a dropping policy, called the Retransmission-Count Based Dropping Policy (RCDP), is introduced, which aims to improve TCP throughput. The dropping policy is deployed at the OBS edge node and takes the number of burst retransmissions attempts into consideration, where bursts that have been retransmitted less are dropped. Clearly, burst retransmission takes time that might eventually cause a timeout to be triggered at the TCP senders. The authors proposed to add a retransmission count (RC) field in the control packet with an initial value of 1. In the event of burst contention, the core node compares the control headers of the contending bursts and drops those with lower RC values. Bursts with larger RC values are given higher priority since their TCP packets have already experienced relatively longer delay. Once a burst is dropped, a corresponding negative acknowledgement (NACK) and the RC value are sent back to the OBS edge node, which then increments the RC field and retransmits the burst. The number of retransmission attempts follows a predefined retransmission policy. The study aims to increase the transmission chances of the

TCP packets which have experienced the longest time in the OBS network. However, the study has not addressed fairness of the policy. Also, the preemption of reserved resources by bursts with larger RC values may have a negative influence on the flows, if they have already launched the corresponding data burst.

2.6.1.7 Burst Delivery, Scheduling, and Fairness

In OBS, packets belonging to a flow that traverses many hops tend to have a higher chance of being dropped than packets in flows with fewer hops. Therefore, TCP flows with more hops may not be able to probe the available bandwidth as effectively as the ones with fewer hop. This raises a fairness trade-off between burst delay and losses. Fairness of burst delivery in OBS network is a complex issue that has been subject to extensive research efforts. A common problem of these studies that the schemes never consider the types of packets assembled in the data burst. In [88] a merit-based channel allocation algorithm using JET signalling is proposed. A merit value is defined to rank each burst according to the length of the route or the potential delay. In [59], a wavelength continuity approach is introduced. This scheme performs backward reservation similar to that defined in Multi-Protocol Label Switching (MPLS). Within this scheme, a control packet traverses from the source to the destination through the predefined path. The destination sends a reservation packet to reserve switching fabric at each intermediate core node backward in parallel with the assembly of the data burst at the source.

Link scheduling algorithms play a vital role in maintaining fairness among bursts. In [40], a batch scheduling algorithm is proposed to maintain optimal link scheduling in terms of the scheduled bursts, which reduces the burst losses. This approach takes into consideration the burst size, arrival time, and link status, to maximize the number of scheduled bursts. However, these batch scheduling algorithms did not consider the burst fairness factor while performing burst scheduling. In [106], two schemes for improving fairness in OBS are presented. The authors first proposed a weighting function for evaluating free channels based on the number of hops between source and destination edge nodes. The second approach enhances the random early discard (RED) scheme, where a burst that makes fewer hops has a higher dropping probability than one with more. In [50], the authors proposed a new channel-

reservation protocol called virtual fixed offset time (VFO). Unlike previously proposed scheduling algorithms, VFO schedules bursts in the order of their arrival instead of the order of the corresponding control packets, which increases the chances of burst delivery [49].

We observe that all these link scheduling or signalling algorithms demonstrated a reduction in the burst dropping probability, thus, increasing the overall network throughput. However, they introduced extra switching-architecture complexity and additional burst delay.

2.6.2 New TCP Congestion Control for OBS Networks

2.6.2.1 BTCP (BLE, BACK, BNACK)

The study in [98] investigated TCP false timeout detection due to random burst-contention loss under a wide range of traffic loads. Three solutions were proposed. The first, called BTCP with burst length estimation (BLE), is based on estimating the number of TCP packets assembled in the burst without the knowledge of the burst assembly algorithm deployed at the OBS edge node. In addition to the *cwnd*, a burst congestion window *burst_wd* is also maintained. If the first loss is a TD, then $burst\_wd = cwnd/2$ (*i.e.*, this is an indication that the packets are assembled over many bursts). However, if the first loss is a TO, then TCP sender first compares *cwnd* with *burst_wd*. If $cwnd \leq burst\_wd$ and $burst\_wd > 3$, the TCP sender considers this TO a false TO. It halves *cwnd* and performs fast retransmission for the missing segments. When $cwnd > burst\_wd$ or $burst\_wd \leq 3$, the sender considers this a true TO event and initiates normal TCP retransmission.

In the second approach, called BTCP with burst acknowledgement (BACK), each TCP packet is acknowledged by a TCP agent located at the OBS edge nodes. This approach can effectively prevent TCP from detecting false TO; however, the end-to-end TCP semantics are violated since ACKs reach TCP senders before the actual completion of packet delivery. The last approach in [98], called BTCP with burst negative acknowledgement, maintains a TCP agent at each OBS core node. Whenever a burst is dropped, the TCP agent disassembles the burst and sends a burst negative ACK (BNACK) to the corresponding TCP sender. The missing segments and the network congestion state are exchanged explicitly between the

TCP senders and the OBS core nodes. In general, reducing the extra control overhead and implementation complexity are challenges when attempting to deploy these solutions.

2.6.2.2 Burst AIMD (BAIMD)

The BAIMD [73] scheme is based on the framework of Generalized Additive Increase Multiplicative Decrease (GAIMD) [11][89][96] for *cwnd* adjustment. Two parameters are defined: α and β, for the additive increment and the reduction ratio for *cwnd* at each TCP sender. Unlike conventional TCP, where α and β are set to 1 and 0.5, BAIMD determines the two parameters dynamically in such a way that *cwnd* is increased by α segments for each acknowledged packet in a round and is decreased multiplicatively by β ($0.5<\beta<1$) for any packet-loss event.

With BAIMD, the sender is not notified explicitly of the burst assembly mechanism and the reasons for burst losses. Each sender treats a packet-loss event as a congestion loss. Obviously, this could lead to an overestimation of network congestion by incorrectly halving *cwnd* for every segment drop. To compensate for this overestimation, BAIMD senders determine the values of α and β dynamically using burst-level status (*i.e.*, the estimated traffic load in the OBS network) such that the summarized effect of the burst-drop event is estimated. The scheme aims to achieve the best throughput for the competing flows. For example, if a burst is lost when the network load is low, the lost packets are considered due to random burst contention, and the multiplicative factor is set to $0.5<\beta<1$. Otherwise, β is set to 0.5 when network load is heavy and the burst dropping is due to congestion.

One of the most important advantages of BAIMD is simplicity, as no burst-level window is maintained at the TCP senders, and no explicit notification specific to each launched TCP segment is exchanged between the OBS edge and the TCP senders. The scheme maintains clean separation between the control signalling at the TCP senders and OBS edge nodes. Most notably, BAIMD senders use the RTT of each launched segment along with the number of TOs as references for sensing the network load. For example, data bursts are subject to extra buffering delay at the OBS edge nodes in response to serious network congestion. Thus, RTT increases substantially. On the other hand, if a data burst is dropped due to random burst contention in barebone OBS, the RTT remains unchanged. Thus, it is considered an

indication of low network load. Figure 2.6 illustrates the functionality of the proposed scheme.

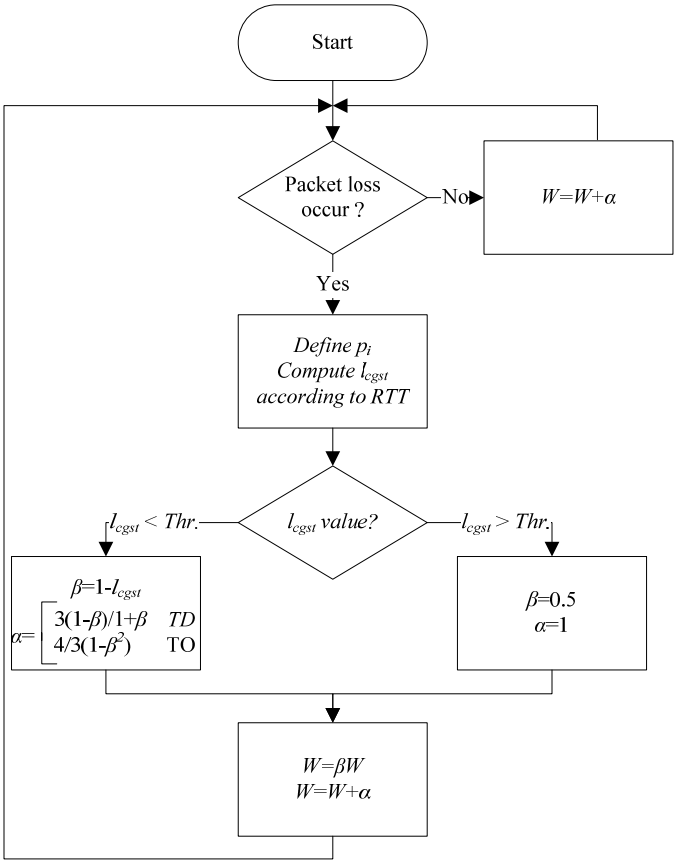

Figure 2.6 The BAIMD congestion control scheme

BAIMD estimates the multiplicative factor β through estimating the traffic load at the OBS layer. BAIMD defines a threshold load value l_{cgst} for computing the multiplicative factor. When a packet loss occurs at time t and all the connected TCPs as well as BAIMD are notified either through TD or TO, the maximum link capacity has been reached at that moment. In the congestion state (*i.e.*, the load is $l_{cgst} \geq 0.6$), the BAIMD senders behave similar to conventional TCP senders with $\beta = 0.5$. Otherwise, β is set to $\beta = 1 - l_{cgst}$ *where,* $0 < l_{cgst} < 0.6$. Once β is computed, BAIMD uses the GAIMD congestion control mechanism

to obtain the sending rate α as follows. In the presence of TD loss, $\alpha = 3(1-\beta)/(1+\beta)$, while in the case of TO loss, $\alpha = \frac{4}{3}(1-\beta^2)$ [11][89][96]. Table 2.2 summarizes the TCP over OBS schemes.

In the following chapter we present our methodology for evaluating TCP over OBS. This includes a detailed analysis on the used simulation topology, simulation parameters, performance evaluation assumptions, performance evaluation parameters, TCP modeling notations, and the TCP implementations examined.

Schemes	Solution Category	OBS Type	Devices Involved			Explicit Notifications	Problem Addressed	
			TCP source	OBS edge	OBS core		Random Burst Losses	Slow Convergence
Adaptive Assembly Period (AAP)	link-layer	Barebone BCR		√				√
Burst Retransmission	link-layer	BCR		√	√		√	
Deflection Routing	link-layer	BCR			√		√	
FDLs	link-layer	BCR			√		√	
Burst Segmentation	link-layer	BCR			√		√	
Wavelength Conversions	link-layer	BCR			√		√	
Retransmission-Count Dropping	link-layer	BCR		√	√		√	√
TCP with Burst Acknowledgement	link-layer	Barebone BCR	√	√	√	√	√	
TCP Decoupling	link-layer	Barebone BCR	√	√			√	
BTCP with Burst Length Estimation	without Signalling	Barebone BCR	√					
BTCP with Burst ACK	Signalling	Barebone/ BCR	√	√		√	√	
BTCP with burst NACK	Signalling	Barebone BCR	√	√	√	√	√	
Burst AIMD	without Signalling	Barebone BCR	√	√	√		√	√

Table 2.2 TCP solutions over OBS Networks

Chapter 3
Methodology

In this chapter we present the methodology used for studying TCP throughput over OBS networks through the rest of this thesis. We explain the modeling notation and the simulation parameters.

3.1 TCP Modeling Notation

In this section, we present the notation used in modeling TCP over OBS networks. In both Chapter 4 and Chapter 5, we have selected TCP SACK [54] as for our modeling since it has been widely deployed in current operating systems and has shown better throughput performance over OBS networks than TCP Reno and New Reno [98][99].

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 RTT without burst retransmission
RTT_r	:	Expected RTT with burst retransmission
b	:	Number of packets that are acknowledged by receiving an ACK
B	:	TCP throughput in packets per second
H	:	Expected number of packets submitted during a TO period
$\overline{RTT}$	:	average round trip time
TOP	:	TCP timeout period
TDP	:	TCP triple duplicate period

RTO	:	Retransmission timeout
Z^{TO}	:	Duration of a sequence of TOs
A	:	Duration of a sequence of consecutive successful rounds
X	:	Number of consecutive successful rounds
Y	:	Number of packets sent before TD or TO expiration
W	:	Current congestion window size in segments
W_0	:	TCP *cwnd* in Vegas stable state
W_m	:	TCP maximum window size
S	:	Number of segments belonging to a single TCP flow being assembled in the current burst
Q	:	Ratio between the probability of TO loss and TD loss
B_a	:	IP-access bandwidth
T_b	:	Burst assembly time

In the next section we present the simulation model including network topology, link capacity, burstification process, *etc*.

3.2 Simulation Model

In our simulation model, we used the *NS-2* network simulator [27][87] for evaluating TCP throughput performance over OBS networks. The NSF network topology is implemented as shown in Figure 3.1. The distances shown are in *km*.

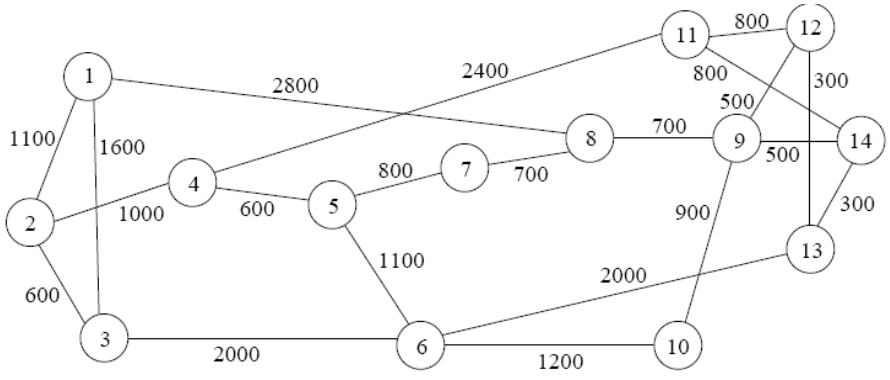

Figure 3.1 NSF OBS simulation topology

Data bursts are generated at the ingress edges (*i.e.*, ingress nodes are the OBS edge source nodes) and traverse through the core nodes with a minimum of four hops before reaching their egress destinations (*i.e.*, egress nodes are the OBS destination edge nodes). Depending on the number of competing TCP flows, the burst dropping probability varies between 10^{-5} and 10^{-1}. TCP flows are classified into fast, medium, and slow flows as in [19]. For a fast flow, all segments from the sending window are assembled into a single data burst. A burst loss causes fast flows to timeout. For a slow flow, one segment from the sending window is included in a given data burst. Therefore, the loss of the data burst results in a loss of a single packet. For a medium flow, the number of segments assembled into a single burst ranges between the fast and the slow flows. The burst loss causes medium flows to generate TD. In our simulations, TCP speed is controlled by the maximum congestion-window size (W_m), the TCP segment length (L), the IP-access bandwidth (B_a), and the burst assembly time (T_b) as follows,

$$\text{Fast flow} \qquad \frac{W_m . L}{B_a} \leq T_b \tag{3.1}$$

$$\text{Medium flow} \qquad \frac{L}{B_a} < T_b < \frac{W_m . L}{B_a} \tag{3.2}$$

$$\text{Slow flow} \qquad \frac{L}{B_a} > T_b \tag{3.3}$$

In this thesis, we focus on fast and medium flows, which are of more practical interest in high-speed networks with relatively high assembly time. Therefore, the mixed time/length-based burst assembly algorithm is used where the maximum burst length is set to 50 *KB* to generate fast and medium flows simultaneously. The TCP maximum congestion window ranges from 10 to 10^4 segments. The core nodes implement the Latest Available Unscheduled Channel with Voice Filling (LAUC-VF) channel scheduling algorithm [91]. One fiber link of 8 wavelengths operating at 10 Gbps is used between adjacent nodes, with a 10 *ms* propagation delay. One bi-directional control channel is allocated along each link. Control-packet processing time is set to 1 μs at both core and edge nodes. The offset time is set to 6 μs which is sufficient for bursts to traverse a minimum of 4 hops. A File Transfer

Protocol (FTP) is used for generating TCP traffic with a 1 *KB* average packet size. TCP throughput is obtained over a simulation period that ranges from 10^3 to 10^4 seconds.

The TCP senders and receivers are attached to the OBS edge nodes. Burst losses occur at the OBS core network due to burst contention. Burst retransmission and deflection routing have been implemented. In our simulation, we examined the TCP throughput over barebone OBS, OBS with burst retransmission, and OBS with burst deflection routing. If the contended burst contends again after the second retransmission attempt, the burst will be dropped. Similarly, if the deflected burst contends after reaching the second node, it will be dropped. In the simulation, RTT increases due to buffering delay at the edge nodes, burst retransmission, and burst deflection.

For the purpose of examining TCP fairness, we use *Jain's fairness index* which is defined as $(\sum_{i=1}^{n} B_i)^2 \Big/ n\sum_{i=1}^{n} B_i^2$, where n is the number of competing flows and B_i is the throughput of the i^{th} flow. The competing flows share the same source and destination. During simulation, the fairness index is obtained to compare flows with similar congestion-control implementations.

In the next section, we evaluate the throughput performance of TCP implementations under a range of burst dropping probabilities. This simulation study is essential to understand the throughput level in the presence of burst losses that affect multiple segments from a single TCP flow.

3.3 Performance Evaluation of Selected TCP Implementations over OBS

In this section, a number of TCP implementations and enhancements which target problems similar to the ones encountered in OBS networks are analyzed. Furthermore, an extensive simulation is performed to evaluate the throughput performance of TCP Reno, New Reno, SACK, DSACK, FACK, TCP with Eifel, BTCP, and BAIMD over the simulation model presented in Section 3.2.

3.3.1 Performance Evaluation Overview

In this section we give an overview of a number of previously reported TCP enhancements and their design principles/premises. TCP SACK detects the loss of multiple packets in a single round trip. The block information in an ACK allows the sender to retransmit the lost packets at once. TCP New Reno [28] is similar to SACK, in which partial ACKs are exchanged between the sender and destination to indicate the loss of multiple packets in one transmission round. TCP with duplicated SACK (DSACK) [26] reports the reception of duplicated packets so that the sender is informed constantly of which segments are received at the destination. With TCP Forward Acknowledgment (FACK) [55], the TCP sender maintains the number of packets received by the receiver through the segment sequence number. Burst TCP (BTCP) [98], TCP Eifel [52][53], TCP with Explicit Loss Notification [1], and TCP with Explicit Congestion Notification [1] were developed to identify suspicious timeouts that cause unnecessary retransmissions. Furthermore, TCP ELN/ECN distinguishes among packet losses due to congestion, contention, link failure, and other reasons.

The TCP implementations which address the problem of slow convergence, such as Fast TCP [36][39], High Speed TCP (HSTCP) [79], and Scalable TCP (STCP) [24] are kept for future work since they are not in the scope of this thesis.

3.3.2 TCP Implementations Not Designed for OBS Networks

Figure 3.2 shows the results of the competition among Reno, New Reno, SACK, DSACK, and FACK, in terms of throughput under barebone OBS. It is clear that TCP SACK, FACK, and DSACK out-perform Reno and New Reno due to the fact that the senders are designed to handle the dropping of a whole block of data segments in a single round, which matches the requirements of OBS well.

New Reno achieved the lowest throughput since it halves *cwnd* every time a segment is lost and successfully retransmitted, which could lead to dramatic reduction of *cwnd* in a single round. This implies that the more TCP Reno packets are assembled in a single burst, the lower the Reno throughput in the presence of burst dropping.

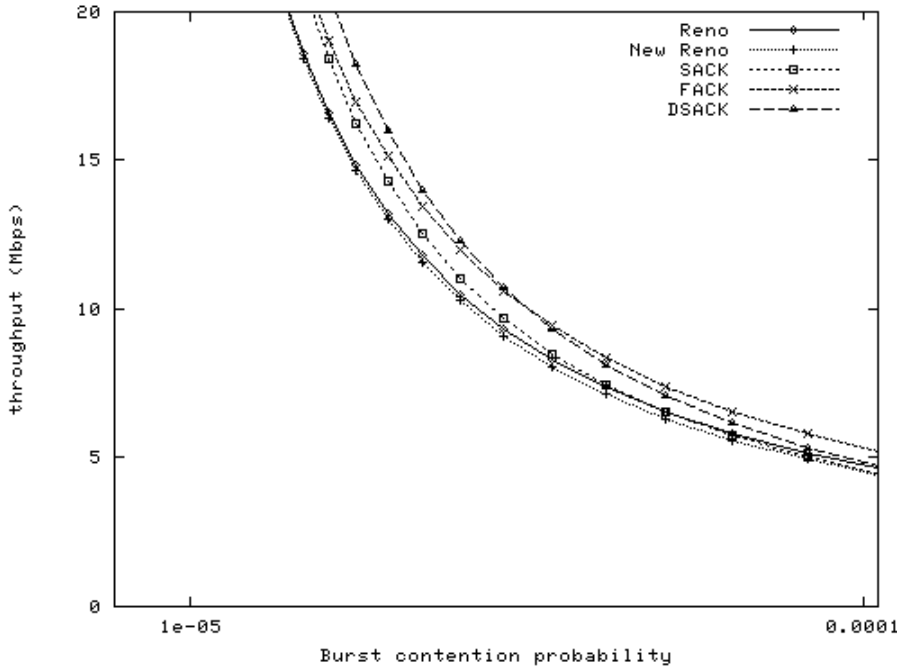

Figure 3.2 TCP Reno, New Reno, SACK, and DSACK throughput in barebone OBS

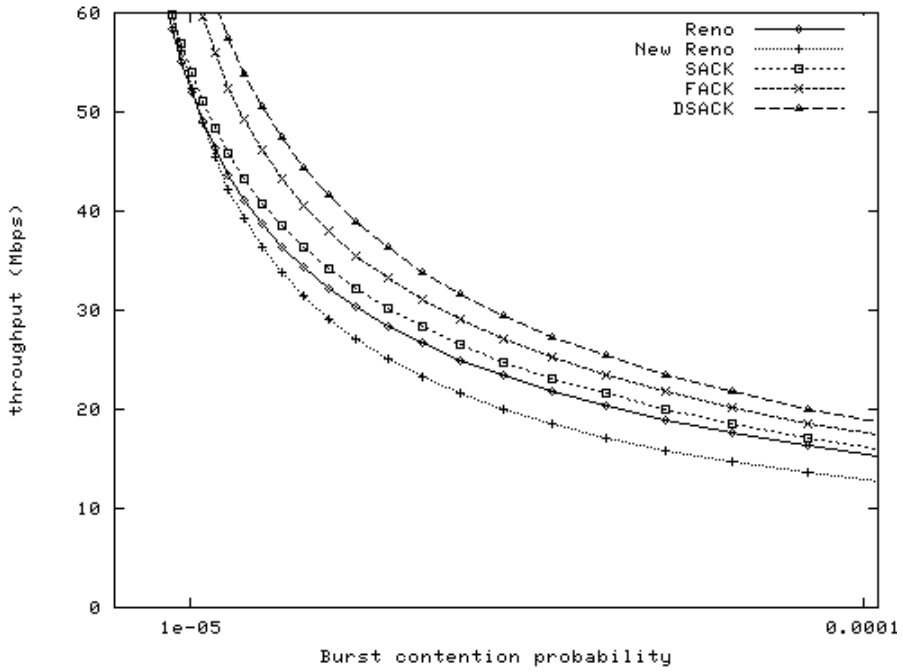

Figure 3.3 TCP Reno, New Reno, SACK, and DSACK throughput in OBS with burst retransmission

From Figure 3.2 we observe that FACK outperforms SACK since it maintains explicit measurements of the total number of packets outstanding in the network. TCP FACK senders maintain two variables called *snd.fack* and *retran_data* representing the forward-most packets received by the receiver at the transmission and the retransmission phases. With the two variables, the senders can determine the exact segment block that has been dropped in the network or received by the receiver. In contrast, SACK and Reno can only estimate the number of packets still in the network by assuming that each TD received is for only one segment. Since FACK determines the exact number of lost packets in the network, it can exploit the full network capacity, while recovering from segment loss and mitigating rate fluctuation, by sending a chunk of segments at once. Given these features, FACK may be the best fit to OBS networks since a data burst may contain a block of FACK segments or only one segment. In either situation, TCP FACK senders handle the correct number of missing segments with the most appropriate adjustment to the sending rate.

In order to examine the performance of the TCP implementations in the presence of out-of-order burst delivery, we enabled burst retransmission at the ingress nodes. If the contended burst fails at the second retransmission attempt, it is dropped. Figure 3.3 shows the performance of TCP Reno, New Reno, SACK, DSACK, and FACK in response to out-of-order delivery in OBS with burst retransmission. With DSACK, the TCP senders can undo the halving of *cwnd* when receiving the second copy of a segment caused by out-of-order burst delivery. From the figure, DSACK has the best ability to recover from the out-of-order delivery [5][21][66].

