Transport Control Protocol in Optical Burst Switched Networks: Issues, Solutions, and Challenges

Basem Shihada, and Pin-Han Ho, University of Waterloo

Abstract

Since its advent in 1981, TCP has been subject to a tremendous amount of research efforts and enhancements for achieving better performance over various network environments and application scenarios. Due to the transmission characteristics of Optical Burst Switched (OBS) networks, such as random burst dropping, retro-blocking (*i.e.*, bursts proceeding or delayed from their actual reservation time slot), burstification delay, and burst signaling delay, TCP could be significantly affected in case no corresponding countermeasure and enhancement is developed. In this review article, we provide a comprehensive survey on reported studies for TCP enhancements over OBS networks in order to mitigate the numerous side effects due to the bufferless characteristic of burst transmission. Furthermore, we closely analyze TCP behavior over OBS networks with various burst transmission characteristics while highlighting the open challenges that have not yet been extensively tackled or solved.

I. Introduction

Currently, the complex Internet infrastructure interconnects millions of communication devices and computing equipments through wired and wireless connections. One of the key success factors of today's Internet infrastructure has greatly relied on the ability of maintaining reliable, self-regulated, and congestion-tolerant transport protocol that serves the end-user applications. Transport Control Protocol (TCP) [1], which was designed for military communication by ARPANET [2], has been taken to serve as the most pervasive and well recognized standard for a majority of currently available Internet-based applications. While analyzing the statistical network data, it has been reported that TCP is the most predominant protocol in terms of the traffic volume (in Bytes), which may take up to 90% of the total Internet traffic [3]. The High-Performance Computing (HPC) networks observed an average share of 83%, and for the NETI@Home data the authors found that TCP flows contribute the major traffic volume seen in our datasets.

A. Introduction on TCP

TCP congestion control mechanisms can be classified into the following three categories: (1) loss-based (*e.g.*, Reno [4] and Sack [5]), (2) delay-based (*e.g.*, TCP Vegas [6], Fast TCP [8][9]), and (3) explicit notification-based (*e.g.*, XCP [7]). They are basically the approaches taken by a TCP sender to determine if the network is in a congestion state such that the transmission rate is adjusted accordingly, while the receiver can be totally reactive to the transmission protocol.

The standard dropping-based TCP protocol stack, *e.g.*, TCP Reno, which follows an Additive Increase Multiplicative Decrease (AIMD) window-based congestion control mechanism for regulating data transmission and maintaining network bandwidth usage simply takes a packet (or TCP segment) dropping event as the indication of network congestion. The AIMD framework is comprised of four stages for each TCP sender: *slow start*, *congestion avoidance*, *fast transmission*, and *fast recovery* [4].

At the slow start phase, the TCP sender has just started or restarted transmitting due to a *Timeout* (TO) event and the *congestion window* (*cwnd*) is initialized to the size of one segment. The size of *cwnd* is exponentially increased at each successfully delivered and acknowledged segment. The exponential growth of the *cwnd* continues until either it exceeds the receiver's advertised *cwnd* or a packet loss is reported, which leads the TCP protocol into the congestion avoidance stage. Note that the network congestion state is reported either when the transmission TO threshold is reached or when the receiver collects triple-duplicated acknowledgments (TD). In the former case where a TO event occurs, which indicates a heavy congestion, the *cwnd* is set to one segment, and the TCP returns to the slow start stage followed by the congestion avoidance stage. In the latter case, which indicates a light congestion, TCP enters the fast retransmission stage by taking a half of the sender's *cwnd* as the slow start threshold *ssthresh* and setting the *cwnd* to *ssthresh* plus three segments. Then the sender retransmits the lost segment and increments its *cwnd* by the segment. After acknowledging the second data segment, the *cwnd* is set to the *ssthresh*.

Different from the dropping-based TCP, delay-based TCP implementations, such as TCP Vegas [6] and Fast TCP [8][9], estimate the bandwidth and congestion status in the network by measuring the delay of each packet transmission in terms of *round trip time* (RTT). The performance of Fast TCP has been evaluated in [8][9], which can be considered as an enhancement of TCP Vegas with better throughput. Nonetheless, in the following context will focus on the introducing TCP Vegas congestion control scheme for better comprehensiveness.

TCP Vegas modifies TCP Reno in the congestion avoidance, slow start, and retransmission stages, which determines the congestion status in the network according to the comparison of estimated and measured throughputs. TCP Vegas first computes *BaseRTT* as the minimum measured RTT, which is mainly determined by the propagation delay and the queuing delay. The *Expected* throughput can thus be derived as $Expected = cwnd / BaseRTT$, where *cwnd* is the current congestion window size. Second, the *Actual* throughput is calculated for every RTT using the most recent measured RTT: $Actual = cwnd / RTT$. TCP Vegas sender then computes the difference between the two quantities (denoted as *Diff*):

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

$$= (\frac{cwnd}{BaseRTT} - \frac{cwnd}{RTT})BaseRTT$$

$$= cwnd(1 - \frac{BaseRTT}{RTT}).$$

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

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

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 the *cwnd* unchanged.