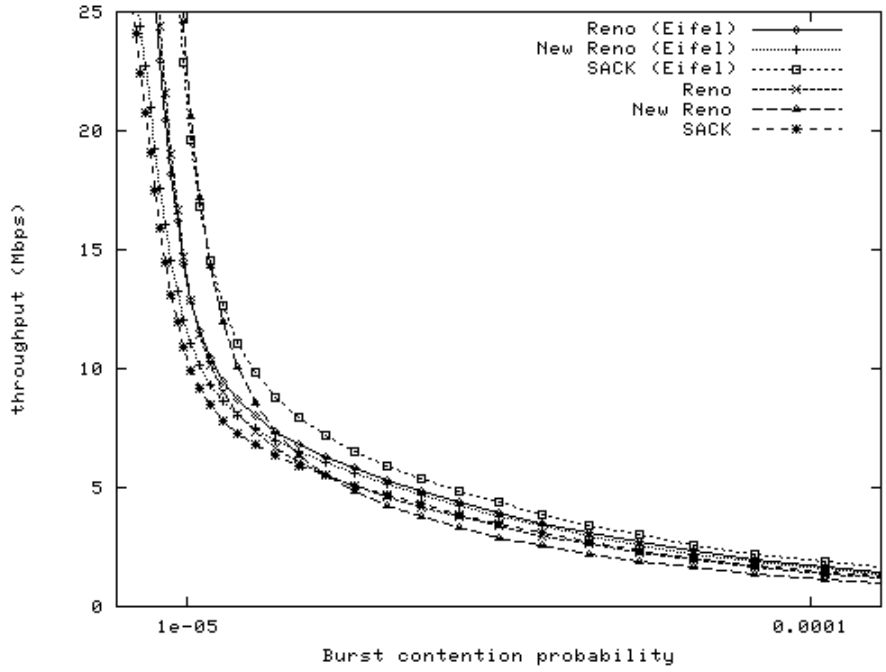

Figure 3.4 Reno (Eifel), New Reno (Eifel), and SACK (Eifel) throughput over barebone OBS

We tested the effect of integrating the Eifel algorithm [67] with traditional TCP implementations for detecting suspicious timeouts. In this experiment, a TCP timestamp facility is enabled [38]. In order to observe the response of the Eifel algorithm upon any suspicious timeout due to sudden packet delay, we chose to delay intentionally every TCP packet launched in the network at the 100^{th}, 600^{th}, and 800^{th} seconds of the simulation time for 100 *ms*, 200 *ms*, and 100 *ms* respectively at core node 8 in Figure 3.1.

Figure 3.4 and Figure 3.5 show the comparison among TCP Reno (Eifel), New Reno (Eifel), and SACK (Eifel) in barebone OBS and OBS with retransmission, respectively. It is clear that the Eifel algorithm has successfully improved the performance of Reno, New Reno and SACK by identifying the retransmission ambiguity due to the inserted delay events.

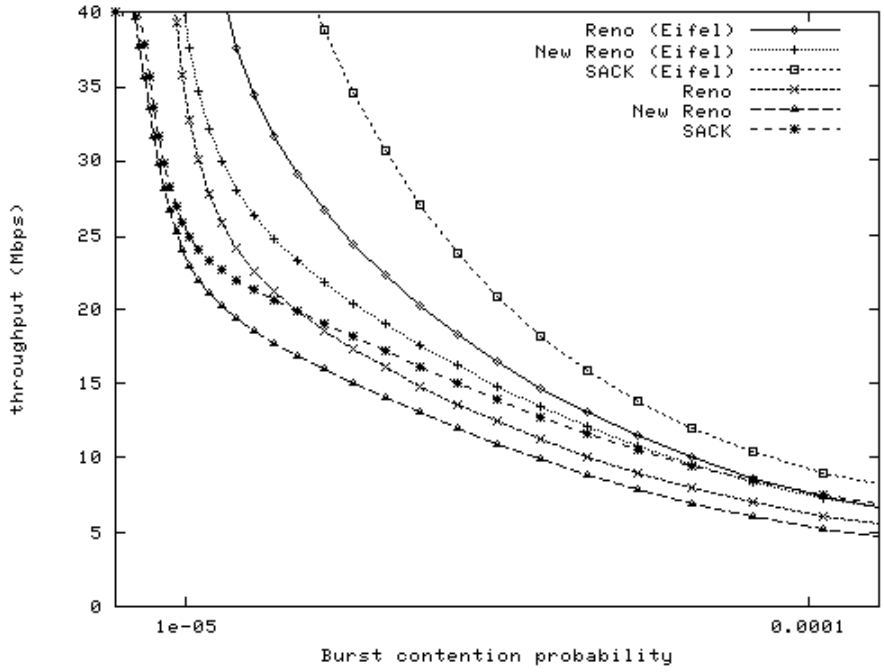

Figure 3.5 Reno (Eifel), New Reno (Eifel), and SACK (Eifel) throughput over OBS with retransmission

Among TCP enhancement schemes not specifically designed for OBS, TCP SACK, FACK, and DSACK outperform all the other schemes by best addressing the issue of multiple segment losses, at the expense of being unable to identify suspicious timeouts. We also observe that the integration of Eifel can greatly improve the throughput performance for the three TCP implementations by identifying suspicious timeouts caused by sudden delays and out-of-order delivery.

3.3.2.1 TCP Implementations Designed for OBS Networks

We also evaluated and compared the throughput performance of the recently proposed TCP implementations designed for OBS networks, including BAIMD [73] and BTCP BACK/BNACK [98]. TCP Reno and SACK are taken for comparison since they achieve higher throughput than New Reno in OBS networks [98]. Figure 3.6 shows the throughput of TCP Reno, SACK and BAIMD under barebone OBS and OBS with retransmission. Under a wide range of burst-dropping events (*i.e.*, due to a mix of random burst contention and

network congestion), BAIMD achieves the best performance among all traditional TCP implementations. Recall that BAIMD derives dynamically β corresponding to the congestion-burst dropping rate to achieve a stable *cwnd* adjustment upon each burst loss event.

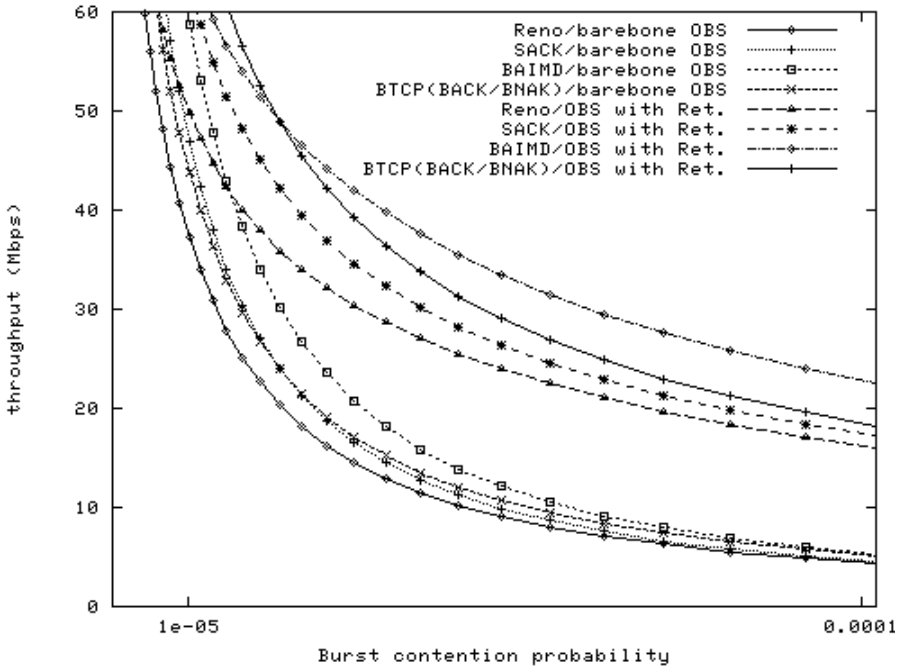

Figure 3.6 TCP Reno, SACK, BTCP (BACK/BNACK), and BAIMD throughput over barebone OBS and OBS with burst retransmission

In Figure 3.6, we also examine BTCP with BACK/BNACK and BAIMD. With BTCP, a TCP agent is placed at the ingress node to enable the explicit loss notification facility and ACKs for the BTCP senders. The simulation results show that BTCP yields slightly better performance than TCP Reno, New Reno, and SACK, but is still much worse than BAIMD. We observe that BAIMD outperforms all the other schemes due to its ability to identify the network status intelligently when a burst is lost and determine the corresponding transmission parameters α and β to achieve a stable sending rate and relevant congestion control.

Table 3.1 summarizes the TCP implementations examined and the resultant throughput performance over OBS networks. The simulation results show that conventional TCP implementations fail to maintain a high throughput level in the presence of burst losses that contain multiple segments from a single TCP flow.

From the TCP variants listed in Table 2.2 and Table 3.1, we observe that the most important characteristics and abilities that a congestion-control mechanism in TCP over OBS should have are following: (1) the ability to handle multiple TCP segment-loss events in a single round trip, (2) the ability to identify suspicious TOs and burst losses under low traffic load, and (3) the ability to compensate for out-of-order delivery in the OBS layer. The following chapters investigate these characteristics of TCP congestion-control mechanisms over OBS. In this context, novel TCP congestion-control mechanisms are proposed to deal with various burst-transmission characteristics.

Schemes	**Suspicious TOs**	**Packet reordering**	**Multiple packet losses in a single RTT**
Reno (Eifel)	√		
New Reno (Eifel)	√		
SACK (Eifel)	√		√
DSACK		√	√
FACK			√
BTCP (BACK, BNACK)	√	√	√
BAIMD	√	√	√

Table 3.1 An overview of TCP implementation performance over OBS

Chapter 4

Statistical AIMD Congestion Control for TCP over OBS Networks

This chapter introduces a novel congestion-control scheme for TCP over OBS networks, called Statistical Additive Increase Multiplicative Decrease (SAIMD). SAIMD maintains and analyzes a number of previous RTTs at the TCP senders in order to identify the confidence with which a packet-loss event is due to network congestion. The confidence is derived by positioning short-term RTT in the spectrum of long-term historical RTTs. The derived confidence corresponding to the packet loss is then used by the policy for congestion-window adjustment. SAIMD only requires statistical information about RTTs measured at a TCP sender, which achieves a clean separation between the TCP and OBS layers. We show through extensive simulation that the proposed scheme can solve the false-congestion-detection problem effectively and outperform conventional TCP significantly. Also, based on the proposed congestion-control algorithm, a throughput model is formulated and verified through simulation.

4.1 Overview

When OBS is deployed as the underlying switching technology, TCP congestion control is subject to great challenge. A number of TCP implementations have been proposed for detecting and controlling network congestion in various network environments, including mobile wireless networks [16], ad hoc networks [17], and optical networks [23][44][93]. Each TCP enhancement has its own design premises, and could be very effective in one circumstance while being much outperformed in another. It has also been proven that joint consideration of the characteristics of the whole network environment in the design of the TCP modifications or extensions is necessary [2]. These facts are especially salient when TCP is extended to OBS networks, where multiple TCP segments can be lost when the OBS network is not congested.

The GAIMD scheme proposes to reduce the "saw-tooth" behaviour of TCP for multimedia applications [11][96]. Instead of increasing *cwnd* by one for successful packet delivery and

decreasing it by half for a packet-loss event, GAIMD increases *cwnd* additively by α segments when no packet is lost in a single round-trip, and decreases multiplicatively by β if a TD packet-loss event occurs. In order to ensure TCP friendliness among competing GAIMD flows, α and β can be set as explained in [11][96]. Since SAIMD aims to address the unnecessary reduction of *cwnd*, we focus on the multiplicative decrease parameter β, while α is set to 1.

4.2 SAIMD over OBS Networks

The SAIMD scheme adopts the framework of GAIMD to enhance the responsiveness of TCP to any burst-loss event that is not caused by congestion. In SAIMD, when a data burst is lost, the corresponding TCP senders will be notified by a TD or TO. In either case, instead of halving *cwnd* or even throttling to slow-start stage, TCP senders reduce *cwnd* by the multiplicative factor β. The factor β is determined dynamically by positioning the short-term RTT statistics in the spectrum of long-term historical RTTs. Here, "statistics" refers to mean, standard deviation, and correlation function as described below.

SAIMD introduces two parameters, *M* and *N*. The parameter *M* is the number of consecutive RTTs measured for the long-term statistics. *M* should be sufficiently large that the derived statistics (*i.e.*, the mean and standard deviation) represent the intrinsic characteristics of the network topology, routing policy, and traffic distribution/pattern. The parameter *N* is the number of consecutive RTTs measured prior to a packet loss for the short-term statistics. The average of the *N* RTTs, denoted by *avg_RTT_N*, is compared with the average of the *M* RTTs, denoted by *avg_RTT_M*, in a TCP session, in order to determine whether the packet-loss event is due to network congestion or due to random burst contention in a lightly-loaded OBS network. In a packet-loss event caused by random burst contention, *avg_RTT_N* is expected to be close to *avg_RTT_M*. A larger *avg_RTT_N* suggests that a packet-loss event is more likely due to network congestion.

The relationship between *avg_RTT_M* and *avg_RTT_N* is based on the following observations. In TCP over OBS networks, packet loss is caused by random burst-contention losses in the OBS core or network congestion in the IP access networks or OBS core. The

difference between random burst contention and network congestion is that network congestion is high resource utilization for a longer period. In the high resource-utilization state, the RTT of each packet is much higher than that in the low-utilization state. This is due to the longer queuing delay in the IP access network. Also, in an OBS core with contention-resolution schemes such as burst retransmission [104][105] and deflection [37][68], bursts will have a higher probability of being retransmitted or deflected, which results in a longer average burst delay.

We further quantify the relation between the long-term and short-term statistics in order to define the confidence with which a packet loss is due to network congestion. We assume that the M consecutive RTTs are random with a mean avg_RTT_M and a variance $Var(RTT)$. We also assume they can be modeled approximately as a Normal distribution. To validate this assumption, we analyze 14,000 consecutive TCP RTTs resulting from running a simulation for 10^3 seconds, with a Chi-square test under the following two network scenarios: one is a barebone OBS network, where delay variation takes place in IP access networks and burst assembly; the other scenario is an OBS network with burst deflection, where delay variation takes place in the IP access networks, burst assembly at the OBS edge nodes, and burst deflection due to longer routes. The *null hypothesis* in the Chi-square test is: "the distribution of the M RTTs cannot be modeled as normal $N(\mu,\sigma)$, where $\mu = avg\_RTT\_M$, and $\sigma = \sqrt{M \cdot Var(RTT)}$". The derived distributions in both network scenarios are shown in Figure 4.1. The simulation experimental parameters are given in Section 3.2. We found that the null hypothesis can be rejected at a 5% confidence level, which validates our assumption that the M consecutive RTTs can be modeled approximately as a Normal distribution.

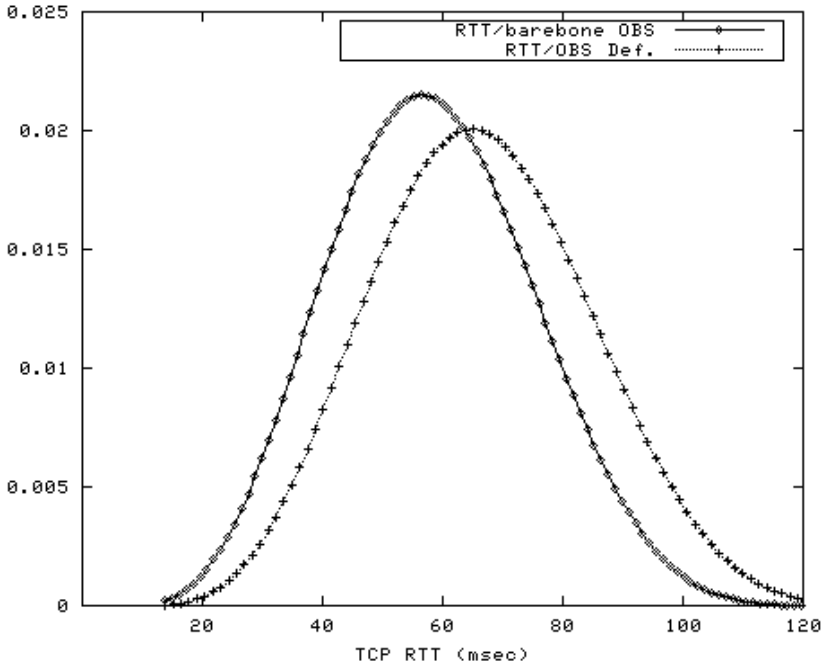

Figure 4.1 TCP RTT distribution histogram

4.2.1 Autocorrelation for Determining a Proper Value of *N*

Selecting a proper value of N is important, since the N RTTs are expected to provide sufficient information about the short-term network status when a packet is lost. If N is chosen too small or too large, the short-term network status may not be represented accurately. Our approach in selecting N employs an autocorrelation function with effective sample data size:

$$R(0,N) = E[RTT(0)\,.\,RTT(0+N)]$$
$$= \frac{1}{M} \cdot \sum_{i=0}^{N} RTT(i) \cdot RTT(i+N) \quad , N > 0$$

where $RTT(i)$ is the RTT of the *ith* packet. The autocorrelation function reflects how well a set of random variables are related to each other. Figure 4.2 shows the numbering of RTTs. The autocorrelation function can reflect the smoothness of the process. $R(0,N)$ has the maximum value when $N = 0$. Also, the stronger correlation within a group of RTTs, the larger the value of $R(0,N)$ outcome [33]. In our scheme, the value of N is selected such that $R(0,N) = R(0,0) \cdot \gamma\%$, where γ is the first-order autocorrelation threshold that determines

N based on the autocorrelation function. In other words, we benefit from the autocorrelation property to determine the effective sample size N, where $N = M\frac{(1-\gamma)}{(1+\gamma)}$.

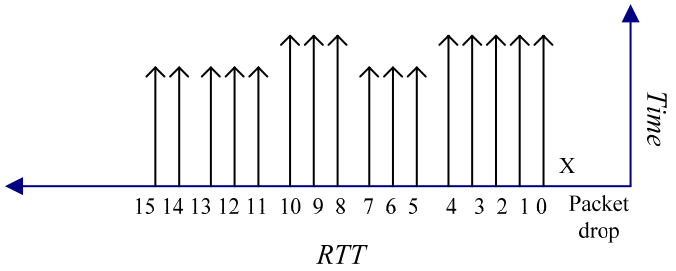

Figure 4.2 Numbering of RTTs, where RTT(0) denotes the RTT right before a packet-loss occurs

In order to represent the short-term network status well, N RTTs should have a strong correlation with each other. Hence, the value of γ should be close to 1. In our study, γ is 90%.

4.2.2 SAIMD Congestion Control

After N is selected based on the approach in the previous subsection, the value of *avg_RTT_N* can be obtained. Then, we define the confidence with which the current packet-loss event of a TCP session is due to network congestion by positioning *avg_RTT_N* in the Normal distribution spectrum of the M RTTs (shown in Figure 4.1). The derived confidence is used to adjust β dynamically at a TCP sender so that it represents the current network status.

For positioning *avg_RTT_N* in the Normal distribution spectrum, a function $z_i = RTT_{conf}(u_i)$ is defined, where u_i is the confidence level. The $RTT_{conf}(u_i)$ returns an RTT value (denoted by z_i) which is larger than a proportion u_i ($0 < u_i \leq 1$) of all RTTs in the Normal distribution curve. A one-to-one mapping between u_i and z_i exists, as shown in the following expression:

$$u_i = cdf(z_i) = \sum_{j=0}^{i} pmf(z_j)$$

$cdf(z_i)$ and $pmf(z_i)$ denote the cumulative density function and probability mass function in the RTT spectrum given the RTT value of z_i [33]. The one-to-one mapping between u_i and z_i is shown in Figure 4.3. In this figure, for example, if the RTT is higher than the mean RTT with u_i = 90% confidence, then the RTT value of z_i is in the range of 100 to 120 *ms*.

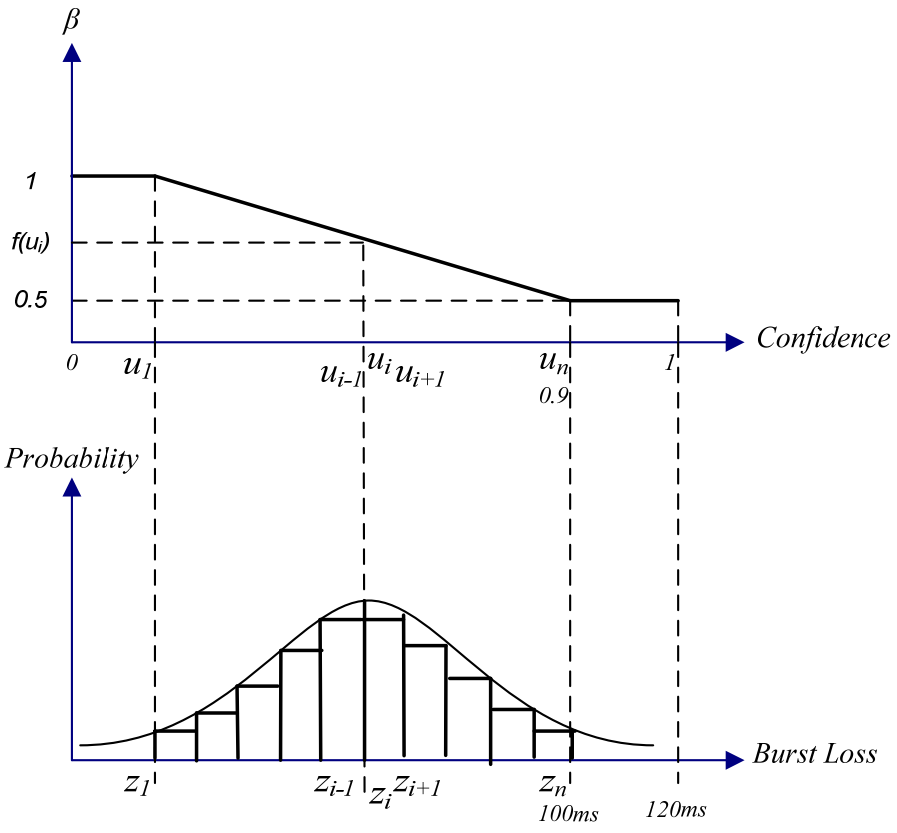

Figure 4.3 The relation of z_i, u_i, and β in the SAIMD scheme.

The proposed policy for adjusting β is as follows. When *avg_RTT_N* is smaller than $z_1 = RTT_{conf}(u_1)$, a low confidence of network congestion is indicated, which yields no adjustment of the *cwnd* in response to packet loss, (*i.e.*, β = 1). When

avg_RTT_N> $z_n = RTT_{conf}(u_n)$ ($u_n > u_1$), it is a strong indication of network congestion. Hence, the TCP sender cuts *cwnd* by half, (*i.e.*, $\beta = 0.5$) in response to a packet loss. When *avg_RTT_N* falls in the interval $[z_1, z_n]$, $\beta = f(u_i)$, where $f(u_i) = 1 - \frac{u_i - u_1}{2(u_n - u_1)}$. That is, we chose to set β according to the piecewise linear function of Figure 4.3, parameterized by u_1 and u_n. Note that u_1 and u_n are two parameters given in advance in order to distinguish network congestion from random burst contention in a lightly loaded OBS network. In this study, u_1 and u_n are set to 50% and 90%, respectively. The policy-based *cwnd* adjustment scheme can be summarized in the following equation:

$$\beta = \begin{cases} 0.5 & avg\_RTT\_N > RTT_{conf}(u_n) \\ f(u_i) & RTT_{conf}(u_n) > avg\_RTT\_N > RTT_{conf}(u_1) \\ 1 & RTT_{conf}(u_1) \geq avg\_RTT\_N \end{cases} \quad (4.1)$$

The dynamic adjustment of β based on the confidence level u_i is illustrated in Figure 4.3. The flow chart for SAIMD is shown in Figure 4.4.

We now discuss two extreme cases for SAIMD. Note that the adjustment of β will only be trigged if a TCP sender detects a TD segment loss. The first extreme case is that a TCP sender starts the data transmission while the network is congested. In this case, the measured RTTs are large at the beginning of the TCP session and the *avg_RTT_N* and *avg_RTT_M* obtained by the TCP sender are very close. Hence, β will be close to 1. For a TD segment loss, *cwnd* will not be reduced enough. SAIMD will then cause persistent congestion in the network and TO segment losses will occur. The TCP sender then enters a slow-start phase and sets *cwnd* to 1. As a result, the network congestion will be relieved. The second extreme case is when there is no RTT variation in a network, which is expected to be rare. In this case, the *avg_RTT_N* and *avg_RTT_M* obtained by the TCP sender are also very close. As a result, *cwnd* will not be reduced enough in response a TD. Instead, the TCP flow will timeout as a response to persistent congestion.

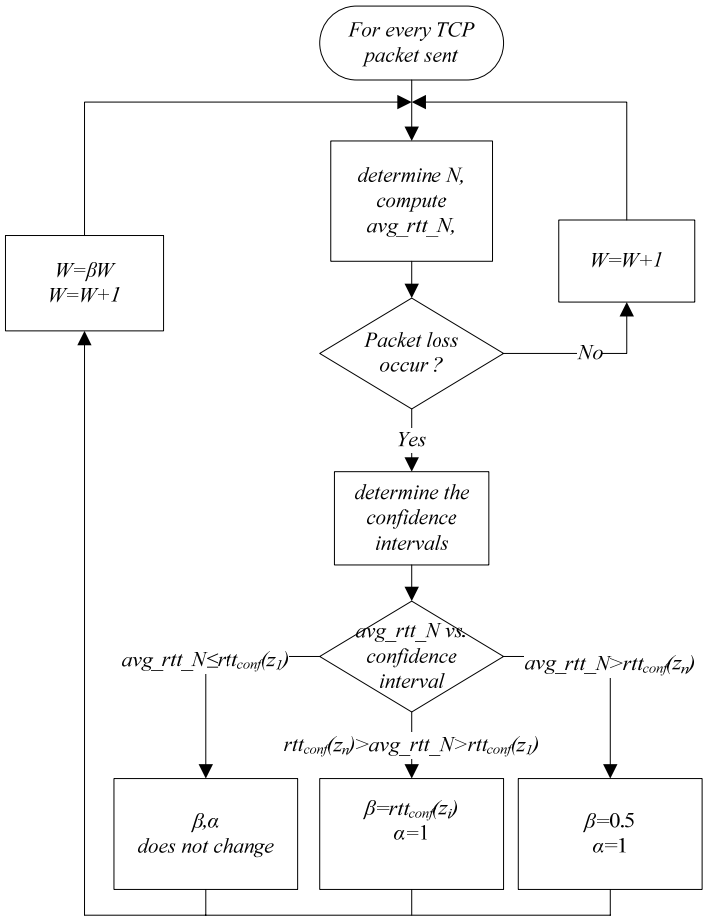

Figure 4.4 The SAIMD congestion control scheme.