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

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

When TCP Vegas sender receives an acknowledgement (*ack*), it records the clock and calculates the estimated RTT using the current time and the timestamp recorded for the associated packet. Vegas then decides whether to retransmit the packet based on the following two conditions: First, when a 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 value. If true, the Vegas sender retransmits the packet without having to wait for the remaining incoming duplicated *ack*s. Second, when an *ack* is received, if it is the first or the second *ack* after a retransmission, Vegas again checks if the time interval since the packet was sent is larger than the TO value. If it is, the Vegas sender retransmits the packet. Since this will catch any other packet that may have been lost prior to the retransmission without having to wait for the remaining duplicate *ack*, hence, Vegas retransmission mechanism reduces the time in detecting a lost packet from the third duplicate *Ack* to the first or the second. After the packet retransmission is triggered by a duplicate *ack*, the *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 TCP congestion-control mechanism belongs to explicit notification-based TCP. In this category, TCP with Explicit Loss Notification (ELN) [10], or Explicit Congestion Notification (ECN) [11] are the two representative approaches, which were developed to identify suspicious packet losses which may cause unnecessary retransmissions. Furthermore, TCP ELN/ECN can distinguish packet losses among congestion, contention, link failure, or other reasons. The packet retransmission starts as soon as the ECN or ELN is received, which is the round after the one that encounters a packet loss.

B. Evolution of Optical Switching Technologies

Optical backbones based on WDM are the most prevalent transportation carrier in modern communication networks. Several switching technologies have been proposed to take advantage of the high-transmission capacity of fiber optics. Early approaches followed Optical Circuit Switched (OCS), where point-to-point lightpaths are established for a relatively long period of time. 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, since the number of available wavelengths is limited to few tens or hundreds, the task of Routing and Wavelength Assignment (RWA), which optimizes the assignment of wavelengths to all users, is NP-Hard. Second, due to its quasi-static nature, OCS can barely support dynamic traffic variation and frequent connection requests.

Due to the ubiquity of the Internet Protocol (IP), much research has addressed the integration of IP with WDM networks. Optical Packet Switching (OPS) [13][14][24] 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) [21] until the switch configuration is completed. OPS follows 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 system can be commercialized due to the high cost of all-optical buffer facilities and the dimensioning of these delay lines.

Optical Burst Switched (OBS) networks have 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 [13][14]. Compared with OPS, OBS is more practical to implement, and combines the best of OCS and OPS networks. With OBS, 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 [13][14]. Since the bursts cut-through each intermediate node in the network core while the control packets are subject to processing at each core node, certain amount of offset time must be imposed between lunching control packets and the corresponding bursts. The calculation of the offset time has to consider the upper bound on number of hops and nodal processing delay. OBS follows AO transmission approach. With its out-of-band signaling 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

C. Optical Burst Switched (OBS) Networks

OBS networks are basically bufferless yet best-effort in nature [12][13][14][15][16]. Basically, with IP over OBS networks, the IP applications and services are overlaid upon the OBS backbone, and the two layers are interconnected through OBS edge nodes and the corresponding IP routers. The term "OBS domain" generally depicts the optical network with Wavelength Division Multiplexing (WDM) technology formed by a group of OBS edge and core nodes, which run a suite of dedicated routing and signaling protocols. The OBS domain routing and signaling protocols could be independent from the upper IP compatible protocols, where an overlay model is followed. Alternatively, the IP and OBS domains may run a joint protocol by making the whole network into an augmented service model, where some extent of routing and traffic engineering information exchange is performed. Our discussions in the paper focus on the latter case in order to achieve a cross-layer design.

Within the OBS domain, a data burst is formed at an OBS ingress edge node by assembling multiple incoming IP packets with a common OBS egress node and similar quality of service (QoS) requirements [12][13], as shown in figure 1. Since each IP-access router is attached to an OBS edge node, the OBS domain which provides high-speed transportation, can serve as the link-layer transmission media for the IP-access routers.
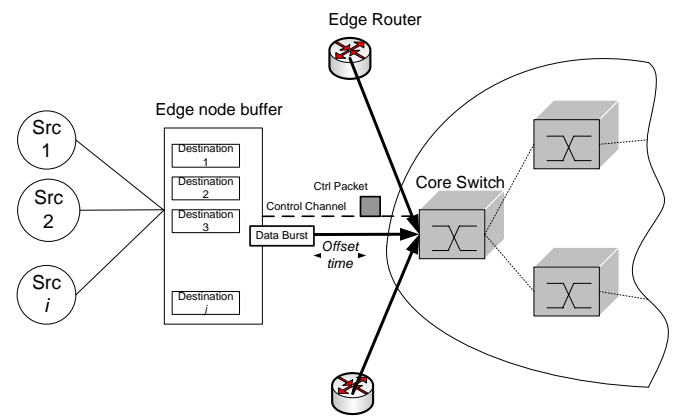

Figure 1. *An illustration of OBS networks*

The bufferless and AO nature make the OBS networks distinguished from the traditional packet-switched networks which significantly rely on the electronic processing capability [12]. The implementation of OBS networks requires more precise signaling and a much higher switching speed than that required by Optical Circuit Switched (OCS) networks [15][16]. The edge nodes are capable of performing electronic-optical (E/O) and optical-electrical (O/E) conversion to collect/sort/assemble incoming IP packets from the higher layers into optical data bursts, and dissemble the optical data bursts back to the IP packets at the destination edge nodes. The edge nodes are also responsible for signaling and routing in the OBS domain. The core switches, on the other hand, are subject to configuration requests via control packets sent by the OBS edge nodes, where the data bursts are forwarded and cut through the OBS domain in an AO manner [12][13][14][15][16].

A burst contains several other fields when assembled at the edge node. Figure 2 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 help to overcome the uncertainty of data burst arrival time and data burst duration due to clock drifts between different core nodes. The payload basically consists of the assembled incoming 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 [13].
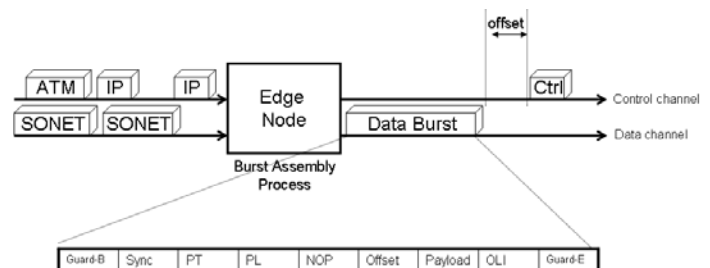

Figure 2. *OBS edge node architecture*

Typically, an OBS core node (switch) consists 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 switching resources are freed after the burst cut-through the switch. The switch matrix control unit and the port forwarding table, which is a lookup table and link scheduling [35][36][37][38][39] module are maintained. The lower layer is responsible for all-optical burst transport functionality. The lower layer consists of optical ports, wavelengths, and optical-to-optical connections. Figure 3 illustrates the generic OBS core switch architecture.
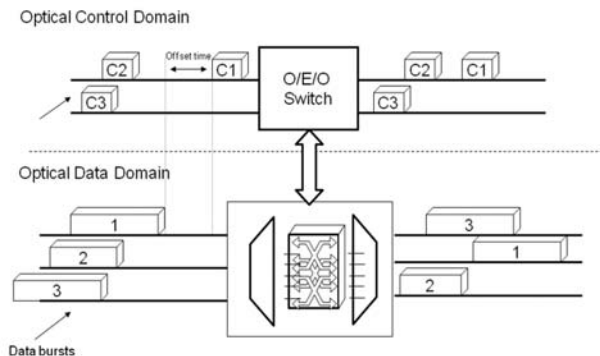

Figure 3. *OBS core switch architecture*

Fixed-path routing is a general approach taken in the OBS domain, where a physical route is pre-defined for each pair of OBS edge nodes. In order to deliver a burst from the source OBS edge to a certain remote one, a corresponding control packet that keeps the length of the burst and the arrival time of the burst at each intermediate node of the given route is created at the edge node and is sent prior to the launching of the data burst [15]. Since the burst cuts-through the intermediate core nodes without any processing delay, the control packet is intrinsically slower than the burst. In order to prevent the burst from catching-up with its control packet, it is important to define *offset time* between the time instants of lunching the burst and the control packet [15][16] as shown in figure 2 and 3. This is also referred to as *delayed reservation*, which serves as the basis for OBS operation. The offset time is added to the average delay of the burst delivery with its minimum value determined by the number of hops along the route that the control packet traverses through. Then the data burst cuts-through the network following their control packet in a one-way (*i.e.*, Tell-and-Go) signaling fashion.

Although OBS can achieve much better flexibility and efficiency than the OCS based wavelength-routed networks, OBS suffers from burst contention which leads to burst drop due to the one-way resource reservation signaling and the lack of optical buffering. Burst contention occurs when more than one burst simultaneously attempt to take a common output port or wavelength channel [12][13], which results in dropping $n - 1$ bursts when there are n contended bursts. Burst retransmission [17][18], burst deflection routing [19][20], Fiber Optic Delay Lines (FDL) [21], burst segmentation [22], and wavelength conversions [23] are some reported

approaches for increasing the reliability of the burst transmission in the OBS domain. Each scheme has imposed some impacts on TCP performance, which motivated numerous related studies to be seen in the past decade.