SAIMD scheme is particularly suitable for high-bandwidth TCP flows that operate for a relatively long period of time with a large *cwnd*. These flows are expected to take an important role in applications such as grid computing. Depending on the number of TCP segments in a contended burst, these flows may either trigger a TO or cut *cwnd* in half as a response to receiving TDs. Once there is a false-congestion detection event which leads to TO or TD, the time required to increase *cwnd* (most likely additively) to its previous value could be very long, which impairs the TCP performance and the desired application scenario.

Compared with conventional AIMD-based TCP, SAIMD incurs additional overhead for maintaining the M RTTs and computing the autocorrelation and confidence intervals for the N RTTs. The cost can be traded off against the long convergence time in recovery from slow

start caused by false congestion detection. Faster recovery is essential for those high-bandwidth flows that may otherwise take hours to recover from a single TO. Note that the computation for the autocorrelation and confidence interval is required only when a segment loss event occurs, and the computation complexity is almost constant regardless of M and N. In addition, SAIMD is intended mainly for long and high-bandwidth TCP flows, rather than short ones such as those of HTTP web services; thus, the resultant additional overhead to the whole network is expected to be negligible.

One potential approach for enhancing SAIMD to cope with the above problems is to exchange information with the OBS domain. This information can accurately describe the network condition and the reasons for burst losses. In Chapter 5, we modify TCP congestion-control algorithm to exploit the advantages of explicit information exchange between TCP and the OBS domain.

4.2.3 Performance Analysis

In order to gain deeper understanding of SAIMD throughput performance, in this section we develop an analytic model for SAIMD over OBS networks. The model notations are presented in Section 3.1. In our model, we define a round as starting when the sender emits the current *cwnd* (in segments) and lasting until either it receives an acknowledgement or the TO expires. We also define a TD loss as a packet loss detected by triple duplicates and define a TO loss as a packet-loss detected after the sender times out.

We obtain the SAIMD TCP SACK throughput in an OBS network for both TD and TO losses by summing the number of packets sent during TD and TO periods, divided by the duration of the periods,

$$B = \frac{E[Y] + Q \times E[H]}{E[TDP] + Q \times E[TOP]} \tag{4.2}$$

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

4.2.4 Triple Duplicate (TD) Losses

As per the model in [60][99], suppose that the (c_i+1)th burst is the first burst lost in the ith TDP, TDP_i, and that the first lost segment is number (a_i+1). As shown in Figure 4.5, h_i additional segments will be sent in the same round after the (c_i+1)th burst is sent and lost. After receiving TD, the TCP sender retransmits all the missing segments contained in the lost burst in the next round. Therefore, in the next round, $W_{X_i} - S$ new segments will be sent, where W_{X_i} is the *cwnd* size in the X_i th round of TDP_i. After recovering all the segments lost in the burst, a new round TDP_{i+1} starts with *cwnd* cut by a factor of β.

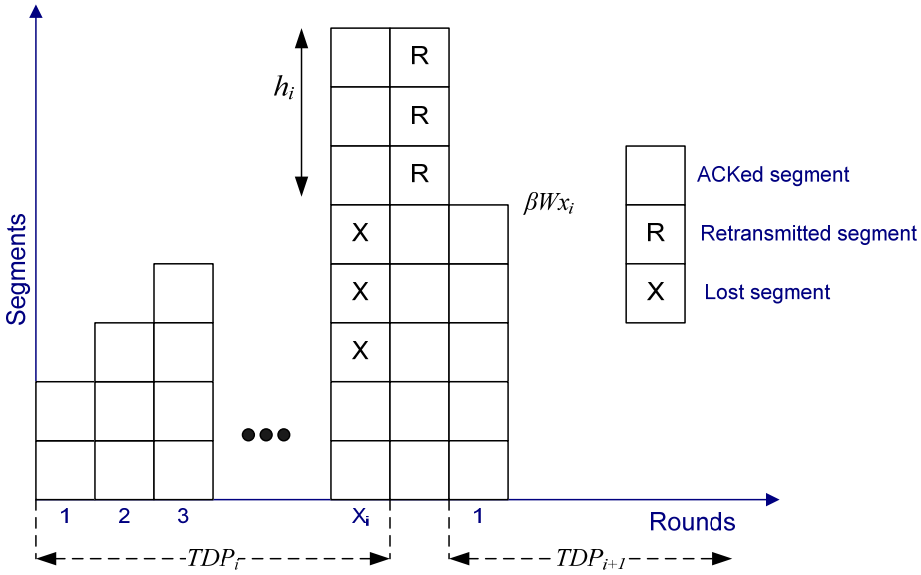

Figure 4.5 Evolution of SAIMD TCP SACK congestion window over OBS networks.

The total number of segments transmitted successfully during TDP_i is $Y_i = a_i + h_i + W_{X_i} - S$. $E[h]$ is approximately equal to $E[\beta]E[W_X]$, since $0 \le h_i \le W_{X_i}$ and the *cwnd* is reduced by β for every TD loss. Thus, we have

$$E[Y] = E[a] + (E[\beta] + 1)E[W_X] - S \tag{4.3}$$

where S is the number of segments belonging to the TCP flow assembled in the current burst. As per Equation 4.1, β is a function of the *avg_RTT_N* and the confidence level u_i. Considering Figure 4.5, we derive $E[\beta]$ as follows,

$$\begin{aligned} E[\beta] = \bar{\beta} &= cdf(z_1) + \sum_{i=1}^{n} f(z_i) \cdot pmf(z_i) + \frac{1 - cdf(z_n)}{2} \\ &= u_1 + \sum_{i=1}^{n} \left[1 - \frac{u_i - u_1}{2(u_n - u_1)} \right] \cdot (u_i - u_{i-1}) + \frac{1 - cdf(z_n)}{2} \\ &= u_1 + \frac{1 - u_n}{2} + \sum_{i=1}^{n} \frac{2u_n - u_1}{2(u_n - u_1)} (u_i - u_{i-1}) + \sum_{i=1}^{n} \frac{u_i (u_i - u_{i-1})}{2(u_n - u_1)} \end{aligned} \tag{4.4}$$

where $cdf(z_i)$ and $pmf(z_i)$ denote the cumulative density function and probability mass function in the RTT spectrum given the RTT value of z_i [33]. Since β has *pmf* p_β, then an alternative way for solving $E[\beta]$ can be through historical collection of the β values, which yields:

$$E[\beta] = \bar{\beta} = \sum_i \beta_i p_\beta(\beta_i) \tag{4.5}$$

where $p_\beta(\beta_i)$ is the probability of the distinct values β_i to exist in the random process.

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

$$P[c = k] = (1 - p)^{k-1} \cdot p \tag{4.6}$$

Given that a_i transmitted segments implies Sc_i transmitted bursts, $a_i = Sc_i$, we have,

$$E[a] = S\ E[c] = S \sum_{k=1}^{\infty} k(1 - p)^{k-1} p = \frac{S}{p} \tag{4.7}$$

By substituting Equation 4.7 into Equation 4.3, we have

$$E[Y] = (\overline{\beta} + 1)E[W_X] + \frac{1-p}{p}S \tag{4.8}$$

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

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

$$W_{X_i} = \beta W_{X_{i-1}} + \frac{X_i}{b} \tag{4.9}$$

By solving Equation 4.9 for X_i and talking the expectation, we have

$$E[X] = b(1-\overline{\beta})E[W_X] \tag{4.10}$$

Since Y_i can be derived by adding the numbers of segments sent in each of X_i successful rounds, plus the additional ($W_{X_i} - S$) segments in the next round of X_i as shown in Figure 4.5, we have:

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

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

By substituting Equation 4.9, we have

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

since the behavior and the size of W_X is depending largely on β, we assume a certain level of correlation existing between both variables. Thus, after substituting Equation 4.10, we get

$$E[Y] = \frac{b(1-\overline{\beta}^2)E[W_X]^2 - b(1-\overline{\beta})E[W_X]}{2} + E[W_X] - S \tag{4.11}$$

By combining Equation 4.11 and Equation 4.8, we have,

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

$E[W_X]$ can be then obtained as

$$E[W_X] = \frac{\bar{\beta} - \frac{b(\bar{\beta}-1)}{2} + \sqrt{(\frac{b(\bar{\beta}-1)}{2} - \bar{\beta})^2 + \frac{2Sb(1-\bar{\beta}^2)}{p}}}{b(1-\bar{\beta}^2)} \tag{4.12}$$

By substituting Equation 4.12 into Equation 4.8, we obtain $E[Y]$ as,

$$E[Y] = \frac{\bar{\beta} - \frac{b(\bar{\beta}-1)}{2} + \sqrt{(\frac{b(\bar{\beta}-1)}{2} - \bar{\beta})^2 + \frac{2Sb(1-\bar{\beta}^2)}{p}}}{b(1-\bar{\beta})} + \frac{1-p}{p}S \tag{4.13}$$

Also, by substituting Equation 4.12 into Equation 4.10, we obtain $E[X]$ as,

$$E[X] = \frac{\bar{\beta} - \frac{b(\bar{\beta}-1)}{2} + \sqrt{(\frac{b(\bar{\beta}-1)}{2} - \bar{\beta})^2 + \frac{2Sb(1-\bar{\beta}^2)}{p}}}{1+\bar{\beta}} \tag{4.14}$$

$E[TDP]$ is then obtained as

$$\begin{aligned} E[TDP] &= \overline{RTT}(E[X]+1) \\ &= \overline{RTT}(\frac{2\bar{\beta} - \frac{b(\bar{\beta}-1)}{2} + \sqrt{(\frac{b(\bar{\beta}-1)}{2} - \bar{\beta})^2 + \frac{2Sb(1-\bar{\beta}^2)}{p}}}{1+\bar{\beta}} + 1) \end{aligned} \tag{4.15}$$

4.2.6 For low burst losses ($W_X = W_m$)

For a very low burst loss probability, the *cwnd* size will most likely be at the maximum, W_m, before a burst loss event occurs. From Equation 4.8 we can obtain,

$$E[Y] = (\beta+1)W_m + \frac{1-p}{p}S \tag{4.16}$$

During each TDP, *cwnd* increases linearly from βW_m to W_m for $(W_m - \beta W_m)$ rounds and then stays at W_m for $(X_i - (W_m - \beta W_m))$ rounds, hence we can obtain the number of segments that are transmitted before a TD loss as $\frac{(W_m - \beta W_m)^2}{2} + W_m(X_i - W_m + \beta W_m) - h_i$. On the other

hand, from Equation 4.7, the total number of segments that are successfully transmitted before a packet-loss is *S/p*. Hence we have

$$\frac{(W_m - \beta W_m)^2}{2} + W_m(X_i - W_m + \beta W_m) - h_i = \frac{S}{p} \tag{4.17}$$

By solving Equation 4.17 for X_i and talking the expectation, we can obtain $E[X]$ as

$$E[X] = W_m(1-\bar{\beta}) - \frac{W_m(1-2\bar{\beta}+\bar{\beta}^2)}{2} + \bar{\beta} + \frac{S}{W_m p} \tag{4.18}$$

The duration of the *TDP* is obtained as,

$$\begin{aligned} E[TDP] &= \overline{RTT}(E[X]+1) \\ &= \overline{RTT}(W_m(1-\bar{\beta}) - \frac{W_m(1-2\bar{\beta}+\bar{\beta}^2)}{2} + \bar{\beta} + \frac{S}{W_m p} + 1) \end{aligned} \tag{4.19}$$

4.2.7 Timeout (TO) Losses

The behavior of TCP SAIMD for a TO loss is the same as that of TCP SACK. Hence, the analysis of TO losses is same as the analysis in [19]. From [99], we have

$$E[H] = E[R] - 1 = \frac{p}{1-p}, \tag{4.20}$$

$$E[TOP] = RTO\frac{f(p)}{1-p}, \tag{4.21}$$

where $f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6$. A TO occur when all bursts are lost in the last X_i round with probability $p(W_x)$. Since a TD is followed by a TO with probability $p(W_x)$ or followed by another TD with probability 1-$p(W_x)$, then,

$$QE[p(W_X)] = E[p(W_X)] = E[p^{\frac{W_X}{S}-1}] \approx p^{\frac{W_X}{S}-1}. \tag{4.22}$$

4.2.8 SAIMD TCP SACK over OBS Throughput Estimation

In the case of $W_X < W_m$, we can obtain the SAIMD throughput by substituting Equations 4.13, 4.15, 4.20, 4.21, and 4.22 into Equation 4.2, which yields

$$B = \frac{\frac{\bar{\beta} - \frac{b(\bar{\beta}-1)}{2} + \sqrt{(\frac{b(\bar{\beta}-1)}{2} - \bar{\beta})^2 + \frac{2Sb(1-\bar{\beta}^2)}{p}}}{b(1-\bar{\beta})} + \frac{1-p}{p}S + \frac{p^{\frac{W_X}{S}}}{(1-p)}}{\overline{RTT}(\frac{2\bar{\beta} - \frac{b(\bar{\beta}-1)}{2} + \sqrt{(\frac{b(\bar{\beta}-1)}{2} - \bar{\beta})^2 + \frac{2Sb(1-\bar{\beta}^2)}{p}}}{1+\bar{\beta}} + 1) + p^{\frac{W_X}{S}-1}RTO\frac{f(p)}{1-p})} \quad (4.23)$$

In the case of $W_X = W_m$, TCP SAIMD throughput can be obtained by substituting Equations 4.16, 4.19, 4.20, 4.21, and 4.22 into Equation 4.2, which yields

$$B = \frac{(\bar{\beta}+1)W_m + \frac{(1-p)S}{p} + \frac{p^{\frac{W_m}{S}+1}}{1-p}}{\overline{RTT}(W_m(1-\bar{\beta}) - \frac{W_m(1-2\bar{\beta}+\bar{\beta}^2)}{2} + \bar{\beta} + \frac{S}{W_m p} + 1) + p^{\frac{W_m}{S}-1}RTO\frac{f(p)}{1-p})} \quad (4.24)$$

4.3 Numerical Results

To verify the proposed Statistical AIMD scheme, simulation is conducted using the simulation parameters presented in Section 3.2. The mixed time/length-based burst-assembly algorithm is adopted, where the burst timeout threshold is 500 *ms* and the maximum burst length is 50*KB*. TCP W_m ranges from 10 to 30 segments. The access bandwidth is set to 100 Mbps. From Equations 3.1 - 3.3, the selected parameters generate fast and medium flows.

SAIMD is triggered after 30 RTT samples (*i.e.*, collecting 30 samples is sufficient to obtain a Normal distribution of RTTs). Burst retransmission and deflection routing have been enabled. In our simulation, we examined the TCP throughput over barebone OBS, OBS with burst retransmission, and OBS with burst deflection routing. In the simulation, RTT increases due to buffering delay at the edge nodes, burst assembly delay, burst retransmission, and deflection.

In Figure 4.6, we show the relationship between the average TCP RTT and the packet-loss probability. Since there is no significant RTT change resulting from higher burst loss probability, in this figure we show RTT values corresponding to a maximum of 10^{-3} burst

contention probability. It is notable that a significant amount of extra delay was incurred by the retransmitted and deflected bursts.

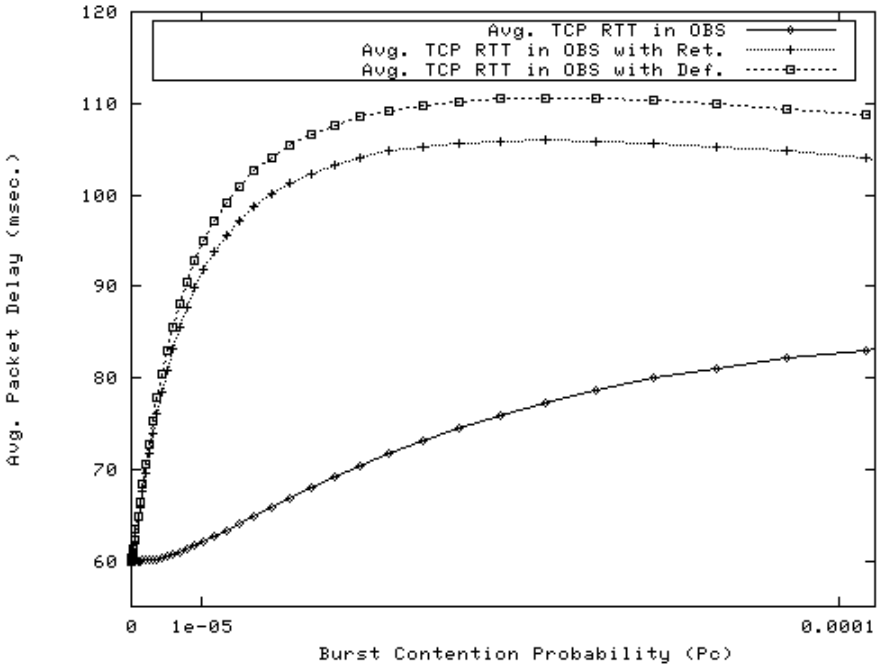

Figure 4.6 Average TCP RTT delay vs. the burst-contention probability

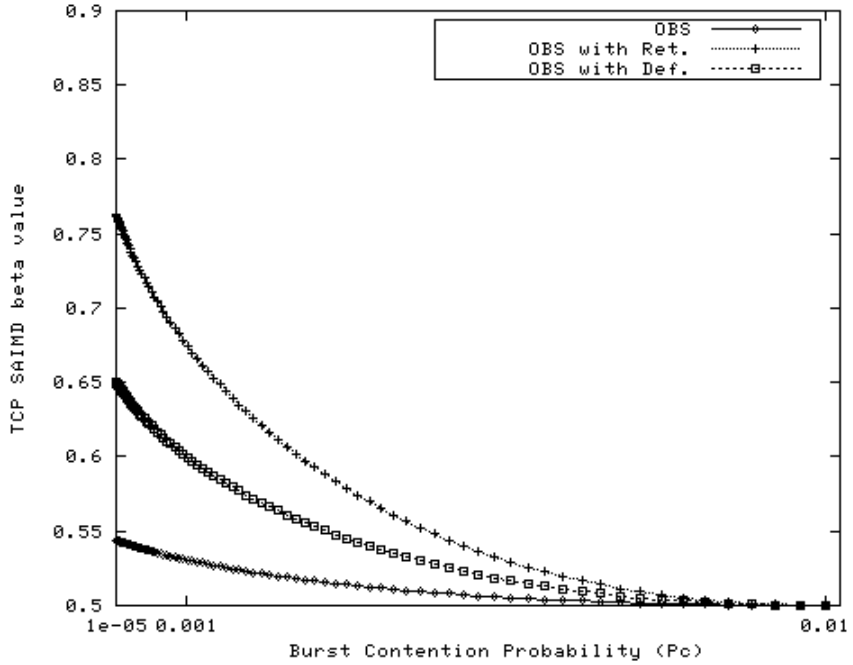

Figure 4.7 Burst-contention probability vs. SAIMD beta value

In Figure 4.7 we show the calculated β values for SAIMD at various packet-loss probabilities.

Figure 4.8 compares the results from the analytical model with those obtained by the simulation. The two agree closely.

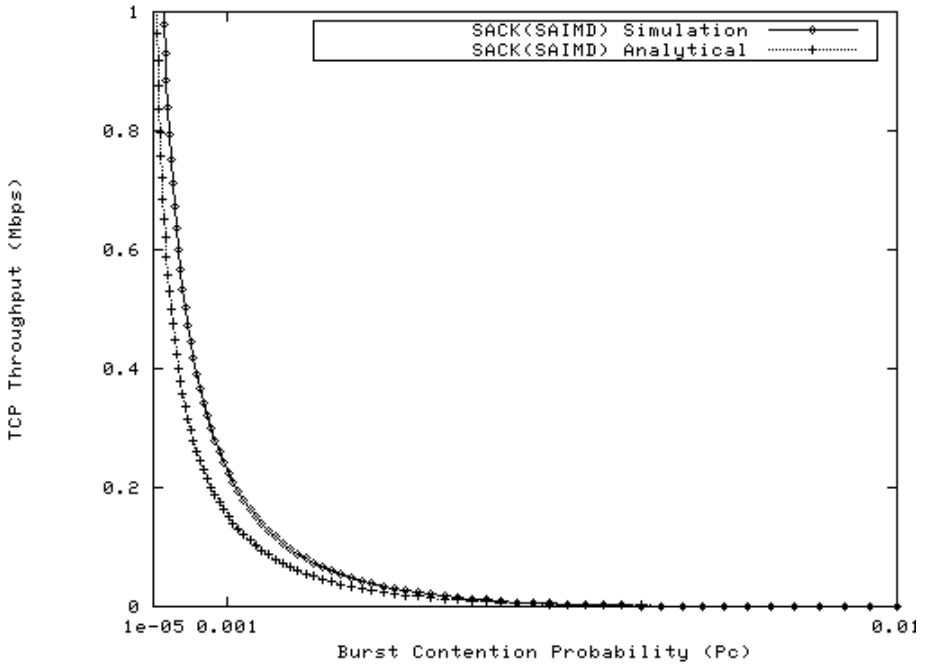

Figure 4.8 SAIMD simulation vs. analytical model throughput

Figure 4.9 shows the throughput performance of SACK/SACK(SAIMD) and Reno/Reno(SAIMD) flows in barebone OBS. The throughput of conventional TCP SACK and Reno is reduced dramatically even when the burst loss rates are quite low because the AIMD (1,0.5) senders always take a burst loss event as due to congestion, which may unnecessarily halve *cwnd*. On the other hand, the SAIMD SACK senders have achieved a 37% throughput increase because SAIMD does not react rigidly to a burst loss as a congestion loss at this low packet-loss probability. Instead, the factor β is manipulated in each TCP sender to guarantee a smaller *cwnd* reduction in response to any packet loss from a random burst contention. SAIMD enhances the TCP SACK and Reno throughputs by approximately 31%.

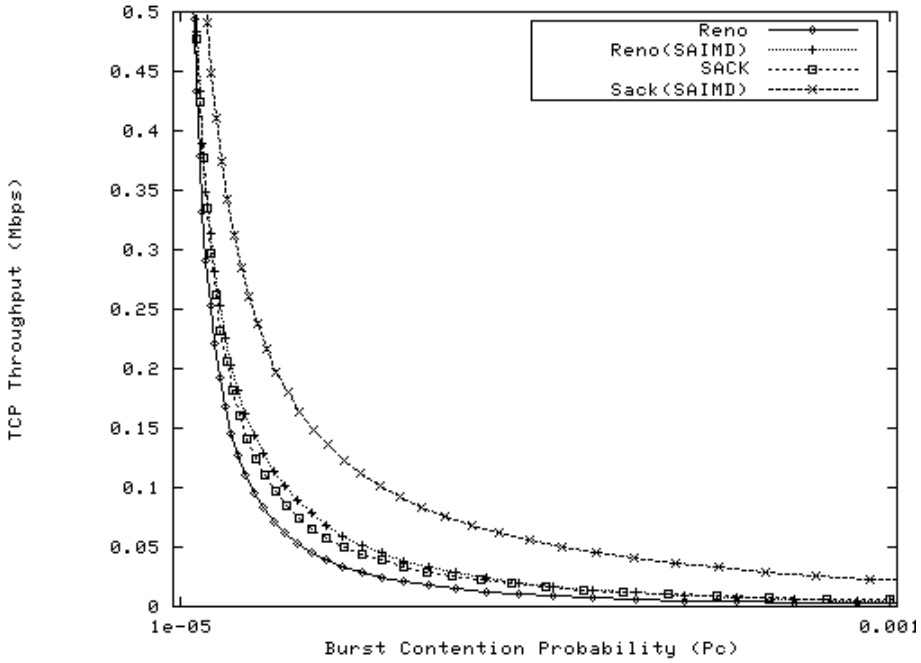

Figure 4.9 Throughput of TCP Reno/(SAIMD), TCP SACK/(SAIMD) in the barebone OBS

Figure 4.10 shows the throughput of TCP SACK/SACK(SAIMD) and Reno/Reno(SAIMD), with burst retransmission. Retransmission has successfully reduced the overall burst loss probability at the expense of longer RTT for TCP senders (as shown in Figure 4.6).

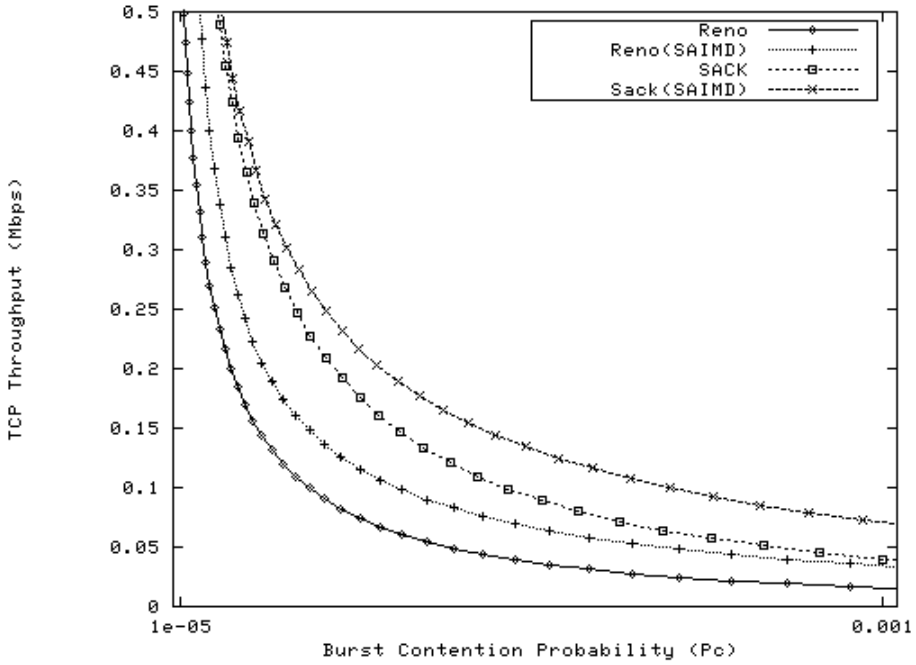

Figure 4.10 Throughput of TCP Reno/(SAIMD), TCP SACK/(SAIMD) in OBS with burst retransmission

Through integrating burst retransmission with SAIMD in the TCP flow-control process, we observe significant improvements in TCP SACK and Reno throughput of 81%. In Figure 4.11, we consider OBS with deflection routing. The simulation result shows that SAIMD achieves the best throughput performance. Using SAIMD can improve the TCP SACK and Reno throughput by 72% in the presence of burst deflection routing.

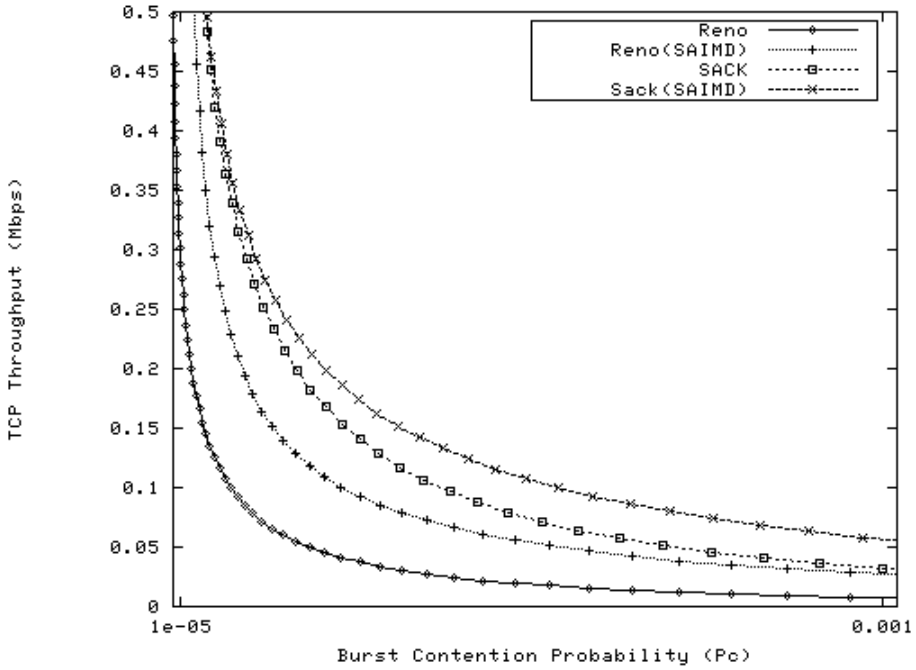

Figure 4.11 Throughput of TCP Reno/(SAIMD), TCP SACK/(SAIMD) in the OBS network with burst deflection

We also examine fairness among Reno, SACK, Reno (SAIMD), and SACK (SAIMD). Figure 4.12 shows the fairness index of TCP flows by Reno, SACK, and Reno (SAIMD), and SACK (SAIMD), over barebone OBS network. SAIMD has a much better fairness index than traditional AIMD in Reno and SACK. This is due to the fact that the SAIMD congestion-control mechanism has successfully and accurately identified burst contention and congestion, which better assists the SAIMD flows to remain close to the equilibrium.

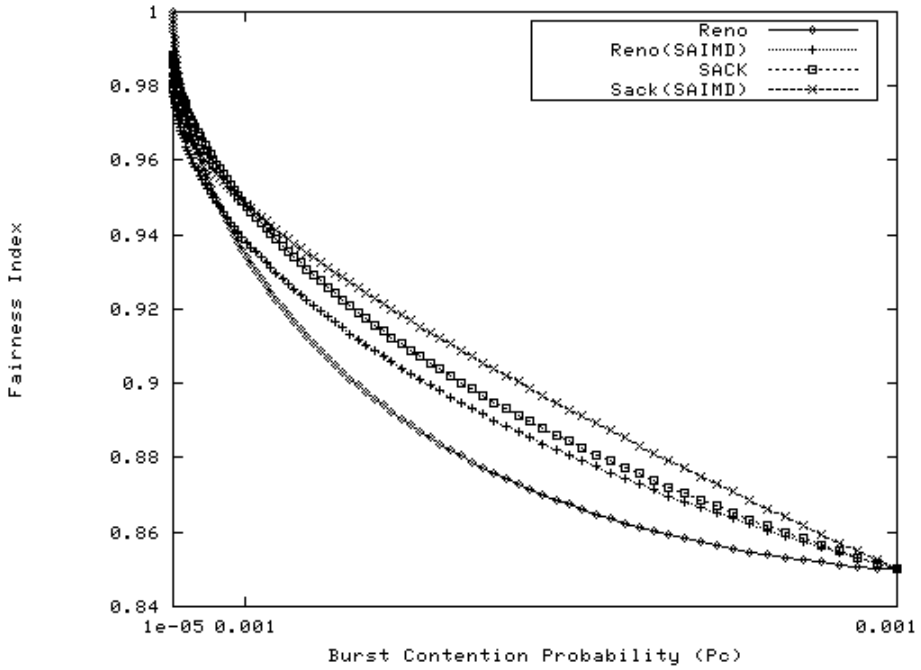

Figure 4.12 Fairness index of Reno, SACK, Reno (SAIMD), and SACK (SAIMD) in a barebone OBS network

The fairness index was also examined with retransmission, as shown in Figure 4.13. Note that burst retransmission is an effective approach for enhancing the overall network throughput by hiding burst-loss events in the OBS domain. The simulation results demonstrate that SAIMD fairness with burst retransmission is better than SACK and Reno. These two experiments prove that SAIMD can maintain a friendly relationship with the other TCPs.

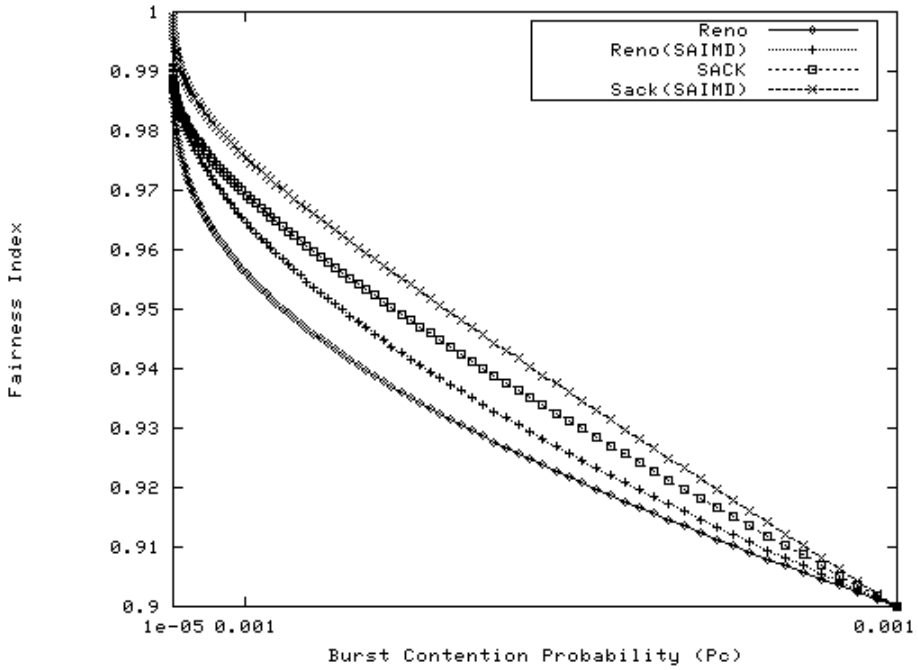

Figure 4.13 Fairness index of Reno, SACK, Reno (SAIMD), and SACK (SAIMD) in OBS with burst retransmission

4.4 Conclusion

In this chapter, a novel Statistical Additive Increase Multiplication Decrease (SAIMD) framework for TCP congestion control in OBS carrier networks is proposed and examined. SAIMD aims to mitigate the vicious effect of TCP false congestion due to the lack of buffering in the OBS domain. The scheme collects and analyzes historical RTTs and adjusts β according to the statistics of the collected RTTs at the occurrence of any segment-loss event. Analysis was conducted to evaluate the TCP throughput using the proposed scheme. Simulations were conducted to validate the proposed TCP throughput model and to evaluate the proposed congestion-control mechanism, by comparing it with conventional AIMD-based TCP Reno and SACK under different network scenarios, such as OBS networks with burst retransmission or burst deflection routing. Simulation results showed that the proposed SAIMD mechanism can outperform conventional TCP implementations significantly. We conclude that the superiority of SAIMD comes from a better understanding of the underlying

burst-transmission behaviour, through the analysis of collected RTT information. SAIMD is particularly beneficial to high-bandwidth and fast TCP flows, in which a false congestion-detection event caused by burst contention could lead to serious impairment of TCP performance.

With SAIMD, the sender does not require explicit knowledge of the burst assembly mechanism or the reasons for burst losses, which in some cases (recall Section 4.2.2) results in an inaccurate estimation of network congestion. In the following chapter, we exploit the advantages of information exchange between TCP senders and the OBS domain under various burst transmission conditions.

Chapter 5
TCP with Dynamic Explicit Burst-Contention Loss over OBS

In this chapter, a novel TCP congestion-control scheme with dynamic explicit Burst-Contention Loss notifications in OBS networks is proposed. The scheme, called TCP-BCL, aims to handle various OBS burst conditions that negatively affect TCP throughput performance and fairness. Based on GAIMD, the basic design principle is to tune the congestion control parameters α and β so that the congestion-window sizes in the corresponding TCP senders can be adjusted appropriately with an explicit notification from the OBS edge node.

5.1 Overview

We observed through the previous TCP performance evaluation studies that most TCP implementations have some limitations and are affected negatively by the bufferless nature of OBS. Based on the amount of asynchronous bursty bandwidth demand at each ingress and egress node pair, the scheme attempts to ensure that decreasing the *cwnd* occurs only in the event of true network congestion. The idea was used by [1] for wireless communication, where ELN achieves state leakage between the lower layer and the TCP layer, similar to the approaches proposed in [98]. With ELN, the cause of each burst loss is reported to the TCP sender, whether due to congestion or other link-transmission conditions. In this way, the TCP sender adjusts *cwnd* according to the network status of the lower layer. In this chapter, we solve the false-congestion detection problem in TCP over OBS networks and avoid unnecessary *cwnd* reduction by introducing a novel congestion-detection scheme that measures the utilization along each route (path) in the OBS network and adjusts *cwnd* based on the utilization and burst-dropping information carried in the explicit notification messages. This is the first study that integrates the explicit notification facilities with GAIMD over OBS networks.

5.2 Dynamic Explicit Burst-Contention Loss Notification (TCP-BCL)

TCP-BCL attempts to improve TCP performance over OBS networks without losing TCP fairness. In terms of the design premise and novelty, the scheme takes the best of BTCP [98] and BAIMD [73], where the GAIMD window-based congestion-control paradigm and the mechanisms of explicit notification and/or signalling between the TCP and OBS layers are both used.

5.2.1 OBS Congestion Identification

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

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

There are two possible approaches to detecting path congestion. The first is to delegate the congestion-detection process to the core nodes. The edge nodes receive explicit signals from the core switches indicating link congestion. A similar approach is proposed in BTCP with BACK/BNACK [98]. It is notable that this approach may not be very practical, since it increases the signalling and computation overhead at the core nodes. Therefore, in the following paragraphs, we introduce a novel mechanism for detecting congestion along an OBS route/path statistically, at the edge node. This approach does not introduce any additional signalling overhead at the core nodes. Path congestion is measured by how

congested the route in the OBS network is, which is an important index for the upper-layer TCP senders using the route to adjust their congestion windows.

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

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

To quantify the relationship between *avg_b_N* and *avg_b_M*, the OBS ingress edge first derives the spectrum of **θ**, which is a histogram of **θ** as shown in Figure 5.1; then the ingress edge positions the *avg_b_N* corresponding to the current burst-drop event in the spectrum in order to identify how likely it is that the route in the OBS domain is congested.

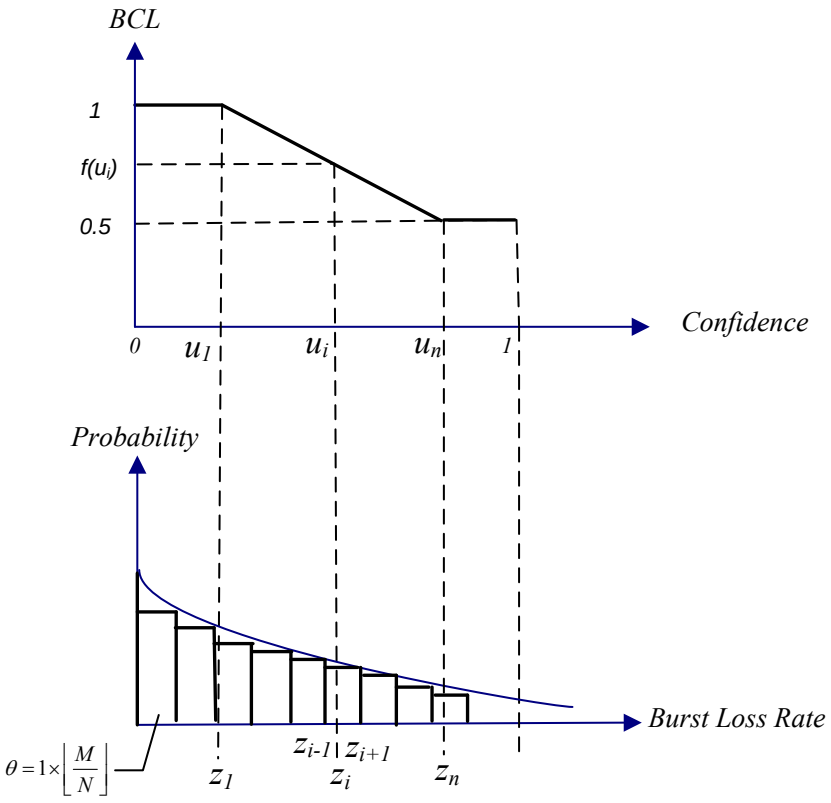

Figure 5.1. The relation of z_i, u_i, and BCL values in the edge node scheme

The following section explains the approach taken by the edge node to determine the burst loss as congestion or random contention loss.

5.2.2 OBS Edge Node Congestion Detection & Signalling

With *avg_b_N* and the spectrum formed by **θ,** we define the confidence with which the current burst-loss event is due to network congestion by positioning *avg_b_N* in the spectrum. A function $z_i = BL_{conf}(u_i)$ is defined, where u_i is the confidence level, and z_i is the

burst-drop rate corresponding to a confidence level of u_i ($0 < u_i \leq 1$) in the spectrum formed by θ. In other words, we have a one-to-one mapping between u_i and z_i as shown in the following expression:

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

Let the terms $cdf(z_i)$ and $pmf(z_i)$ denote the cumulative density function and probability mass function in the burst losses spectrum given the burst loss value of z_i, respectively. The dynamic determination of network condition based on the confidence level u_i by the edge node is illustrated in Figure 5.1.

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

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

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

The adjustment of BCL values based on the confidence level u_i is illustrated in Figure 5.1. The flowchart for the proposed TCP-BCL scheme is shown in Figure 5.2. The two parameters *M* and *N* are custom-designed, and in this study, *M* and *N* are chosen to be 500 and 10, respectively. In other words, the current network congestion status is determined by the position of the previous 10 burst deliveries in the spectrum formed by the previous 500 burst deliveries. Finding optimal values of *N* and *M* is an open problem that is left for future research. The scheme is expected to be stable since any increase of *avg_b_N* will be positively fed back by shifting the spectrum formed by θ farther to the right.

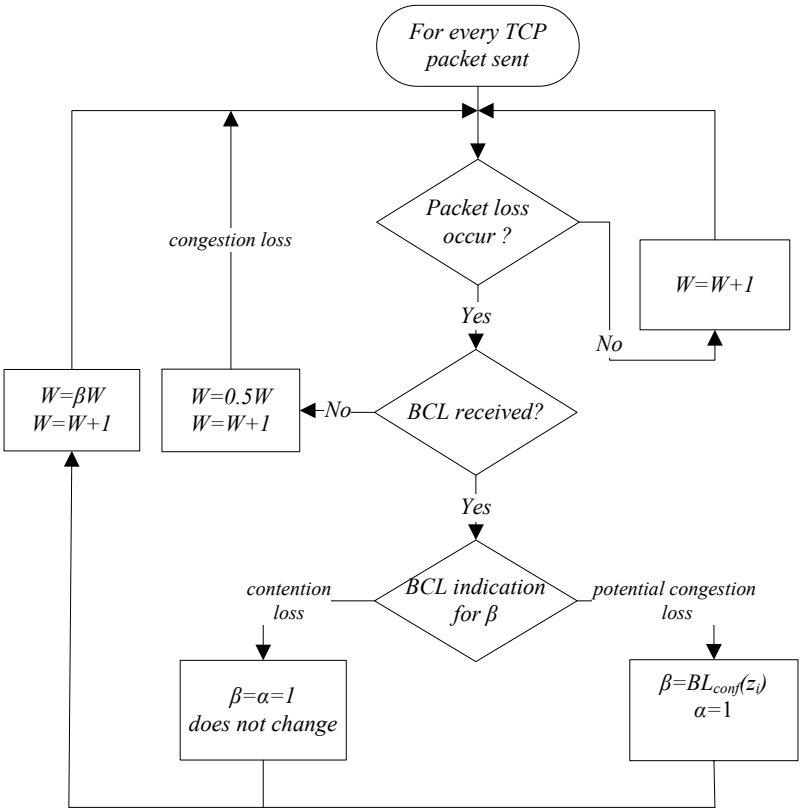

Figure 5.2 Flowchart for TCP-BCL congestion control scheme.

In order to enable explicit burst contention notification, an agent is placed at each OBS ingress edge node, and is responsible for determining the network status (such as link utilization) when a burst-drop event is encountered. The agent sends BCL notifications to the corresponding TCP senders if the burst-drop event is judged due to contention. In other words, TCP-BCL has the TCP senders take every segment-loss event as due to congestion when a BCL is not received. This distinguishes our scheme from BTCP in [98] since TCP-BCL only signals the dropped bursts at low link utilization, while with BTCP, the TCP agent reports every burst-drop event. Hence, the number of notifications in TCP-BCL is less than in BTCP. Furthermore, our design can significantly reduce the intra-domain signalling and core node processing overhead.

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

5.3 TCP-BCL Performance analysis

In this section, we analyze the throughput performance of the TCP-BCL flows in OBS networks. Once again, we obtain the TCP-BCL SACK throughput in OBS network for both TD and TO with Equation 4.2. It is worth recalling that TCP-BCL uses the GAIMD congestion control mechanism. The relation between α and β for ensuring friendliness among TCP flows in the case of triple-duplicate ACK*s* is obtained from [11] as:

$$\alpha = \frac{3(1-\beta)}{1+\beta} \tag{5.1}$$

while in the case of timeout,

$$\alpha = \frac{4}{3}(1-\beta^2) \tag{5.2}$$

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

5.3.1 TCP-BCL in Triple Duplicate (TD) Losses

Using Equations 4.2 to 4.8 derived in Section 4.2.3, in the following sections we derive the throughput model under high and low packet losses.

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

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

$$W_{X_i} = \beta W_{X_{i-1}} + \frac{\alpha X_i}{b} \tag{5.3}$$

Solving for X_i, we have

$$E[X] = \frac{b(1-\beta)E[W_X]}{\alpha} \tag{5.4}$$

Since Y_i can be derived by summing the number of segments sent in X_i successful rounds and the additional ($W_{X_i} - S$) segments in the next round of X_i as shown in Figure 4.5, we have:

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

By substituting Equation 5.3, we have

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

after substituting Equation 5.4, we get

$$E[Y] = \frac{b(1-\beta^2)E[W_X]^2 - b(1-\beta)E[W_X]}{2\alpha} + E[W_X] - S \tag{5.5}$$

By combining Equation 5.5 and Equation 4.8, we have,

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

$E[W_X]$ can be then obtained as

$$E[W_X] = \frac{\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{b(1-\beta^2)} \tag{5.6}$$

By substituting Equation 5.6 into Equation 4.8, we obtain $E[Y]$ as,

$$E[Y] = \frac{\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{b(1-\beta)} + \frac{1-p}{p}S \tag{5.7}$$

Also, by substituting Equation 5.6 into Equation 5.4, we obtain $E[X]$ as,

$$E[X] = \frac{\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{1+\beta} \tag{5.8}$$

$E[TDP]$ is then obtained as

$$\begin{aligned} E[TDP] &= \overline{RTT}(E[X]+1) \\ &= \overline{RTT}(\frac{2\alpha\beta - \frac{b(\beta-1)}{2} + \sqrt{(\frac{b(\beta-1)}{2} - \alpha\beta)^2 + \frac{2\alpha Sb(1-\beta^2)}{p}}}{1+\beta} + 1) \end{aligned} \tag{5.9}$$

5.3.1.2 For low burst losses ($W_X = W_m$)

For a very low burst loss probability, *cwnd* will most likely remain at the maximum W_m, before a burst loss event occurs. From Equation 4.8 we can obtain,

$$E[Y] = (\beta+1)W_m + \frac{1-p}{p}S \tag{5.10}$$

During each TDP, *cwnd* increases linearly from βW_m to W_m for $(W_m - \beta W_m)$ rounds and then stays at W_m for $(\alpha X_i - (W_m - \beta W_m))$ rounds, hence we can obtain the number of segments that are transmitted before a TD loss as $\frac{(W_m - \beta W_m)^2}{2} + W_m(\alpha X_i - W_m + \beta W_m) - h_i$. On

the other hand, from Equation 4.6, the total number of segments that are transmitted successfully before a packet loss is $\alpha S/p$. Hence we have

$$\frac{(W_m - \beta W_m)^2}{2} + W_m(\alpha X_i - W_m + \beta W_m) - h_i = \frac{S}{p} \tag{5.11}$$

Solving for X_i, we can obtain

$$E[X] = \frac{W_m(1-\beta)}{\alpha} - \frac{W_m(1-2\beta+\beta^2)}{\alpha 2} + \frac{\beta}{\alpha} + \frac{S}{\alpha W_m p} \tag{5.12}$$

The duration of the *TDP* is obtained as,

$$\begin{aligned} E[TDP] &= \overline{RTT}(E[X]+1) \\ &= \overline{RTT}\left(\frac{W_m(1-\beta)}{\alpha} - \frac{W_m(1-2\beta+\beta^2)}{\alpha 2} + \frac{\beta}{\alpha} + \frac{S}{\alpha W_m p} + 1\right) \end{aligned} \tag{5.13}$$

5.3.2 Timeout (TO) Losses

The behaviour of TCP-BCL for a TO loss is same as that of TCP SACK. Hence, the analysis of TO losses is same as the analysis in [99]. TCP-BCL transmits the same number of packets between two TOs as traditional TCP. However, for TCP-BCL, the packet retransmission starts as soon as the BCL is received, which is the RTT after the loss round. Thus, TCP-BCL waits for *TOP= RTT/p*. From [99], we have:

$$E[H] = E[R] - 1 = \frac{p}{1-p}, \tag{5.14}$$

$$E[TOP] = \frac{RTT(f(p))}{p(1-p)}, \tag{5.15}$$

where $f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6$, and similar to Equation 4.22

$$Q(E[W_X]) \approx p^{\frac{W_X}{S}-1}. \tag{5.16}$$

5.3.3 TCP-BCL SACK over OBS Throughput Estimation

In the case of $W_X < W_m$, we can obtain the TCP-BCL throughput by substituting Equations 5.10, 5.9, 5.14, 5.15, and 5.16 into Equation 4.2, which yields

$$B = \frac{\dfrac{\alpha\beta - \dfrac{b(\beta-1)}{2} + \sqrt{(\dfrac{b(\beta-1)}{2} - \alpha\beta)^2 + \dfrac{2\alpha S b(1-\beta^2)}{p}}}{b(1-\beta)} + \dfrac{1-p}{p}S + \dfrac{p^{\frac{W_X}{S}}}{(1-p)}}{\overline{RTT}(\dfrac{2\alpha\beta - \dfrac{b(\beta-1)}{2} + \sqrt{(\dfrac{b(\beta-1)}{2} - \alpha\beta)^2 + \dfrac{2\alpha S b(1-\beta^2)}{p}}}{1+\beta} + 1) + p^{\frac{W_X}{S}-1}\dfrac{RTT(f(p))}{p(1-p)})} \tag{5.17}$$

In the case of $W_X = W_m$, TCP-BCL throughput can be obtained by substituting Equations 5.9, 5.13, 5.14, 5.15, and 5.16 into Equation 4.2, which yields

$$B = \frac{(\beta+1)W_m + \dfrac{(1-p)S}{p} + \dfrac{p^{\frac{W_m}{S}+1}}{1-p}}{\overline{RTT}(\dfrac{W_m(1-\beta)}{\alpha} - \dfrac{W_m(1-2\beta+\beta^2)}{2\alpha} + \dfrac{\beta}{\alpha} + \dfrac{S}{\alpha W_m p} + 1) + p^{\frac{W_m}{S}-1}\dfrac{RTT(f(p))}{p(1-p)})} \tag{5.18}$$

5.4 TCP-BCL Numerical Results

In this section, we evaluate the throughput performance of TCP-BCL over OBS. We compare TCP-BCL throughout with BTCP and BAIMD as well as the TCPs that were originally designed for general packet-switched networks, including TCP Reno and SACK. The simulation parameters presented in Section 3.2 were used. The mixed time/length-based burst-assembly algorithm is adopted, where the burst timeout threshold is 100 *ms* and the maximum burst length is 50*KB*. TCP W_m is set to 128 segments. The access bandwidth is set to 100 Mbps.

5.4.1 TCP-BCL Numerical Results

Figure 5.3 compares the analytical results obtained from Equation 5.17 and Equation 5.18 and the simulation results obtained for TCP-BCL fast flows. It is notable that the simulation results match the analytical results.

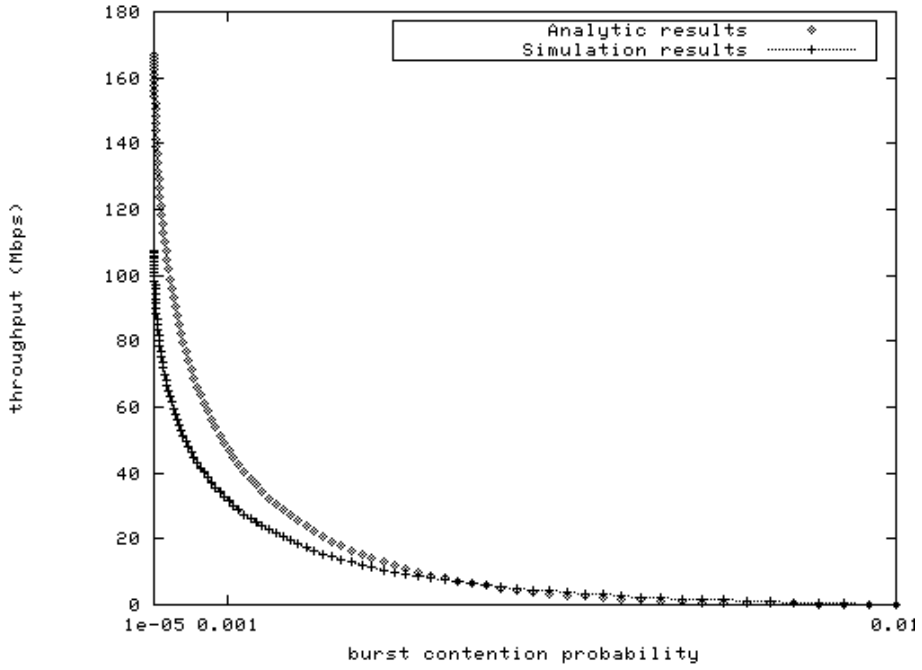

Figure 5.3 TCP-BCL throughput over OBS networks.

Figure 5.4 shows the throughput performance by TCP-BCL, TCP Reno and SACK under barebone OBS and OBS with burst retransmission. It is worth nothing that if the contended burst fails after the second transmission attempt, the burst will be dropped. BCL notifications are not sent to the TCP senders that have segments in contended bursts that are retransmitted successfully. It is clear that TCP-BCL outperforms the two conventional TCP schemes significantly.

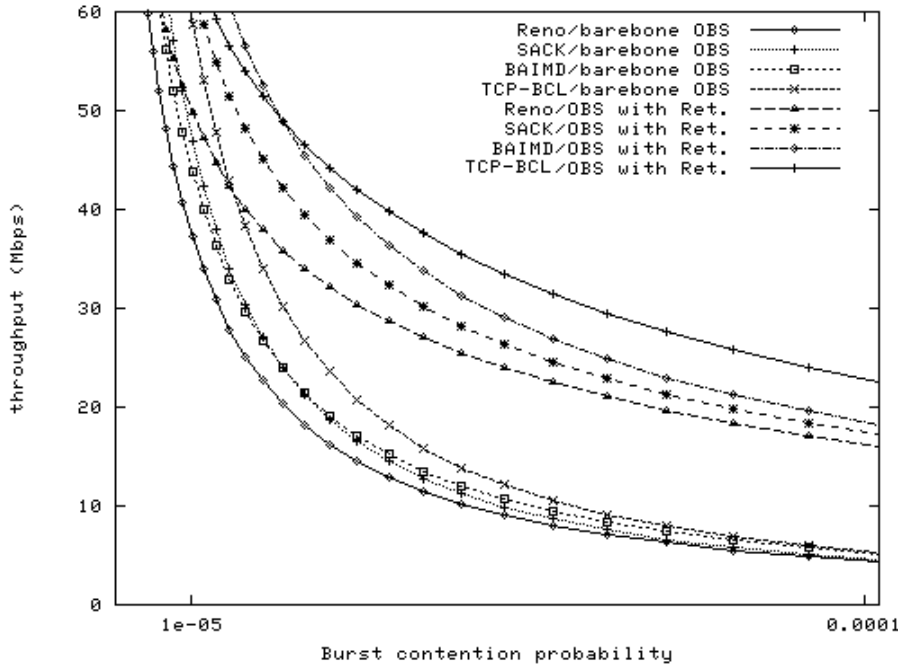

Figure 5.4 TCP Reno, SACK vs. TCP-BCL throughput over barebone OBS and OBS with burst retransmission

In Figure 5.5, TCP-BCL is compared with BAIMD and BTCP. TCP-BCL still achieves higher throughput than the other two by combining the best features of BTCP and BAIMD. The explicit notifications inform the TCP-BCL senders of a specific OBS-domain channel status, so the two control parameters can be adjusted appropriately in response to the burst-dropping event

Through the simulation studies, we verified that TCP-BCL can solve the false-congestion-detection problem well and achieve better throughput than all the other TCP implementations based on AIMD. The overhead incurred is the explicit notification when a burst loss is determined to be due to random contention. With the aid of explicit notifications and implementation of the TCP agent at the OBS edges, TCP-BCL can outperform BAIMD and BTCP (BACK/BNACK) solidly. In addition, it can effectively maintain fairness among competing AIMD flows and awareness of non-congestion burst dropping by manipulating the α and β values in the TCP senders. This suggests TCP-BCL can serve as a candidate TCP implementation for the future Internet with OBS infrastructure.

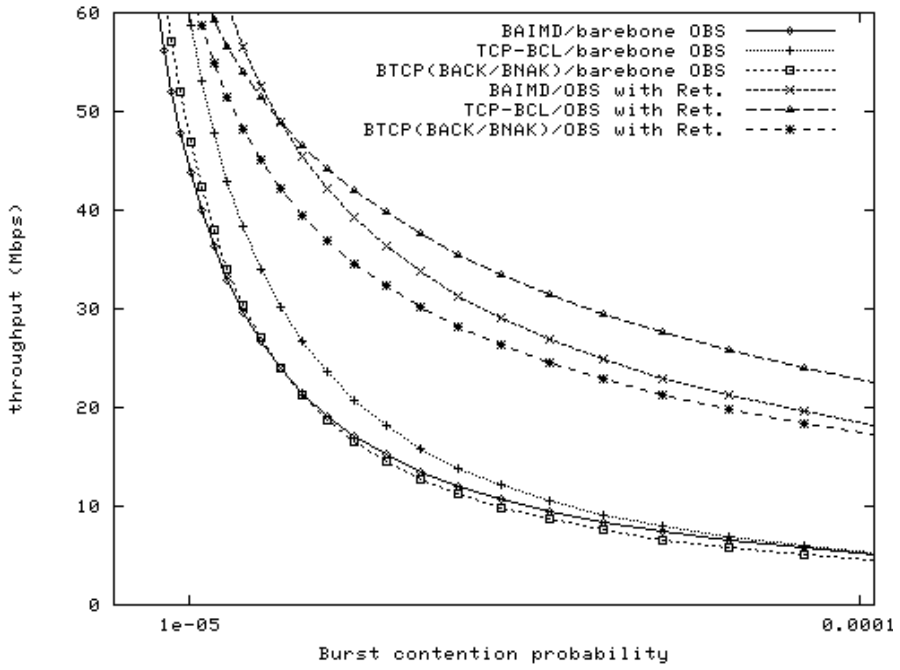

Figure 5.5 BTCP, BAIMD, and TCP-BCL throughput over barebone OBS and OBS with burst retransmission.

5.4.2 TCP-BCL Fairness

In this section, fairness among TCP-BCL, BAIMD, SACK, and Reno is examined. Figure 5.6 shows the fairness index of flows by TCP-BCL, BAIMD, and TCP Reno. We can see TCP-BCL has a much better fairness index than BAIMD. This is due to the fact that the congestion control mechanism of BAIMD does not rely on explicit signalling between the TCP senders and the OBS edge nodes, which causes an underestimation of network congestion. Thus, it results in selecting larger values of β while keeping α at 1.

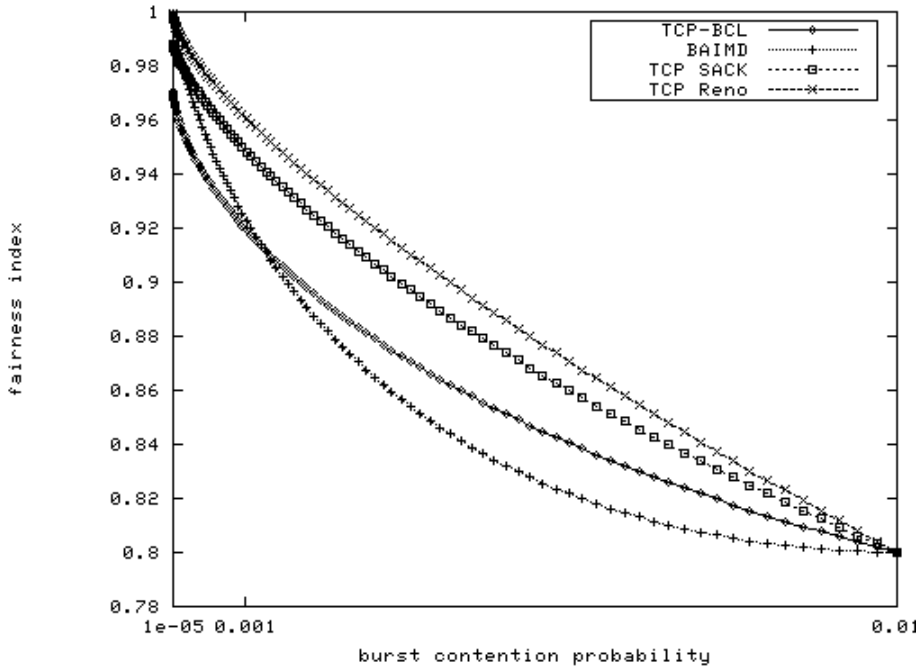

Figure 5.6 Fairness index of TCP-BCL, BAIMD, SACK, and Reno in barebone OBS.

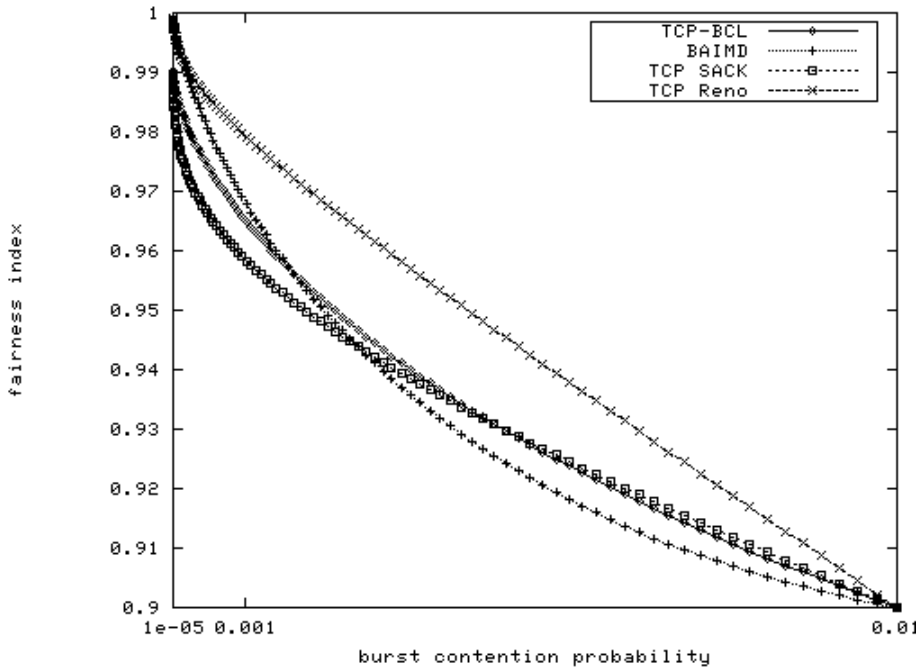

Figure 5.7 Fairness index of TCP-BCL, BAIMD, SACK, and Reno in OBS with burst retransmission

The fairness index is also examined in Figure 5.7 while enabling burst retransmission. The simulation results show that the throughput of TCP-BCL is close to SACK and BAIMD, which verifies that TCP-BCL maintains a friendly relation with the other TCPs.

5.5 Conclusion

In this chapter, we proposed a new TCP implementation in IP over OBS networks, called TCP with Explicit Burst-Contention Loss Notification (TCP-BCL), attempting to combine the best enhancements of BAIMD and BTCP with (BACK/NACK). Through simulation and analytical modeling, we demonstrated that TCP-BCL can counter the negative effects of the lack of buffers in OBS, and is a better candidate in OBS networks than the conventional AIMD architecture. We have shown the superiority of the proposed scheme in terms of throughput, link utilization, and fairness.

After analyzing the throughput performance of dropping-based TCP over OBS networks with explicit and non-explicit notification, we extend our study to cover the second largest TCP category, which is the delay-based. In the following chapter, we analyze the effects of the OBS transmission characteristics on delay-based TCPs. For this purpose, we have chosen TCP Vegas since it is the most common implementation of delay-based TCP.

Chapter 6
Delay-Based TCP (Vegas) over OBS

In this chapter we investigate the behaviour of delay-based TCP Vegas over barebone OBS and OBS with burst retransmission. We evaluate the impacts of extra delay introduced by burst retransmission on TCP Vegas. A novel scheme based on TCP Vegas is proposed, which adopts a threshold-based mechanism for determining network congestion in OBS networks with burst retransmission. Analytical models for the throughput of TCP Vegas and threshold-based TCP Vegas are formulated and are further validated through simulation [76][77][78]. To verify the effectiveness of the proposed threshold-based TCP Vegas, simulation is conducted to compare the proposed scheme with conventional TCP implementations. We observe a significant improvement in terms of TCP throughput for threshold-based TCP implementation over OBS networks with burst retransmission.

6.1 Threshold-based TCP Vegas over OBS

Delay-based TCP implementations, such as TCP Vegas [10] and Fast TCP [36][39], measure the delay of each packet transmission to estimate available bandwidth and congestion status in networks. The performance of Fast TCP has been evaluated in [36][39] and can be considered a high-speed TCP Vegas. Recalling Section 2.1, TCP Vegas modifies Reno in the congestion avoidance, slow start, and retransmission phases.

There are several issues when conventional TCP Vegas congestion control is adopted in OBS networks. Given the physical route between the source and destination OBS edges, the delay experienced in the OBS domain is primarily the sum of burst assembly delays and link propagation delays, which do not vary with the network traffic load. Hence, conventional TCP Vegas cannot effectively detect network congestion in the OBS domain. Furthermore, in the event that all the packets in a single congestion window are assembled into a single burst, the TCP Vegas sender may suffer from the false-congestion-detection problem if the burst is lost in the OBS domain due to random contention, which will fatally impair TCP throughput.

When TCP Vegas runs over OBS networks with burst retransmission, the false-congestion-detection problem can be mitigated significantly. However, although burst retransmission hides some burst-loss events from the upper TCP, it incurs extra delay. This additional delay enlarges the RTTs perceived by the TCP sender for the packets in the retransmitted bursts. Hence, we are motivated to further enhance conventional TCP Vegas, so that TCP senders can tell whether the increase in RTTs is due to network congestion, or due to retransmission in lightly-loaded OBS networks.

We observe that as the IP access network becomes congested, TCP Vegas will detect the increase in RTTs due to queuing delay and packet loss. If the OBS network becomes heavily loaded, burst contention events happen frequently. Thus, TCP Vegas would often detect RTT increases. If neither the IP access network nor the OBS network are congested, random burst contention losses occur less frequently. Hence, TCP Vegas senders can only detect network congestion (in both OBS and IP access networks) if RTTs increase frequently.

Based on these observations, we propose a threshold-based TCP Vegas scheme to distinguish between network congestion and random burst contention loss at low traffic loads. A new parameter, *T*, is the threshold on the number of packets with longer RTT than the minimum RTT measured in the previous *N* rounds. The parameter is used to evaluate the extra delay caused by burst retransmission, and can be manipulated adaptively in order to mitigate the impact of false congestion identification [76][78].

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

TCP packets whose RTTs are larger than their $MinRTT(i-1)$ exceeds the threshold T, it means that the network is congested. Hence threshold-based Vegas recognizes the network congestion and calculates the *actual* throughput as usual.

The basic idea behind threshold-based TCP Vegas is to reduce the sensitivity of TCP Vegas to the increases in RTTs caused by burst retransmission in the OBS domain. Instead of changing *cwnd*, the sensitivity of TCP Vegas can also be reduced by decreasing α or increasing β. However, changing α and β makes it difficult for TCP Vegas as to estimate the available bandwidth in the networks.

In threshold-based TCP Vegas, both N and T should be chosen much larger than the number of packets from a single TCP Vegas connection that are assembled into a burst, so that TCP Vegas is able to detect the frequency of retransmission in the OBS domain based on the record of a number of bursts. In general, packets from a TCP connection assembled in the same burst have the same measured RTT. By analyzing the variation pattern of historical RTTs, TCP Vegas can obtain the number of packets from a TCP Vegas connection that are assembled into a burst. T and N can also affect TCP performance. N should be much larger than T so that exceeding the threshold T implies a high probability of actual network congestion. Hence, we take $N = kT$ where $k > 1$. The TCP throughput performance is evaluated according to different values of T and N the following section. Figure 6.1 summarizes the proposed threshold-based TCP Vegas.

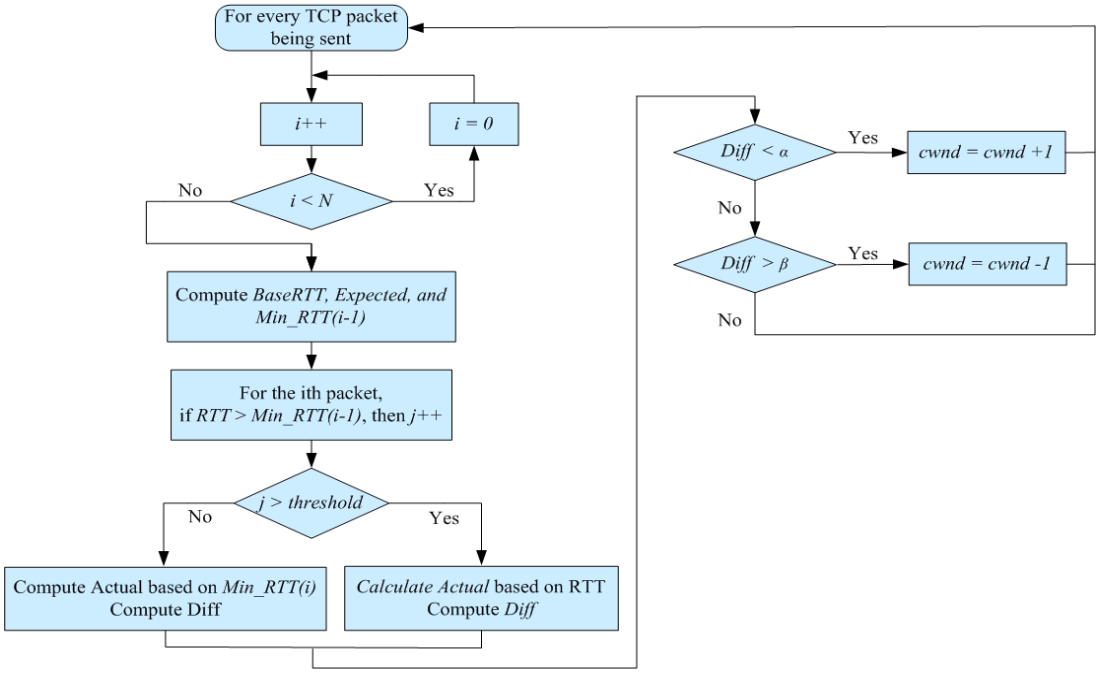

Figure 6.1 Threshold-based TCP Vegas congestion-control.

6.2 TCP Vegas Performance Modeling over OBS

In this section, we analyze the throughput of the proposed threshold-based TCP Vegas over OBS networks with burst retransmission. We assume that the access bandwidth of a TCP flow is high enough that all TCP packets in a single congestion window are assembled in a burst. Such a TCP flow is referred to as a fast TCP flow. Fast TCP flows will not trigger triple duplicates since packets in an entire congestion window are lost due to a burst loss. We assume that the burst assembly delay is fixed, and the occurrences of burst losses and contentions are independent based on a given and fixed burst dropping probability and burst contention probability. The Burst dropping probability is the probability that a burst is dropped, and is usually much lower than the burst contention probability, defined as the probability that a burst experiences contention [104][105]. This assumption is justified as some of the contended bursts can reach their destinations successfully after being retransmitted. Thus, the burst dropping probability is always lower than the burst contention probability.

We first analyze the throughput of TCP Vegas over barebone OBS networks based on the analytical model proposed in [65], followed by the analysis of TCP Vegas over OBS networks with burst retransmission. Last, we analyze the throughput of the proposed threshold-based TCP Vegas over OBS networks with burst retransmission by integrating the two previous models. The notations used are presented in Section 3.1.

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

$$B = \frac{E[Y]+E[H]}{E[A]+E[TOP]} \tag{6.1}$$

6.2.1 TCP Vegas over Barebone OBS

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

Figure 6.2 shows the evolution of *cwnd* for TCP Vegas over a barebone OBS networks. The evolution of *cwnd* can be partitioned into the following periods: (1) the slow start period, the duration from A to B in Figure 6.2, in which *cwnd* starts at 2 and doubles every other round until it reaches the expected slow-start threshold. (2) The transition period from B to C. Since *RTT* is close to *BaseRTT*, *Diff* calculated based on Equation 2.3 will be a very small value less than α. Hence, in the transition period, *cwnd* increases by 1 every RTT, until TCP Vegas reaches the stable state, $\alpha \le Diff \le \beta$. (3) The loss-free period that includes a series of consecutive successful rounds from C to D, during which Vegas is in the stable state (*cwnd* remains unchanged) and no burst loss occurs. (4) The TO period, which may include consecutive timeout events. The duration from slow start to the first encountered TO period (or the beginning of slow start of the next sequence of rounds) is referred to as the slow-start-

to-slow-start (SS2SS) period [65]. We assume that the burst dropping probability is low enough that there is no burst loss during the slow start period or the transition period.

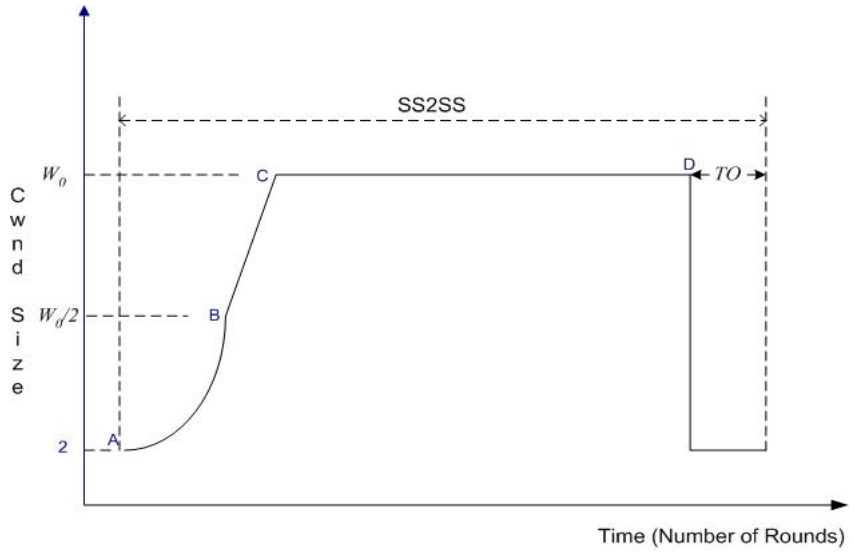

Figure 6.2 Evolution of *cwnd* for TCP Vegas over a barebone OBS

In Figure 6.2, W_0 is the congestion-window size when Vegas is in the stable state, *i.e.* $\alpha \le Diff \le \beta$. Since TCP Vegas adjusts the backlog packets in the network between α and β, the expected value of *Diff* is estimated as $\frac{\alpha+\beta}{2}$ based on [65]. By applying Equation 2.3, we have

$$Diff = W_0 (1 - \frac{BaseRTT}{R}) = \frac{\alpha + \beta}{2}, \tag{6.2}$$

By solving for W_0, we have,

$$W_0 = \min\left(\frac{\alpha+\beta}{2} \times \frac{R}{R - BaseRTT}, W_{\max}\right). \tag{6.3}$$

We now calculate the TCP Vegas throughput by computing the expected number of packets transmitted in the period from A to D. In the slow start period from A to B, *cwnd* doubles every other round until it reaches the slow start threshold, $W_0/2$ since the expected

cwnd when the TO occurs is W_0 [65]. Thus, based on Equation 17 in [65], the number of packets sent in the period from A to B, Y_{AB}, is

$$Y_{AB} = 2\sum_{i=0}^{\log\frac{W_0}{4}} 2^i = 2^{\log W_0} - 4, \tag{6.4}$$

and the duration of the slow start phase is

$$A_{AB} = 2(\frac{\log W_0}{4})R = 2(\log W_0 - 2)R, \tag{6.5}$$

where R is the expected RTT.

In the transition period from B to C, let the average number of packets transmitted be Y_{BC} and the duration of that period be A_{BC}. Based on Equation 18 in [65], Y_{BC} can be obtained by aggregating the number of packets sent when *cwnd* increases from $W_0/2$ to W_0-1 as follows:

$$Y_{BC} = \sum_{i=W_0/2}^{W_0-1} i = \frac{3W_0^2}{8} - \frac{W_0}{4}. \tag{6.6}$$

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

$$A_{BC} = (\frac{W_0}{2})R. \tag{6.7}$$

We then calculate the average number of packets transmitted in the loss-free period from C to D (denoted by Y_{CD}) and the duration of that period (denoted as A_{CD}). We assume a Bernoulli burst-loss model, where the random variable X is geometrically distributed with $\Pr(X = k) = (1-p)^k p$, where $k = 0,1,2,3\ldots$.

Hence, the number of rounds during the loss-free period, $S_{lossfree}$, is

$$S_{lossfree} = \sum_{k=0}^{\infty} k(1-p)^k pk = (1-p)/p. \tag{6.8}$$

Then the duration of the loss-free period is

$$A_{CD} = RS_{lossfree} \tag{6.9}$$

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

$$Y_{CD} = W_0 S_{lossfree} . \tag{6.10}$$

In the TO period, since the timeout process of TCP Vegas is similar to TCP Reno, based on Equations 14 and 16 in [19], we have the mean duration for successive TOs as

$$E[TOP] = RTO \frac{f(p)}{1-p}, \tag{6.11}$$

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

$$E[H] = \frac{p}{1-p} . \tag{6.12}$$

We then have the average number of packets sent during the consecutive 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 6.3 - 6.12 into Equation 6.1, we obtain TCP Vegas throughput over a barebone OBS networks as follows:

$$B_{barebone} = \frac{Y_{AB} + Y_{BC} + Y_{CD} + E[H]}{A_{AB} + A_{BC} + A_{CD} + TOP} . \tag{6.13}$$

6.2.2 TCP Vegas over OBS with Burst Retransmission

We now analyze TCP Vegas throughput over an OBS network with burst retransmission. Recall that burst retransmission can greatly improve burst loss probability, but it causes longer RTTs for the retransmitted bursts. Figure 6.3 shows the evolution of *cwnd* for TCP Vegas over an OBS network with burst retransmission.

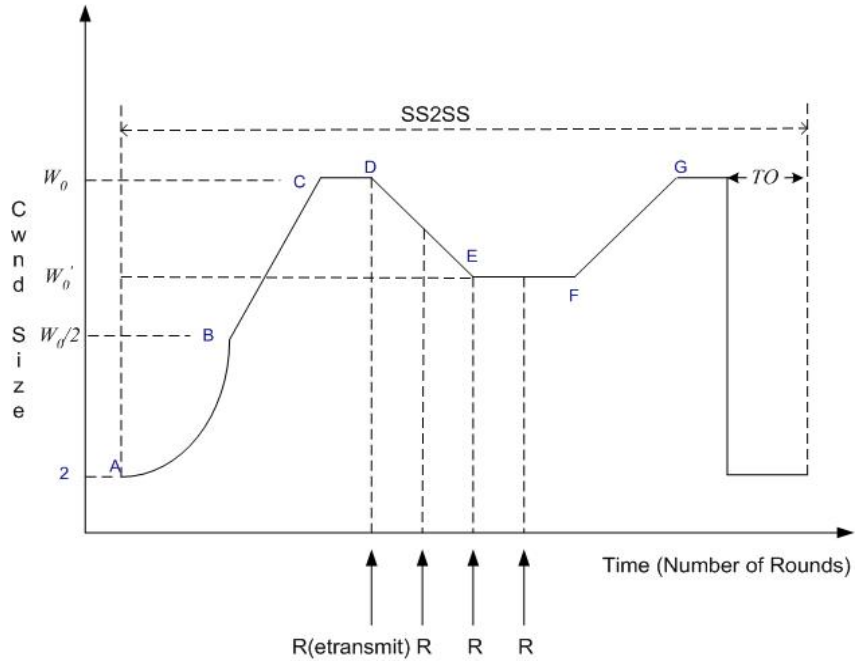

Figure 6.3 Evolution of TCP Vegas *cwnd* over an OBS networks with burst retransmission

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

$$p_{sr} = \frac{p_c - p}{1 - p}. \tag{6.14}$$

The SS2SS period is partitioned into 4 periods: (1) the slow start period from A to B, (2) the transition period from B to C, (3) the loss-free period from C to G, where rounds may

suffer longer RTTs due to burst retransmission, (4) the TO period. In Figure 6.3, the slow-start period, the transition period, and the TO period are similar to those in Figure 6.2. In this section, we focus on the analysis of the loss-free period from C to G.

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

$$p_{nc} = \frac{1 - p_c}{1 - p} . \tag{6.15}$$

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

$$Diff_C = W_0 (1 - \frac{BaseRTT}{R}) . \tag{6.16}$$

During the period from C to G, consider a round with the equal to W_0 (see the point D in Figure 6.3). If the round is retransmitted successfully in the OBS domain, all the packets sent in this round will experience a longer delay, RTT_r. Hence, the *Diff* calculated at the point D is

$$Diff_D = W_0 (1 - \frac{BaseRTT}{RTT_r}) . \tag{6.17}$$

If $\alpha \le Diff_D \le \beta$, TCP Vegas will leave *cwnd* unchanged from C to G. Hence, by inverting Equation 6.17, we can obtain the range of RTT_r for which *cwnd* remains constant, that is,

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

If $Diff_D > \beta$, then *cwnd* decreases by 1. If multiple consecutive rounds are retransmitted in the OBS network, *Diff* will keep decreasing for each of the consecutive rounds due to the decrease in *cwnd*, until *Diff* reaches β, (the duration from D to E in Figure 6.3). After *Diff* reaches β, *cwnd* remains unchanged for future consecutive rounds retransmitted in the OBS domain (see the duration from E to F in Figure 6.3). Let W_0' be the lower bound of *cwnd* from C to G. We have *Diff* at the point E as,

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

Then we have

$$W_0' = \left\lceil \frac{\beta RTT_r}{RTTr - BaseRTT} \right\rceil . \tag{6.18}$$

Note that for a round with a longer delay RTT_r when $Diff_D > \alpha$, *cwnd* will remain unchanged, because if $W_0 = W_{\max}$, *cwnd* has already reached the maximum window size. If $W_0 < W_{\max}$, then $Diff_C \geq \alpha$ and $R < RTT_r$. Thus, we have $Diff_D \geq Diff_C \geq \alpha$ from Equations 6.16 and 6.17.

For any non-retransmitted round during the loss-free period, if $cwnd < W_0$, *cwnd* increases by 1 and $Diff = cwnd\ (1 - \frac{BaseRTT}{R}) < \alpha$. For example, during the period from F to G in Figure 6.3, *cwnd* increases by 1 for each non-retransmitted round until it reaches W_0.

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

Since the probability that a round is successfully retransmitted is p_{sr} and the probability that a round does not experience contention is p_{nc}, we model the state of *cwnd* as a Markov Chain shown in Figure 6.4 [31]. Let π_i be the probability that $cwnd = W_0 - i$. We solve for π_i by local balance as follows.

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

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

...

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

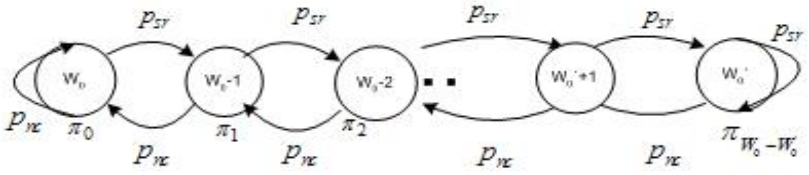

Figure 6.4 Markov chain for the state of *cwnd*

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

$$\pi_0 = \frac{1}{\sum_{i=0}^{W_0-W_0'} (\frac{p_{sr}}{p_{nc}})^i} = \begin{cases} \frac{1}{W_0 - W_0' + 1}, & if \quad p_{sr} = p_{nc}, \\ \frac{1-(\frac{p_{sr}}{p_{nc}})^{W_0-W_0'+1}}{1-\frac{p_{sr}}{p_{nc}}}, & if \quad p_{sr} \neq p_{nc}. \end{cases} \tag{6.19}$$

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

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

For the case when $BaseRTT \Big/ (1 - \frac{\alpha}{W_0}) \le RTT_r \le BaseRTT \Big/ (1 - \frac{\beta}{W_0})$, *cwnd* remains unchanged from C to G and then $E[W_{CG}] = W_0$.

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

$$Y_{CG} = E[W_{CG}]S_{lossfree}. \tag{6.20}$$

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

$$A_{CG} = (p_{nc}R + p_{sr}RTT_r)S_{lossfree}. \tag{6.21}$$

From Equations 6.4, 6.6, and 6.20, we obtain $E[Y]$ as

$$E[Y] = Y_{AB} + Y_{BC} + Y_{CG}\,, \tag{6.22}$$

and from Equations 6.5, 6.7, and 6.21, we obtain $E[A]$ as

$$E[A] = A_{AB} + A_{BC} + A_{CG}\,. \tag{6.23}$$

We then obtain TCP Vegas throughput over OBS networks with burst retransmission by substituting Equations 6.11, 6.12, 6.21, and 6.23 in Equation 6.1:

$$B_{ret} = \frac{Y_{AB} + Y_{BC} + Y_{CG} + E[H]}{A_{AB} + A_{BC} + A_{CG} + TOP}. \tag{6.24}$$

6.2.3 Threshold-based TCP Vegas over OBS with Burst Retransmission

Based on the two analytical models, we can analyze the throughput of threshold-based Vegas flows over OBS with burst retransmission. Figure 6.5 illustrates the evolution of *cwnd* during the time when N consecutive packets are sent using threshold-based TCP Vegas. While the number of TCP packets whose *RTTs* are larger than the current *MinRTT* is under the threshold T, TCP Vegas stays with *MinRTT* for calculating *Diff*. Hence, the *cwnd* evolution of threshold-based TCP is similar to that of TCP Vegas over a barebone OBS, where $R = MinRTT$. Thus, the throughput before reaching T, denoted as $B'_{barebone}$, can be calculated based on Equation 6.13, where

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

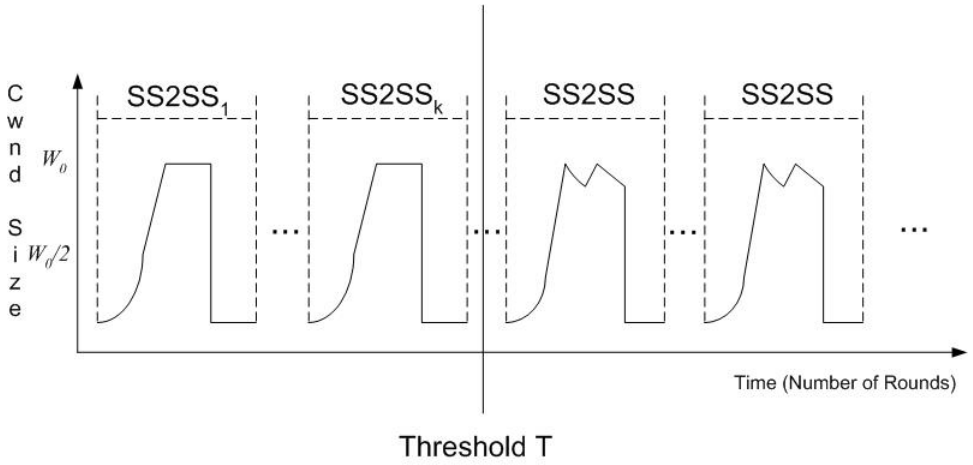

Figure 6.5 Evolution of threshold-based Vegas *cwnd* over OBS network with burst retransmission

After the number of TCP packets whose *RTTs* are larger than the current *MinRTT* reaches *T* (after $SS2SS_k$ in Figure 6.5), *cwnd* evolution is similar to that of TCP Vegas over OBS networks with burst retransmission. Hence, the throughput after reaching *T*, B'_{ret}, can be calculated based on Equation 6.24.

Let S_1 be the expected number of *SS2SS* periods before reaching *T* and let S_2 be the expected number of *SS2SS* periods after reaching *T*. We know that the expected number of consecutive successful rounds in an *SS2SS* period is $E[X] = (1-p)/p$. Hence, the total number of rounds before reaching *T* is $(S_1 E[X])$, and then the number of rounds that are retransmitted and experience extra delay is $(S_1 E[X] p_{sr})$. On the other hand, when the threshold *T* is reached, the number of rounds that are retransmitted is approximately T/W_0 since the expected *cwnd* when the TO occurs is W_0. We have $S_1 E[X] p_{sr} = T/W_0$, and then

$$S_1 = \left\lceil \frac{T}{W_0 E[X] p_{sr}} \right\rceil. \tag{6.25}$$

In each *SS2SS* period before reaching *T*, the number of packets sent is $(Y_{AB} + Y_{BC} + Y_{CD})$ from Figure 6.2. Then, the total number of packets sent before reaching the threshold is $S_1(Y_{AB} + Y_{BC} + Y_{CD})$. Since the total number of packets measured is less than *N*, the total number of packets sent after reaching the threshold *T* is $[N - S_1(Y_{AB} + Y_{BC} + Y_{CD})]$, where

$N > S_1(Y_{AB} + Y_{BC} + Y_{CD})$. Since the number of packets sent is $(Y_{AB} + Y_{BC} + Y_{CG})$ in each *SS2SS* period after reaching *T*, we can obtain the number of *SS2SS* periods sent after reaching *T* as

$$S_2 = \begin{cases} \left\lceil \dfrac{N - S_1(Y_{AB} + Y_{BC} + Y_{CD})}{Y_{AB} + Y_{BC} + Y_{CG}} \right\rceil, N > S_1(Y_{AB} + Y_{BC} + Y_{CD}) \\ 0, \quad otherwise \end{cases} \tag{6.26}$$

Hence, we obtain the throughput during the time when N consecutive packets are sent as

$$B_{threshold-based} = \frac{S_1}{S_1 + S_2} B'_{barebone} + \frac{S_2}{S_1 + S_2} B'_{ret} \tag{6.27}$$

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

6.3 Numerical Results

In this section we present the numerical results obtained from both the analytical models. In order to obtain precise protocol performance results, we limit the network complexity as well as the parameter space to a multi-hop network as shown in Figure 6.6, where the distances between the nodes are in *km*. The edge nodes, E_1 and E_2, are connected to the network core nodes with a 10 *ms* link propagation delay. Each link consists of one bi-directional control channel for control signalling and a single fiber link for data-burst transfer. Each data link consists of 8 wavelengths operating at a transmission rate of 10 Gbps. The mixed time/length based burst assembly algorithm is adopted, where the burst timeout threshold is set to 500 *ms*, and the maximum burst length is set to 50 *KB*. The control-header processing time is set to be 1 μs. The core nodes implement the *LAUC-VF* channel scheduling algorithm [91]. In the burst retransmission mechanism, bursts subject to any contention are allowed to be retransmitted only once in order to have the best chance of meeting the timeout threshold. In order to simulate the random burst contention phenomena, different contention probabilities

are imposed at the burst scheduling phase. The random burst contention probability p_c ranges between 10^{-5} and 10^{-2}. The average *RTT* in the simulation is approximately 110 *ms*.

In the simulation, TCP senders and receivers are attached to the OBS edge nodes E_1 and E_2 respectively. In order to simulate fast TCP flows, a high access bandwidth is allocated to each flow. The packet delay in the access network is set to be constant so that burst retransmission delay is the only reason for longer-round trip times. The FTP application generates TCP segments with an average size of 1*KB*. In all experiments, the maximum window size of TCP is 10^4 segments. We select $\alpha = 600$ and $\beta = 800$ for both TCP Vegas and threshold-based Vegas to utilize the bandwidth capacity quickly and efficiently. The TCP throughput is obtained over a simulation period of 10^3 seconds. Results in the following sections are based on simulation unless stated otherwise.

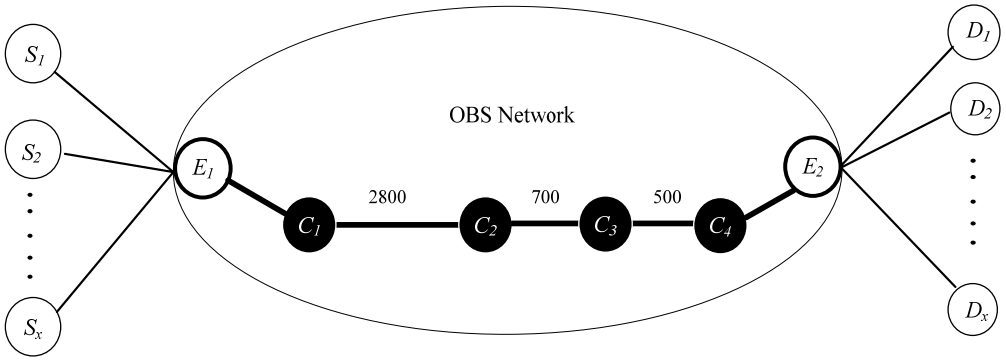

Figure 6.6 The network topology adopted in the simulation

6.3.1 Threshold-based TCP Vegas over OBS

In the following experimental studies, we compare the throughput of conventional TCP Vegas, threshold-based Vegas, and loss-based TCP SACK (*i.e.*, TCP SACK performs better than other existing loss-based TCP implementations in OBS networks [98]).

Figure 6.7 shows the throughput of TCP Vegas, threshold-based Vegas, and SACK over the barebone OBS network. Note that the burst contention probability is equal to the burst loss probability in the barebone OBS network. Threshold-based Vegas uses various values of the

threshold T from 100 to 400, while the number of consecutive TCP packets N is chosen to be $4T$. We see that threshold-based TCP Vegas with different N and T values and conventional TCP Vegas with $N = 0$ and $T = 0$ perform very similar, such that their throughput plots overlap. This is due to the fact that the round-trip time does not vary significantly in the barebone OBS network, and T and N can not enhance the threshold-based Vegas. We can also see that the TCP Vegas versions perform much better than loss-based TCP SACK in the barebone OBS network.

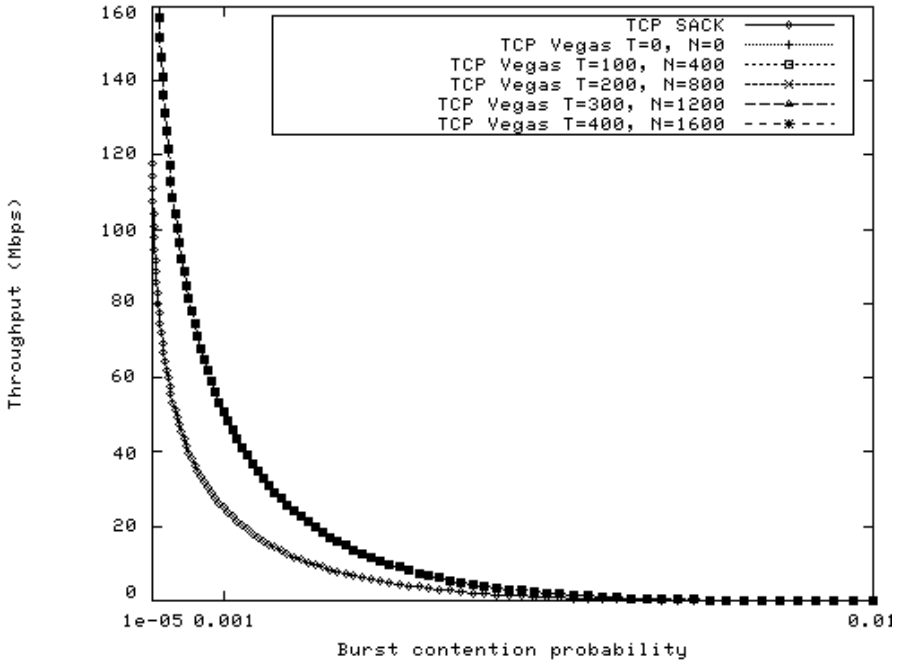

Figure 6.7 Throughput of Vegas, threshold-based Vegas, and SACK over a barebone OBS

Figure 6.8 compares the throughput of TCP Vegas, threshold-based Vegas, and TCP SACK over the OBS network with burst retransmission. Note that, with one attempt at retransmission, the burst loss probability in the simulation is much smaller than the burst contention probability. We observe that threshold-based Vegas performs much better than conventional Vegas and TCP SACK. For example, when the burst contention probability is 10^{-4}, TCP Vegas with $T = 300$ improves the throughput by 73% compared to conventional Vegas. We can also see that, with a higher threshold, threshold-based Vegas performs better.

When the burst contention probability is 10^{-4}, the threshold-based Vegas with T = 400 improves throughput by 82% compared to T = 200. This is because threshold-based Vegas with T = 400 more accurately detects the congestion state in the OBS network and delays triggering congestion avoidance.

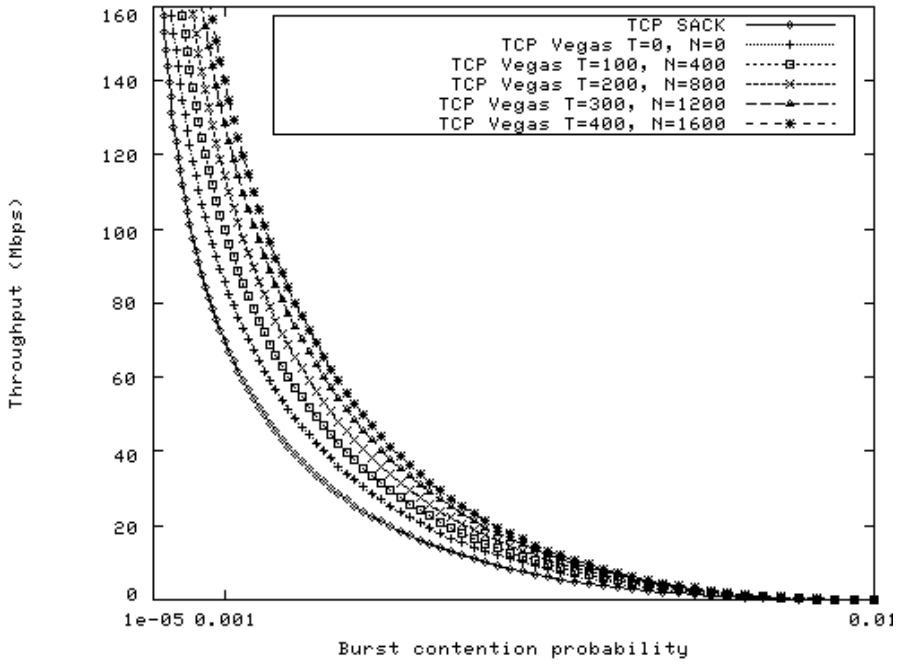

Figure 6.8 Throughput comparison of TCP Vegas, threshold-based Vegas (N=4T), and SACK over an OBS network with burst retransmission

Figure 6.9, we examine the effect of T and N on threshold-based Vegas. We observe that, with a fixed T, varying N does not result in a major throughput change. For example, with T = 200, the throughputs of N = 400 and N = 800 are very close. We can also see that the throughput is mainly affected by the value of T. For instance, the throughout of T = 200 increases 86% compared to the throughput of T = 100 when N = 400.

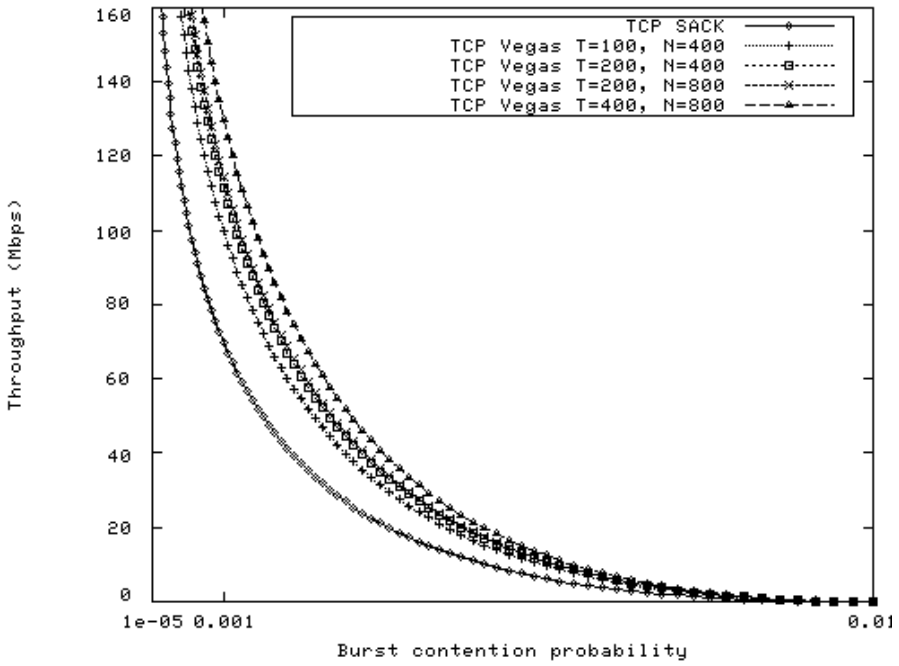

Figure 6.9 Throughput comparison of the threshold-based Vegas with fixed *N* or fixed *T* values

6.3.2 Threshold-based TCP Vegas Fairness Evaluation

In this section, we examine the fairness of TCP Vegas, Threshold-based Vegas, SACK and Reno. In our simulation, we generate three flows of each TCP version. The competing flows share the same source and destination. We then calculate the fairness index for each TCP version after obtaining the throughput of each flow.

Figure 6.10 compares the fairness index of TCP Vegas, threshold-based Vegas, SACK, and Reno. It is notable that the fairness index of threshold-based Vegas with larger *T* value results in lower fairness for the other co-existing flows. This result is due to the fact that the window-size reduction in threshold-based Vegas is substantially delayed when RTT increases. Thus, threshold-based Vegas is given the chance to occupy more link bandwidth at the expense of other TCP streams. The fairness index shows that the throughput of threshold-based Vegas is close to that of SACK and Reno. Also, the throughput is close among competing threshold-based Vegas flows while varying *T* and *N* values. Thus, we can see that threshold-based Vegas can maintain a friendly condition along with other TCP versions.

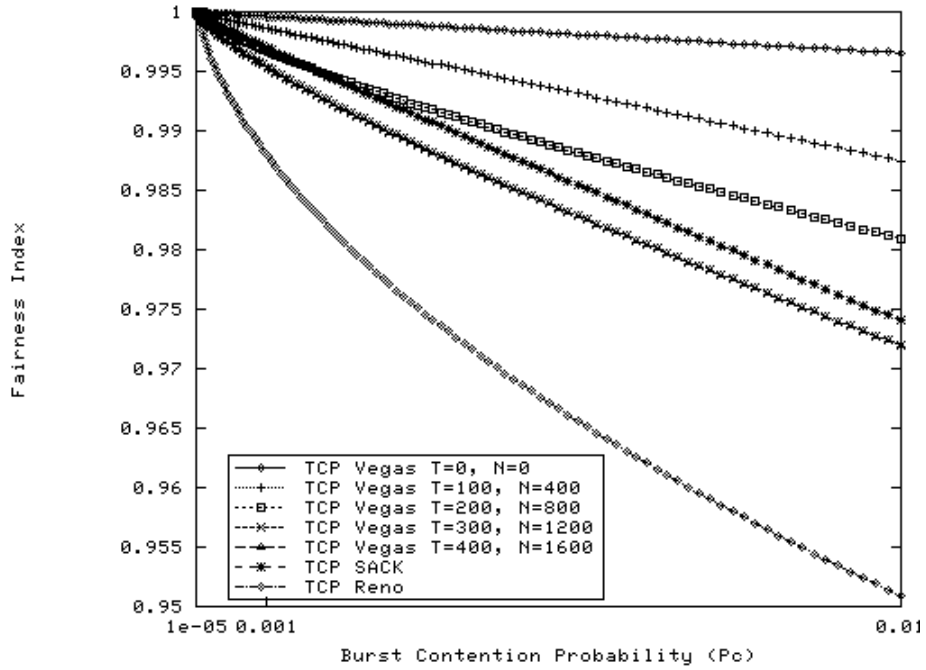

Figure 6.10 Fairness index of Vegas, threshold-based Vegas ($N = 4T$), SACK, and Reno

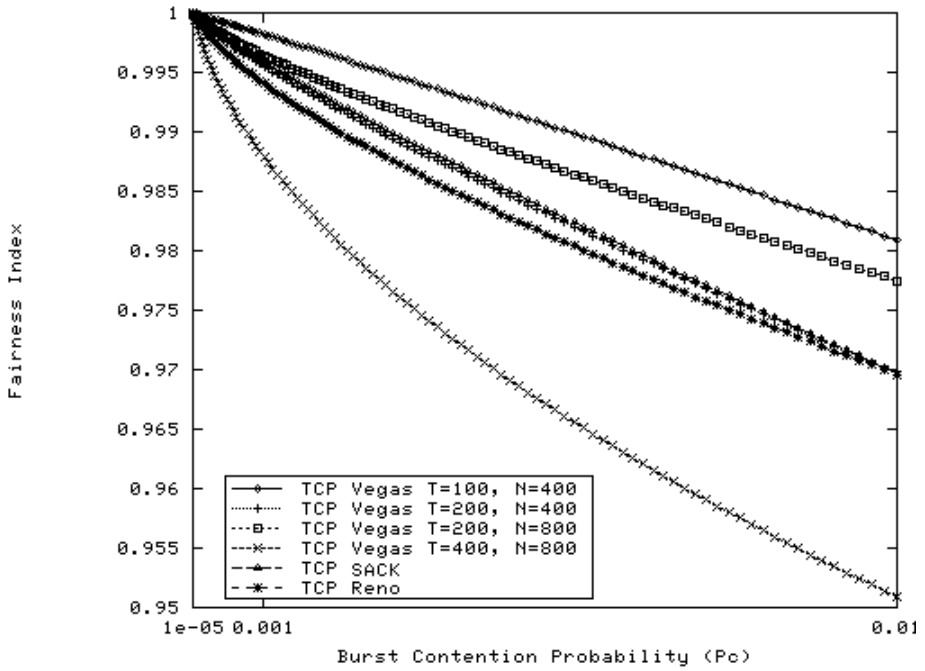

Figure 6.11 Fairness index of Vegas, threshold-based Vegas, SACK, and Reno with varying T and N values

In Figure 6.11we also examine the fairness of the threshold-based Vegas while fixing N and varying T. We also observe that varying T has a major impct on the threshold-based Vegas throughput and the associated fairness.

6.3.3 Analytical Results

In this section, we verify the analytical models proposed in Section 6.2. The analytical results yield the TCP throughput under different burst contention probabities p_c. Figure 6.12 compares the throughput of the conventional TCP Vegas obtained from Equation 6.13 to the throughput obtained from simulation with different burst contention probability p_c. The average RTT delay was 110 *ms*. We can see that the simulation results and the analytical results are very close.

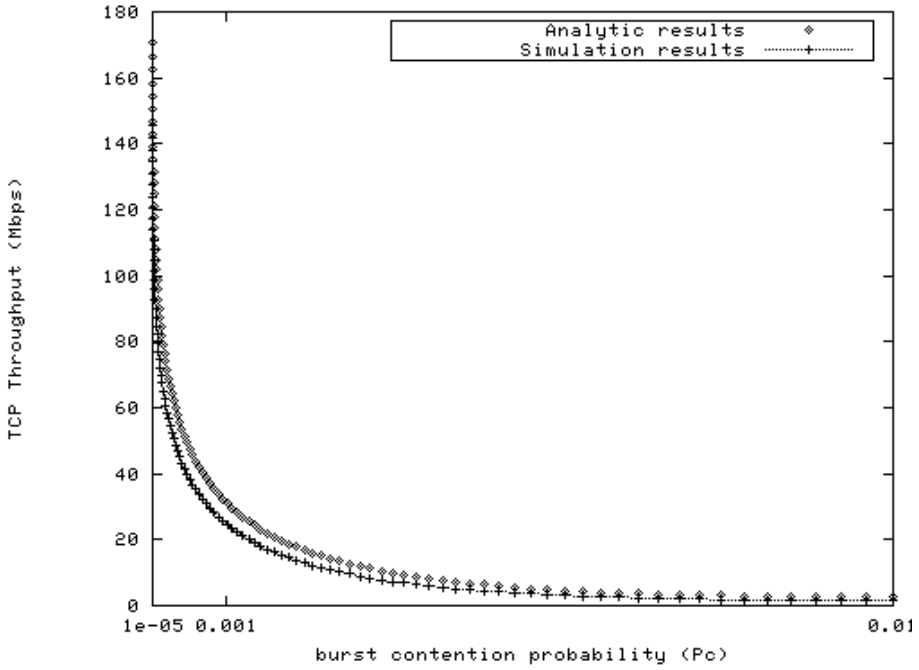

Figure 6.12 TCP Vegas throughput over a barebone OBS network

Figure 6.13 compares the analytical results obtained by Equation 6.24 and the simulation results. In the experiment, the OBS edge node enables burst retransmission. Bursts are retransmitted at most once. The average RTT delay in the simulation was 110 *ms*. Bursts that

are retransmitted experience an average of 100 *ms* extra delay. We see that the simulation results match the analytical results very well.

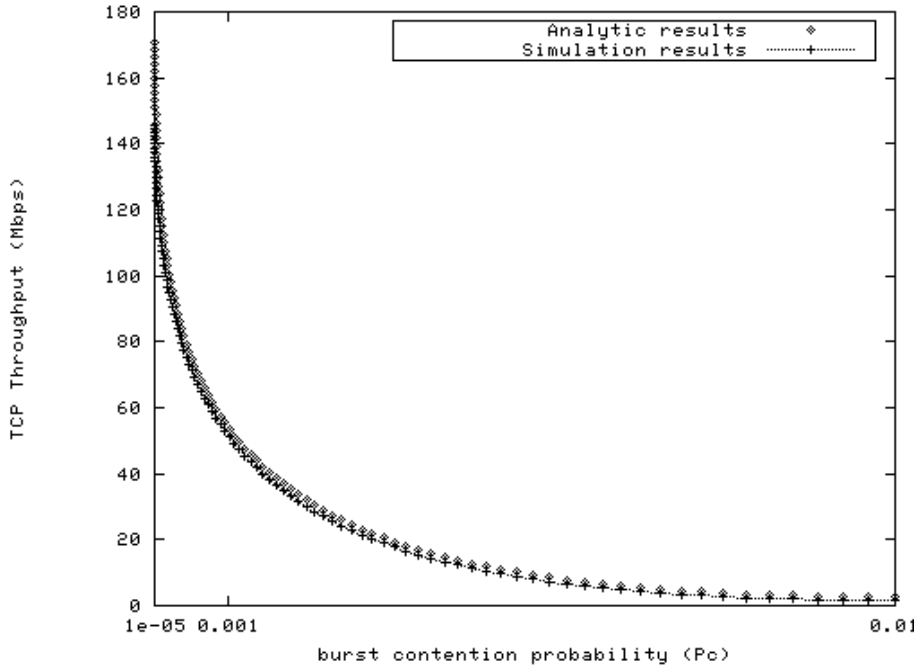

Figure 6.13 TCP Vegas throughput over an OBS network with burst retransmission

Figure 6.14 compares the analytical results obtained by Equation 6.27 and the simulation results obtained for threshold-based Vegas fast flows. In this experiment we compare T = 100, N = 400 and T=400, N=1600. We can see that the simulation results match the analytical results.

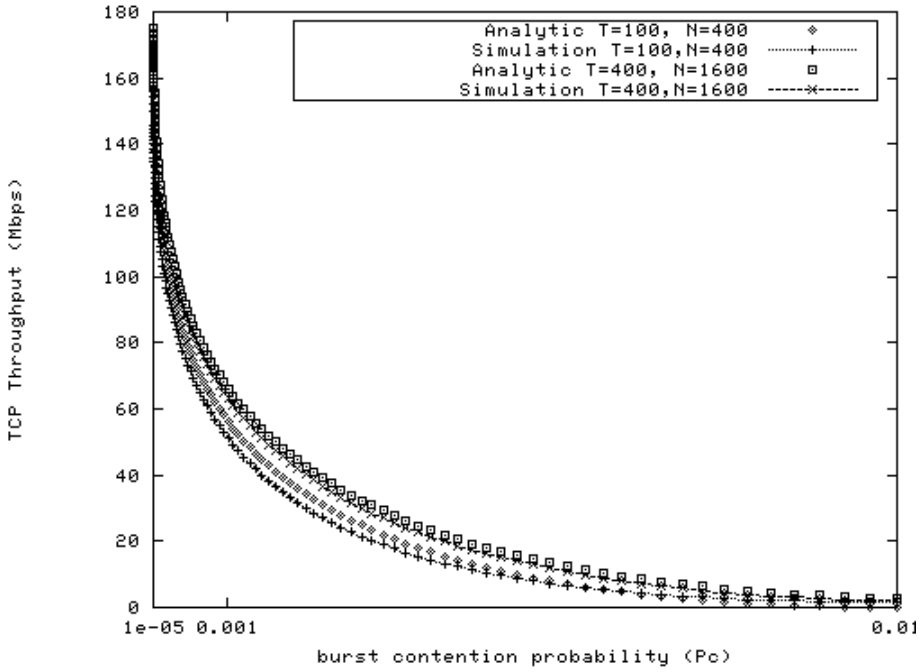

Figure 6.14 Threshold-based Vegas throughput over an OBS network with burst retransmission

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

6.3.4 Threshold-based TCP Vegas Steady State over OBS

In previous sections, we assume that the burst contention probability does not change when the TCP throughput varies. However, if all senders are transferring large files using TCP, the burst contention probability will be affected by the TCP throughput of the senders. At the same time, we see that TCP throughput is tightly coupled with the burst contention probability. Hence, there may exist a steady state at which the TCP throughput and the burst contention probability are stabilized.

In this section, we apply the analytical model verified in the previous section to analyze the steady-state throughput of threshold-based Vegas in an OBS network with burst

retransmission. We aim to obtain the proper N and T values which contribute to the expected steady-state threshold-based Vegas throughput.

The fixed-point method based on the TCP feedback mechanism is adopted from [13]. For a given burst contention probability, we can first calculate the TCP sending rate based on Equation 6.27, shown by a dashed line in Figure 6.15. Next, for a given TCP sending rate, we calculate the corresponding network input load to the OBS network. This load is then used to calculate the burst contention probability in the OBS network. The solid line in Figure 6.15 plots this relationship between the TCP sending rate and the burst contention probability. The intersection point of a dashed line and a solid line is a steady state.

In the latter step, given TCP throughput, the burst contention probability in an OBS network with burst retransmission is obtained as follows. Let the throughput of a TCP flow from source s to destination d be $S_{s,d}$ and the size of a TCP packet be pkt. The input load from a TCP flow is

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

Let p_c be the burst contention probability along the path from s to d. Since all the dropped bursts are able to retransmit only once, we compute the total load including the retransmitted traffic,

$$\Re_{s,d} = \rho_{s,d}(1 + p_c)$$

We can then compute the total load on link l_{ij} to be

$$\Re_{ij} = \sum_{(\forall s,d | l_{ij} \in route(s,d))} \Re_{s,d}$$

We assume that the burst arrivals on each link follow a Poisson process [41][103]. By using the Erlang-B formula, we can obtain the burst contention probability on link l_{ij} as

$$p_{ij} = ErlangB(\Re_{ij}, m)$$

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

$$p_c' = 1 - \prod_{(\forall s,d | l_{ij} \in route(s,d))} (1 - p_{ij}) \tag{6.28}$$

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

Figure 6.15 shows the analytical results of the steady-state threshold-based Vegas throughput over OBS with burst retransmission. From the figure, we can see that, when *T* is 200 and *N* varies from 800 to 1600, the steady state throughputs of threshold-based Vegas are identical. This is also true when *T* is 400 and *N* varies. These results validate the findings in Section 6.3.1, where we concluded that the threshold *T* has a dominant effect on the throughput.

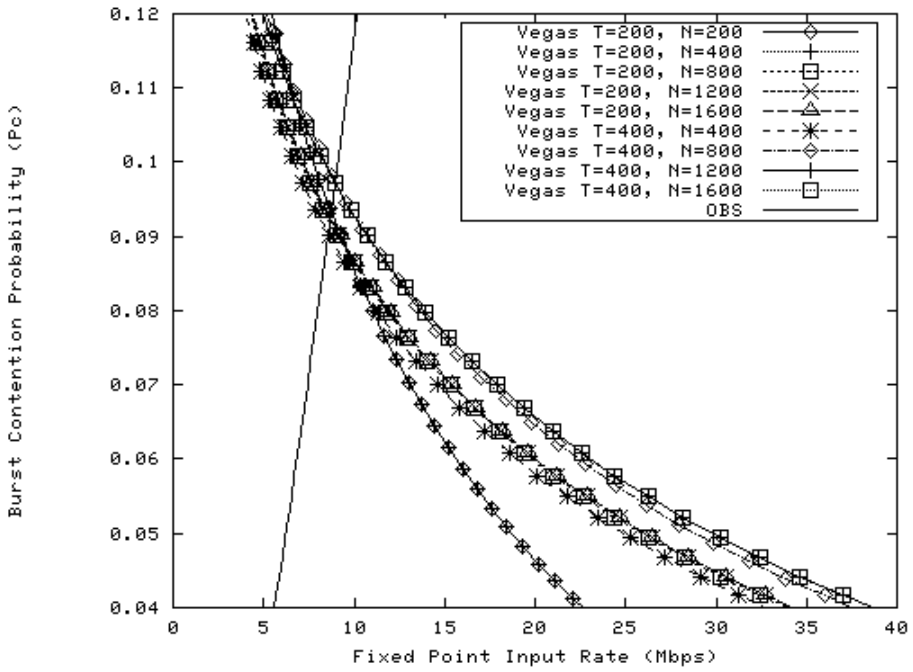

Figure 6.15 Analytical results of the steady-state input rate of threshold-based Vegas while varying *N* values

Figure 6.16 plots analytical results of the steady-state threshold-based Vegas throughput with fixed $N = 1600$. From this experiment, we observe that the threshold-based Vegas flows with $T \geq 400$ do not have a significant throughput increase. In the flows where $T > 400$ and $N > 800$, the steady state throughput remains close to the flows with $T = 400$ and $N = 800$. Therefore, we conclude that $T = 400$ and $N = 800$ are optimal for obtaining the best threshold-based throughput.

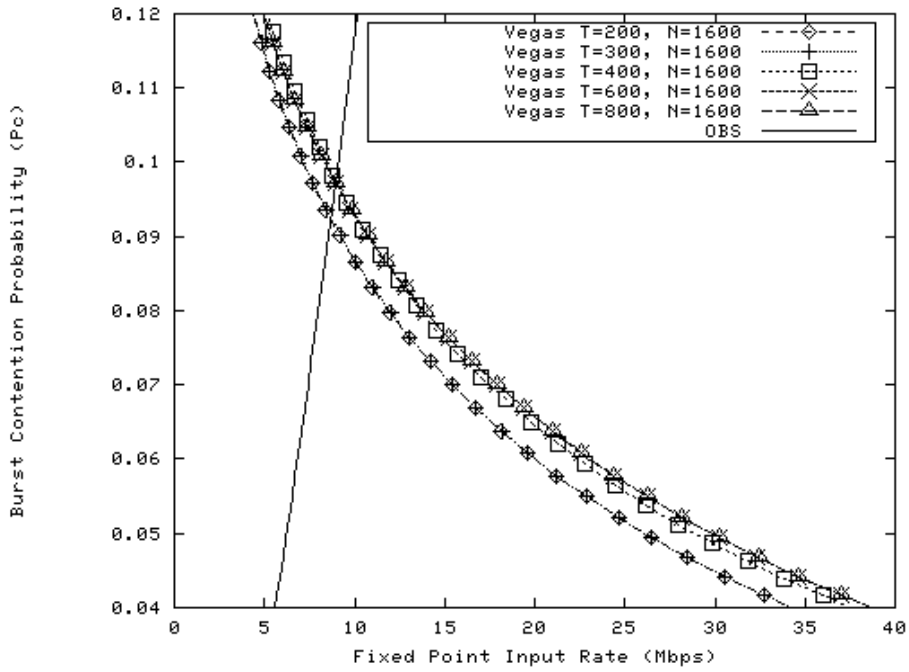

Figure 6.16 Analatical results of the steady-state input rate of Threshold-based Vegas while varying T values

6.4 Conclusion

In this chapter, we investigated the issues of delay-based TCP Vegas over OBS networks by proposing a threshold-based TCP Vegas mechanism that can effectively detect network congestion by manipulating a threshold parameter when TCP runs over OBS networks with burst retransmission. Our simulation results showed that the proposed scheme can outperform conventional TCP Vegas and TCP SACK significantly. Simulation results show

that the threshold T has a more significant impact on the TCP throughput than N. We also analyze the throughput performance of a fast threshold-based Vegas source over an OBS network with burst retransmission. Based on the analytical model, we obtain the steady state TCP throughput and observe the threshold value that results in an optimal throughput. The analytical model provides insights regarding the effect on the TCP throughput of burst contentions and losses in the OBS domain.

Chapter 7
Conclusions & Future Work

In this thesis, we investigated dropping-based, explicit-notification-based, and delay-based implementations of TCP over OBS networks. A suite of interoperable strategies that can facilitate TCP congestion-control mechanisms was developed to operate well with the intrinsic characteristics/behaviours of burst transmission in OBS networks. We studied the TCP false-congestion-detection problem extensively in bufferless OBS networks, which has been considered the most serious challenge in achieving efficient TCP performance over OBS networks. Furthermore, we studied the effect on the delay- and dropping-based TCPs of deploying burst-contention resolution schemes such as burst retransmission and burst deflection. The thesis has focused particularly on developing TCP performance modeling through novel analytical approaches. The thesis also provided a comprehensive framework and design paradigm for future research in TCP over OBS networks.

7.1 Contributions of the Thesis

The primary contribution of this thesis is the design and implementation of a suite of more robust TCP congestion-control mechanisms that handle the burst transmission behaviour and characteristics of OBS networks. More specifically, we made the following contributions.

1. We analyzed the impact on TCP throughput when adopting OBS transmission in both delay- and dropping-based TCP protocols. A detailed TCP performance evaluation was conducted under a number of transmission scenarios and network environments of interest. The negative impact of burst dropping on conventional TCP congestion-control was demonstrated. Furthermore, the performance of a number of well-known TCP implementations over OBS networks was examined, including TCP Reno, New Reno, SACK, DSACK, FACK, Eifel, BAIMD, and BTCP implementations.
2. We developed and evaluated a dropping-based scheme for TCP traffic regulation over OBS networks, called Statistical Additive Increase Multiplication Decrease

(SAIMD), where a policy-based approach for dynamically adjusting the multiplicative parameter β at the TCP senders was devised. Through simulations and analytical evaluation, we showed that SAIMD has better throughput than conventional TCP Reno and SACK under a wide range of network transmission conditions. Packet-loss probabilities and burst contention resolution strategies such as burst retransmission and burst deflection were examined.

3. We redefined the concept of network congestion in bufferless networks. Also, we proposed an efficient approach for detecting network congestion in OBS networks. In this context, we developed and analyzed a TCP congestion-control scheme with dynamic explicit Burst-Contention Loss notifications (TCP-BCL). The proposed scheme was shown to handle various OBS bursty phenomena that negatively affect TCP throughput performance and fairness. Under the proposed scheme, the performance impact on TCP of burst dropping due to random burst contention is considered and investigated. An analytical model is developed for the proposed scheme and is verified through extensive simulation.

4. The associated issues of delay-based TCP Vegas in OBS networks have been examined and analyzed. A novel threshold-based TCP Vegas is introduced, which is characterized by being able to identify network congestion status effectively through manipulating a threshold parameter, in the presence of burst retransmission and deflection in the OBS network. The simulation results showed that threshold-based TCP Vegas outperforms conventional TCP Vegas and TCP SACK significantly. We also evaluated different values of T and N thresholds through both simulation and analytical approaches. The corresponding analytical model is validated through extensive simulation, and provides insights on the effects of burst contentions and burst losses on TCP throughput.

Table 7.1 summarizes the achieved contributions categorized by solution category, supported OBS type, devices involved, and problems addressed.

Schemes	Solution Category		OBS Type		Devices Involved			Problems Addressed		
	Notification	Without Notification	Barebone	BCR	TCP Sender	OBS Edge	OBS Core	Random Burst	False TO	Burst Reordering
Burst AIMD		√	√	√	√			√	√	
TCP-BCL	√		√	√	√	√		√	√	√
TCP-ENG	√		√	√	√	√	√	√	√	√
Statistical AIMD		√		√	√			√	√	
Threshold-based Vegas		√		√	√			√	√	

Table 7.1 Thesis contributions of TCP over OBS

7.2 Future Research Directions and Open Problems

Although TCP has been subject to extensive research efforts in the past decades, TCP over OBS is a relatively unexplored research area, with a limited number of studies that have tackled only some of the unique features of these transmission characteristics. Through a close analysis of TCP enhancements listed in Chapter 2, we observed that the most important characteristics and abilities that a congestion-control scheme in TCP over OBS should have are: (1) be able to handle multiple TCP segment-loss events in one round trip, (2) be able to identify false TOs and burst losses in low-traffic situations, and (3) be able to compensate for out-of-order delivery in the OBS domain. In the following section, we identify open problems in the area of TCP over OBS networks.

7.2.1 Integration of Link-Layer Solutions with TCP

Although with less emphasis on TCP enhancements over OBS, there have been extensive studies of OBS networks that aim to reduce burst dropping, provide QoS in burst transmission, conduct a cross-layer design optimization in terms of burst assembly delay, burst size, and burst delivery, *etc*. The link-layer schemes, such as adaptive assembly algorithms [14], burst retransmission [104][105], burst deflection [37][68], and FDLs [18], have contributed successfully to reducing burst-loss probability at the expense of introducing extra delay and design complexity. It is important to integrate the link-layer solutions with the mechanisms for TCP throughput and reliability enhancement before OBS can be deployed practically in the Internet.

We found that reducing the burst-dropping probability may not in all cases result in significant enhancement to TCP throughput. For example, burst retransmission in OBS networks can improve TCP performance greatly. However, the persistence of retransmission, deflection, or segmentation are subject to further considerations since too much persistence leads to TO in the TCP layer, which significantly throttles the TCP transmission. One of the most important challenges in tackling this problem is to properly define the TO threshold at the TCP senders in the presence of various traffic loads at edge and core nodes. The determination of the threshold value should take into consideration the number of packets assembled in a single burst. A TCP snooping mechanism similar to schemes proposed for wireless networks such as ELN [10], JTCP [34], TCP Peach [1], ATCP [51], and TCP Casablanca [7] can be developed to evaluate the persistence of burst retransmission in estimating the RTT. This enhancement aims to reduce the risk of having a false TO while benefiting from burst retransmission or burst deflection.

The research on TCP over OBS presented in BTCP [98], BAIMD [73], TCP-BCL [72][75], and SAIMD [71][74] has successfully mitigated the effects of false congestion detection due to OBS networks. This work improves the TCP senders' reaction to each packet-loss event. However, each scheme suffers from some overhead as well, such as high computation complexity and/or extra signalling in the OBS network. Also, some of the above schemes may fail to overcome the problem of losing a large number of packets assembled in single

burst. Furthermore, it is necessary to evaluate the convergence rate of TCP in the presence of either constant RTT in barebone OBS or different values of RTT due to the deployment of burst contention resolution schemes.

7.2.2 TCP Convergence Rates for Large Bandwidth-Delay Product Networks

There exists a significant lack of performance evaluations on the TCP modifications proposed for network environments with large bandwidth-delay products over OBS networks. Fast TCP [36][39], Binary Increase Congestion control (BIC) TCP [93], Explicit Control Protocol (XCP) [23], TCP with Simple Available Bandwidth Utilization Library (SABUL) [79], High Speed (HSTCP) [24], and Scalable TCP (STCP) [44], are among the most famous promising solutions. XCP showed stability and efficiency using ECN through extending ECN and Core Stateless Fair Queue (CSFQ), which made XCP aware of the per-flow state and buffer-size status. XCP uses Multiplicative Increase Multiplicative Decrease (MIMD) to control the congestion window and AIMD to maintain fairness [23]. SABUL introduced a hybrid approach by merging rate-based transmission via the UDP and reliable retransmission via TCP, where the UDP channel is adopted for transmitting data at high rates, while the TCP channel is used to resend missing data segments to ensure reliability [79].

Depending on the current *cwnd*, HSTCP uses *a(cwnd)* and *b(cwnd)* for computing the next window size. This scheme is known to be a safe incremental approach [24]. A simulation-based study of HSTCP over OBS is presented in [108]. Using small burst-assembly delay and moderate burst dropping, the study shows that HSTCP throughput is affected in several ways.

Scalable TCP (STCP) is a sender-side TCP that offers a robust mechanism to improve performance in high-speed wide-area networks using traditional TCP receivers. STCP increases the *cwnd* by 0.01 for each acknowledged packet (not in the fast recovery stage), and cuts the *cwnd* by 0.875 for each packet-loss event [44].

There exist several proposals that solve the TCP slow-convergence problem over high-speed networks. There is a need to evaluate burst-assembly delay and burst dropping over

these TCP congestion-control algorithms. There are great opportunities for investigating the effect of the burst-assembly delay, burst dropping vs. packet aggregation gain on Fast TCP, Scalable TCP, XCP, and SABUL. It is known that the above schemes can achieve faster convergence of TCP throughput in large-bandwidth high-delay networks by enlarging their *cwnd* quickly. However, with large *cwnd*, the number of ACKs is decreased significantly. In the presence of random burst losses that contain a large number of ACK packets, there is dramatic damage to the TCP ACK-clocking which forces TCP to fall into false detection of network congestion. Thus, an unnecessarily large number of packets is retransmitted. Furthermore, ACK losses are expected to affect a large number of TCP senders since the burst can assemble many ACK packets due to their small size. To our knowledge, the effect of ACK packet losses over OBS networks has not been addressed in the literature.

7.2.3 Performance Modeling for TCP over OBS Networks

The previously reported TCP over OBS performance modeling technique follows the packet-oriented approach [19][71][72][74][75][77][99][105]. Recently, new modeling techniques have been proposed. In the early 1990's, the fluid modeling technique proposed in [57] added new dimensions for modeling a large number of TCP flows. However, the fluid modeling approach requires strict assumptions, such as (1) having a very large number of TCP flows, (2) with Poisson arrival of loss events, and (3) with strong correlation among losses in one RTT but independent of those in other RTTs. Regarding the first assumption, there is no evidence that the number of TCP flows is sufficiently large at the OBS edge node. In OBS, since both random burst drops and dropping due to persistent congestion may occur, the second assumption is subject to further investigation. The third assumption can be justified partially since, in the barebone OBS, the RTT is more or less fixed. This is because the third assumption can only hold for TCP flows which can emit their entire *cwnd* while being assembled in one burst (*e.g.*, fast flows [19]). We conclude that the fluid model for evaluating TCP throughput performance over OBS requires significant improvements before it can be considered applicable.

The synchronization modeling approach proposed in [86] benefits from ACK-clocking to

include the burstiness factor in the fluid model. Note that the fluid model assumes that there is no burstiness and the TCP rates of different flows are differentiable. Therefore, it may take infinitely long to converge. The synchronization approach has been used for modeling Fast TCP and obtaining its stability by [85] and [86]. In order to obtain sufficient analysis of TCP performance while considering TCP stability, scalability, and responsiveness, the synchronization modeling approach needs to capture the bufferless nature of OBS links (*i.e.*, fixed RTTs), the burst aggregation factors, and the burst-loss distributions.

7.3 Recommendations

From our study of TCP over OBS, we observe that delay-based TCPs, in particular TCP Vegas, perform wore than other TCP congestion-control categories. Delay-based TCPs were proposed to detect network congestion at earlier stages by sensing the RTT fluctuation. In IP networks, it has been proven that Vegas improved TCP throughput by 30% to 70% compared to dropping-based TCP (*e.g.,* Reno). However, Vegas cannot detect network congestion in barebone OBS due to the fixed RTT. Also, Vegas falls into false-congestion detection due to the sudden increase in RTT when OBS uses burst retransmission or deflection. Therefore, we conclude that the delay-based TCPs are not suited for OBS networks.

Considering the TCP design requirements presented in Section 7.2, dropping-based TCP, in particular TCP SACK, is suited best for OBS. Conventional dropping-based TCP does not require RTT values to perform congestion control. However, collecting historical information about RTTs can provide further advantages for determining network congestion and burst-loss status. Maintaining statistical RTTs can be done by TCP senders as in SAIMD, or can be obtained explicitly from the OBS layer as in TCP-BCL. There is a trade-off among the various schemes. For example, SAIMD introduces extra computation overhead at the TCP sender side. On the other hand, TCP-BCL requires extra signalling between the OBS edge node and the TCP senders. If a burst that contains many packets from different TCP sources is dropped due to random contention, the edge node is responsible for signalling the affected senders. This signalling effort complicates the functionality of the edge node, which in many cases corresponds to hundred of thousands of flows.

In SAIMD, the cost of maintaining historical RTTs and deriving the appropriate β is nonetheless a trade-off with the long convergence time in recovery from slow-start caused by false congestion detection in long-lasting high-bandwidth TCP flows. Furthermore, the computation for the autocorrelation and confidence interval is performed for each loss event and has a fairly fixed complexity regardless of M and N sizes. Therefore, we do recommend using SAIMD approach for TCP over OBS since the resultant additional overhead to the whole network is trivial.

Appendix A
Acronyms

AAP	:	Adaptive Assembly Period
AIMD	:	Additive Increase Multiplicative Decrease
AO	:	All Optical
ARQ	:	Automatic Repeat Request
ATCP	:	TCP for Mobile Ad Hoc Networks
BACK	:	Burst Acknowledgement
BAIMD	:	Burst Additive Increase Multiplicative Decrease
BCR	:	Burst Contention Recovery
BIC	:	Binary Increase Control
BLE	:	Burst Length Estimation
BNACK	:	Burst Negative Acknowledgement
BTCP	:	Burst TCP
burst_wd	:	Burst Window
CDF	:	Cumulative Density Function
CSFQ	:	Core Stateless Fair Queue
cwnd	:	Congestion Window
DSACK	:	Duplicated Selective Acknowledgement
E/O	:	Electrical to Optical
ECN	:	Explicit Congestion Notification
ELN	:	Explicit Loss Notification
FACK	:	Forward Acknowledgement
FDL	:	Fiber Delay Lines
FEC	:	Forward Error Correction
FTO	:	False Timeout
GAIMD	:	Generalized Additive Increase Multiplicative Decrease
HPC	:	High-Performance Computing

HSTCP	:	High Speed TCP
IP	:	Internet Protocol
JET	:	Just Enough Time
JIT	:	Just In Time
JTCP	:	Jitter-Based Transmission Control Protocol
LAUC-VF	:	Latest Available Unscheduled Channel with Void Filling
MIMD	:	Multiplicative Increase Multiplicative Decrease
MPLS	:	Multi-Protocol Label Switching
NS	:	Network Simulator
O/E	:	Optical To Electronic
OBS	:	Optical Burst Switching
OCS	:	Optical Circuit Switching
OPS	:	Optical Packet Switching
PDF	:	Probability Density Function
PMF	:	Probability Mass Function
QoS	:	Quality of Service
RC	:	Retransmission Count
RCDP	:	Retransmission-Count Based Dropping Policy
RED	:	Random Early Discard
RTT	:	Round Trip Time
rwnd	:	Receiver Window
SABUL	:	Simple Available Bandwidth Utilization Library
SACK	:	Selective Acknowledgement
SAIMD	:	Statistical Additive Increase Multiplicative Decrease
SONET	:	Synchronous Optical Networking
STCP	:	Scalable TCP
SV	:	Min-Starting Void
TCP	:	Transport Control Protocol
TCP-BCL	:	TCP with Burst Contention Loss
TD	:	Triple Duplicate

TO	:	Timeout
TOP	:	Timeout Period
UDP	:	User Datagram Protocol
VFO	:	Virtual Fixed Offset Time
WDM	:	Wavelength Division Multiplexing
WR	:	Wavelength Routed

Bibliography

[1] I. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A new congestion control scheme for satellite IP networks", *IEEE/ACM transaction on networking*, vol. 9, no. 3, pp 307-321, 2001.H. Balakrishanan,

[2] C. Barakat, E. Alman, and W. Dabbous, "On TCP performance in a heterogeneous network: a survey", *IEEE Communication Magazine*, vol. 38 no. 1 pp. 40-46, 2000.

[3] T. Battestilli, H. Perros, "An introduction to optical burst switching," *IEEE Optical Communications*, vol. 41, pp 510-515, 2003.

[4] G. Bernstein, B. Rajagopalan, D. Saha, "Optical network control: Architecture, Protocol, and Standards", *Pearson Education Inc*., 2004.

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

[6] S. Bhandarkar, N. Sadry, N. Reddy and N. Vaidya, "TCP-DCR: a novel protocol for tolerating wireless channel errors", *IEEE Transactions on Mobile Computing*, 2004.

[7] S. Biaz and N. Vaidya, "De-Randomizing congestion losses to improve TCP performance over wired-wireless networks", *IEEE/ACM Transactions on networking*, vol. 13, no. 3, pp. 596-608, 2005.

[8] E. Blanton, M. Allman. "On making TCP more robust to packet reordering". *ACM Computer Communication Review*, vol. 32, no. 1, 2002

[9] L. Brakmo, S. Brakmo, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," Proceedings of *SIGCOMM*, 1994.

[10] 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.

[11] 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.

[12] F. Callegati, "Optical buffers for variable length packets", *IEEE Communication Letters*, Vol. 4, no. 9, 2000.

[13] C. Cameron, J. Choiy, S. Bilgrami, et al, “Fixed-Point performance analysis of TCP over optical burst switched networks”, Proceedings of *ATNAC*, 2004.

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

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

[16] X. Chen, H. Zhai, J. Wang, and Y. Fang, "A Survey on Improving TCP Performance over Wireless Networks," Kluwer Academic Publishers/Springer, *Resource Management in Wireless Networking*, vol. 16, pp. 657-695, 2005.

[17] X. Chen, H. Zhai, J. Wang, and Y. Fang, "TCP Performance over Mobile Ad Hoc Networks", *Canadian Journal of Electrical and Computer Engineering (CJECE) (Special Issue on Advances in Wireless Communications and Networking)*, vol. 29, no. 1/2, pp. 129-134, 2004.

[18] I. Chlamtac, A. Fumagalli, et al., “CORD: Contention Resolution by Delay Lines,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 1014-1029, 1996.

[19] A. Detti and M. Listanti,” Impact of segment aggregation on TCP Reno flows in optical burst switching networks, “ Proceedings of *IEEE Infocom*, 2002.

[20] K. Dolzer, C. Gauger, J. Spth, & S. Bodamer, “Evaluation of reservation mechanisms for optical burst switching”, *International Journal of Electronics and Communications*, vol. 55 no. 1, pp. 18–25, 2001.

[21] P. Du, and S. Abe, “TCP Performance Analysis of Optical Burst Switching Networks with a Burst Acknowledgement Mechanism”, Proceedings of *Asia-Pacific Conference on communication and International Symposium on Multi-Dimensional Mobile Communication* (APCC2004/MDMC2004), August, 2004.

[22] A. Erramilli, M. Roughan, D. Veitch, and W. Willinger, “Self-similar traffic and network dynamics,” Proceedings of the *IEEE*, vol. 90, no. 5, pp. 800-819, 2002.

[23] A. Falk, D. Katabi, "XCP Protocol Specification", Internet Draft, February, 2004.

[24] S. Floyd, “High-Speed TCP for large congestion windows”, RFC 3649, Experimental, December, 2003.

[25] S. Floyd, “TCP and explicit congestion notification”, *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, pp8-23, 1994.

[26] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, “An extension to the selective acknowledgment (Sack) option to TCP”, RFC 2883, 2000.

[27] S. Floyd, S. McCanne, “Network Simulator,” LBNL public domain software, available via ftp from ftp.ee.lbl.gov. *NS-2* is available at http://www.isi.edu/nsnam/ns/

[28] S. Floyd, T. Henderson, “The New Reno modification to TCP's fast recovery algorithm”. RFC 2582, 1999.

[29] C. Fraleigh, S. Moon, B. Lyles, et al, “Packet-Level Traffic Measurements from the Sprint IP Backbone”, *IEEE Network*, vol. 17 no. 6, pp. 6-16, 2003.

[30] C. Gauger, “Dimensioning of FDL Buffers for Optical Burst Switching Nodes”, Proceedings of the 6th *ONDM*, Torino, 2002.

[31] B. V. Gnedenko & I.N. Kovalenko, Introduction to Queuing Theory, Birkhäuser Boston, 1989.

[32] S. Gowda, R. Shenai, K. Sivalingam and H. Cankaya, "Performance evaluation of TCP over optical burst-switched (OBS) WDM networks", Proceedings of *IEEE International Conference on Communications*, pp. 1433-1437, 2003.

[33] J. Gubner, “Probability and random processes for electrical and computer engineers”, Cambridge University Press, pp 240-262, 2006.

[34] E. H.-K. Wu and M.-Z. Chen, “JTCP: Jitter-based TCP for heterogeneous wireless networks”, *IEEE JSAC*, vol. 13, no. 4, pp 757-766, 2004.

[35] J. He, S.-H. Gary Chan, “TCP and UDP performance for internet over optical packet-switched networks”, Proceedings of *IEEE ICC*, 2003.

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

[37] C. Hsu, T. Liu, and N. Huang, “Performance analysis of deflection routing in optical burst-switched networks,” Proceedings of *IEEE Infocom*, 2002.

[38] V. Jacobson, R. Braden, and D. Borman, “TCP extensions for high performance”, RFC 1323, 1992.

[39] C. Jin, D. Wei, and S. Low, “FAST TCP: Motivation, Architecture, Algorithms, Performance,” Proceedings of *IEEE Infocom*, 2004.

[40] A. Kaheel, and H. Alnuweiri, “Batch scheduling algorithms for optical burst switching networks”, *Proceedings of IFIP Networking,* 2005.

[41] A. Kaheel, H. Alnuweiri, F. Gebali, "Analytical Evaluation of blocking probability in optical burst switching networks", Proceedings of *IEEE ICC*, France, 2004.

[42] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," Proceedings of *ACM SIGCOMM*, 2002.

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

[44] T. Kelly, "Scalable TCP: improving performance in highspeed wide area networks", Proceedings of, *ACM SIGCOMM*, 2003.

[45] L. Kim, S. Lee, and J. Song, "Dropping Policy for Improving the Throughput of TCP over Optical Burst-Switched Networks", In the proceedings of *ICOIN*, 2006.

[46] J. Kurose, and K. Ross, "Computer networking a top-down approach featuring the Internet" Addison Wesley Longman, third edition, 2004.

[47] Y. Lee, and B. Mukherjee, "Traffic engineering in next-generation optical networks", *IEEE Communications Surveys and Tutorials*, vol. 6, no. 3, 16-33, 2004.

[48] S. Lee, L. Kim, "Drop Policy to Enhance TCP Performance in OBS Networks", *IEEE Communication Letters*, vol. 10, no. 4, 2006.

[49] J. Li, C. Qiao, and Y. Chen, "Recent progress in the scheduling algorithms in optical-burst-switched networks", Journal of optical networking, vol. 3, no. 3, pp 229-241, 2004.

[50] J. Li, C. Qiao, J. Xu, and D. Xu, "Maximizing throughput for optical burst switching networks", Proceedings of *IEEE Infocom*, 2004.

[51] J. Liu, and S. Singh, "ATCP: TCP for mobile Ad Hoc networks", *IEEE JSAC*, vol. 19, no. 9, pp 1300-1315, 2001.

[52] R. Ludwig, A. Gurtov, "The eifel response algorithm for TCP", Internet draft draft-ietf-tsvwg-tcp-eifel-response-05.txt, 2004.

[53] R. Ludwig, R. Katz, "The eifel algorithm: making TCP robust against spurious retransmissions", *ACM Computer Communications Review*, vol. 30, no. 1, pp. 30-36, 2000.

[54] M. Mathis, J. Semke, J. Mahdavi, T. Ott, "The macroscopic behaviour of the TCP congestion avoidance algorithm", *Computer Communication Review*, vol. 27, no. 3, pp. 67-82, July 1997.

[55] M. Mathis and J. Mahdavi, "Forward Acknowledgment: Refining TCP Congestion Control", Proceedings of *ACM SIGCOMM*, 1996.

[56] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," RFC 2018, 1996.

[57] V. Misra, W.-B. Gong, and D. Towsley, "Fluid based analysis of a network AQM routes supporting TCP flows with an application to RED," Proceedings of *ACM SIGCOMM*, 2000.

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

[59] I. Ogushi, S. Arakawa, M. Murata, and K. Kitayama "Parallel reservation protocols for achieving fairness in optical burst switching," Proceedings of *IEEE Workshop on High Performance Switching and Routing*, 2001.

[60] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," Proceedings of *ACM SIGCOMM*, pp. 303-314, 1998.

[61] J. Postel, "Transmission Control Protocol," RFC 793, Protocol Specification, DARPA Internet Program, 1981.

[62] C. Qiao, "Labeled optical burst switching for IP-over-WDM integration", *IEEE Communication Magazine,* vol. 38, no. 9, pp. 104-14, 2000.

[63] 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.

[64] S. Ryu, C.M. Rump, and C. Qiao, "Advances in internet congestion control," *IEEE Communications Surveys & Tutorials*. vol. 5, no. 1, pp 28-39, 2003.

[65] C. Samios and M. K. Vernon, "Modeling the Throughput of TCP Vegas," Proceedings, *International Conference on Measurement and Modeling of Computer Systems SIGMETRICS*, 2003.

[66] P. Sarolahti, M. Kojo, "F-RTO: an algorithm for detecting spurious retransmission timeouts with TCP and SCTP", internet draft draft-ietf-tcpm-frto-01.txt, 2004.

[67] M. Schlager, "Documentation on the eifel algorithm implementation for the network simulator (NS), 2000, available at http://www-tkn.ee.tu-berlin.de/~morten/eifel/

[68] M. Schlosser, E. Patzak, P. Gelpke, "Impact of deflection routing on TCP performance in optical burst switching networks", 7th *International conference on Transparent Optical Networks*, 2005.

[69] B. Shihada and P-H. Ho, "A Domain-based resource reservation protocol for next generation optical burst switching internet," *5th IASTED Optical Communication Systems and Networks (OCSN)*, Canada, 2005.

[70] B. Shihada and P-H. Ho, "Transport Control Protocol (TCP) in Optical Burst Switched Networks: Issues, Solutions, and Challenges", Accepted at *IEEE Communications Surveys and Tutorials*, 2007

[71] B. Shihada, P-H. Ho, and Q. Zhang, "A Novel Congestion Detection Scheme for TCP over OBS Networks", Proceedings of the *IEEE Globecome*, 2007.

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

[73] B. Shihada, P-H. Ho, F. Hou, and *et. al.*, "BAIMD: a responsive rate control for TCP over optical burst switched (OBS) networks," Proceedings of *IEEE International Conference on Communications* (ICC), 2006.

[74] B. Shihada, P-H Ho, and Q. Zhang, "SAIMD: A Congestion Detection Scheme for TCP over OBS Networks", submitted to *Journal of Lightwave Technology (JLT),* March, 2007.

[75] B. Shihada, P-H Ho, X. Jiang, & M. Guo," TCP-BCL: A Novel TCP Implementation with Dynamic Explicit Burst-Contention Loss Notification over OBS Networks", Submitted to *IEEE Transactions on Parallel and Distributed System*, April, 2007.

[76] B. Shihada, Q. Zhang, and P-H. Ho "Threshold-based TCP Vegas over Optical Burst Switched Networks", Proceedings of *15th IEEE International Conference on Computer Communications and Networks (ICCCN)*, 2006.

[77] B. Shihada, Q. Zhang, and P-H. Ho, "Performance Evaluation of TCP Vegas over Optical Burst Switched Networks", Proceedings of *IEEE Broadnets*, the 6th *International Workshop on Optical Burst/Packet Switching* (*WOBS*)*,* 2006.

[78] B. Shihada, Q. Zhang, P-H. Ho, & J. Jue, "A Novel Implementation of TCP Vegas for Optical Burst Switched Networks", submitted to the *IEEE Journal on Selected Areas in Communications* (*JSAC*), February, 2007.

[79] H. Sivakumar, R. L. Grossman, M. Mazzucco, Y. Pan, Q. Zhang, "Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks", *Journal of Supercomputing*, 2004.

[80] J. Soldatos, G. Kormentzas, “On the building blocks of quality of service in heterogeneous IP networks,” *IEEE Communications Surveys & Tutorials,* vol. 7, no. 1, 2005.

[81] W. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms,” *RFC 2001*, 1997.

[82] W. Stevens, “TCP/IP Illustrated”, Volume 1, Addison Wesley Longman, 1994.

[83] V. Vokkarane, J. Jue, and S. Sitaraman, “Burst segmentation: an approach for reducing packet-loss in optical burst switched networks”, *IEEE International Conference on Communications* (*ICC*), 2002.

[84] H. Vu, & M. Zukerman, “Blocking probability for priority classes in optical burst switching networks”, *IEEE Communications Letters*, vol. 6 no. 5, pp. 214–216, 2000

[85] J. Wang, D. Wei, S. Low, “Modeling and Stability of Fast TCP”, in the proceedings of *IEEE Infocom*, pp. 938-948, 2005.

[86] D. Wei, “Congestion control algorithms for high speed long distance TCP connections”, Master thesis, Caltech, 2004.

[87] B. Wen, N. Bhide, R. Shenai, and K. Sivalingam, "Optical Wavelength Division Multiplexing (WDM) Network Simulator (OWns): Architecture and Performance Studies", *SPIE Optical Networks Magazine Special Issue on Simulation, CAD, and Measurement of Optical Networks*, 2001.

[88] J. White, R. Tucker, and K. Long, “Merit-based scheduling algorithm for optical burst switching,” Proceedings of *International conference on optical networks*, 2002.

[89] J. Widmer, R. Denda, M. Mauve, “A Survey on TCP-Friendly Congestion Control”, *IEEE Network Magazine*, special issue on "Control of Best Effort Traffic", vol. 15, no. 3, pp. 28-37, 2001.

[90] C. Xin, C. Qiao, Y. Ye, and S. Dixit, “A Hybrid Optical Switching Approach,” Proceedings of *IEEE Globecom* 2003.

[91] Y. Xiong, M. Vandenhoute, and H. Cankaya, “Control architecture in optical burst-switched WDM networks,’ *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838-51, 2000.

[92] J. Xu, C. Qiao, J. Li, and G. Xu, “Efficient channel scheduling algorithms in optical burst switched networks”, Proceedings of *IEEE infocom*, 2003.

[93] L. Xu, K. Harfoush, and I. Rhee, “Binary increase congestion control (BIC) for fast long-distance networks”, Proceeding of *IEEE infocom*, 2004.

[94] J.Y. Wei and R. I. McFarland, “Just-In-Time signalling for WDM optical burst switching networks”, *Journal of Lightwave Technology*, vol. 18, pp. 2019–2037, 2000.

[95] S.Y. Wang, "Using TCP congestion control to improve the performances of optical burst switched networks," Proceedings of *IEEE ICC*, 2003.

[96] Y. Yang, S. Lam, “Generalized AIMD congestion control,” University of Texas, Technical Report TR-2000, 2000.

[97] M. Yoo, C. Qiao, “Just-Enough-Time (JET): A high speed protocol for bursty traffic in optical networks”, Proceedings of *IEEE/LEOS Technology for Global Information Infrastructure*, 1997.

[98] X. Yu, C. Qiao, and Y. Liu, “TCP implementation and false time out detection in OBS networks,” Proceedings of *IEEE Infocom*, 2004.

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

[100] X. Yu, Y. Chen, and C. Qiao, “Study of traffic statistics of assembled bursts in optical burst switched networks”, Proceedings of *Opticomm*, 2002.

[101] A. Zalesky, Le Vu Hai, M. Zukerman, Z. Rosberg, and E.W.M Wong, “Evaluation of limited wavelength conversion and deflection routing as methods to reduce blocking probability in optical burst switched networks”, Proceedings of *IEEE International Conference on Communications* (ICC), 2004.

[102] Q. Zhang, V. M. Vokkarane, B. Chen, and J. P. Jue, “Early drop scheme for providing absolute QoS differentiation in optical burst-switched networks”, Proceedings of *IEEE High Performance Switching and Routing*, 2003.

[103] Q. Zhang, V. Vokkarane, J. Jue, and B. Chen, “Absolute QoS Differentiation in Optical Burst-Switched Networks,” *IEEE Journal on Selected Areas in Communications, Optical Communications and Networking Series*, vol. 22, no. 9, pp. 1781–1795, Nov. 2004.

[104] Q. Zhang, V. Vokkarane, Y. Wang, and J. P. Jue, “Analysis of TCP over Optical Burst-Switched Networks with Burst Retransmission,” Proceedings of *IEEE Globecom*, 2005.

[105] Q. Zhang, V. Vokkarane, Y. Wang, and J. P. Jue, "Evaluation of Burst Retransmission in Optical Burst-Switched Networks," Proceedings of *2nd International Conference on Broadband Networks*, 2005.

[106] B. Zhou, M. Bassiouni, G. Li, "Improving fairness in optical-burst-switching networks", *Journal of Optical Networking*, vol. 3, no. 4, pp.214, 2004.

[107] 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.

[108] L. Zhu, N. Ansari, J. Liu," Throughput of high-speed TCP in optical burst switching networks", *IEE proceedings Communications*, vol. 152, no. 3, pp. 349-352, 2005.