D. Overview of the Article

The goal of this article is to give an overview on the state of the art of TCP developments over OBS networks, including the recent proposed modifications and research challenges in such a deployment scenario. In addition, we are committed to conduct a comprehensive survey and tutorial on the reported solutions in TCP over OBS networks by classifying them into the following three categories: (1) link-layer solutions (*i.e.*, solutions implemented at OBS domain), (2) modifications or enhancements based on conventional TCP (TCP originally designed for IP packet-switched networks), and (3) newly designed TCP congestion-control mechanisms.

With the first category, the solutions, which aim to asset TCP in understanding various burst transmission behaviors, require an implementation of additional supporting functions at either the OBS edge node or the OBS core node. Generally there are two problems incurred in this solution category. First, it breaks the TCP end-to-end semantics. Second, it introductions additional control overhead that impair the applicability of the scheme. The solutions in the second category can overcome the abovementioned problems by preserving the end-to-end semantics of TCP congestion control such that no extra signaling overhead is required by any intermediate network device except for the TCP senders and receivers. However, the price is paid at the effectiveness of the scheme, which may make TCP senders hard to follow the fluctuation of network congestion state in the OBS domain. Finally, the third category of solutions includes a totally new transport protocol that copes with the underlying burst transport behaviors.

The rest of the paper is organized as follows. In Section II, the issues of running TCP over OBS networks are identified, where the problems due to the bufferless transmission characteristics in the OBS networks are formulated. In Section III, a comprehensive survey is given on the recently proposed solutions for TCP over OBS networks in all the three categories. Further discussions on the algorithms and some open research problems are given in Section IV. Finally, we summarize and conclude the paper in Section V.

II. ISSUES OF TCP OVER OBS NETWORKS

In this section, the taxonomy of OBS networks, namely, *barebone OBS* and *burst contention resolution (BCR) OBS*, are introduced. The transmission characteristics of each type are further identified. Afterwards, we discuss the fundamental problems of running TCP over both types of OBS networks.

A. Taxonomy of OBS Networks

Considering the OBS resource-reservation-signaling mechanisms, implicit [15] and explicit [16] resource reservations have been reported. In terms of burst assembly mechanisms, time-based, burst-length-based, mixed (both time

and burst length), and optimized burst-size-based OBS networks have been reported [12][13][14]. Since burst contention resolution mechanisms play vital role in the deployment of TCP over OBS, this article classifies OBS networks according to whether there exist any burst contention resolution mechanism in OBS domain, where two categories can be derived, namely barebone OBS and burst contention resolution (BCR) OBS.

- **Barebone OBS**

In barebone OBS, control packet is sent to reserve the intermediate switching fabric in a one-way signaling scheme. After a predefined offset time, the burst cuts-through certain path until it reaches the destination. In literature, two major resource-reservation protocols in OBS domain, called Just-In-Time (JIT) and Just-Enough-Time (JET) are widely investigated. The JIT resource-reservation protocol introduced in [16] follows the explicit-reservation approach. Upon receiving the control packet, the switching fabric is reserved. 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 along the same route to release the reserved resources.

The JET resource-reservation protocol introduced in [15] follows the delayed reservation with implicit resource release approach. The intermediate core nodes which are traversed by the burst can automatically release the reserved switching fabric shortly after the burst leaves the switch. Since both schemes never guarantee the channel availability, in case two or more data bursts attempts to access a common output port simultaneously, one of them must be dropped. The dropped bursts will be lost since barebone OBS does not employ any buffering or burst contention resolution mechanism. It is worth noting that the burst contention losses occur even while the traffic load is low.

- **OBS with Burst Contention Resolution (BCR)**

In contrast to the barebone OBS networks, BCR OBS defines the efforts of burst contention resolution such as retransmission of the contended bursts at the edges [17][18], burst deflection routing [19][20], optical buffering [21], burst segmentation [22], and wavelength conversion [23], etc. With burst retransmission, deflection, or buffering, the bursts subject to contention in OBS domain can be retransmitted at the edge node, deflected to an alternate route at the core node, or buffered using FDL at the core node, respectively. Hence, the burst random contention, which is the case where a data burst is dropped when the channel utilization is low, may contribute to a significant portion of the total burst loss probability. With burst segmentation, the contended bursts are segmented at the core node such that the overlapped segments are discarded, while the rest of the burst can successfully be transmitted. With wavelength conversion, optical data flows on a certain wavelength can be converted to another wavelength, which can effectively reduce the resource segmentation and result in a better chance of successfully forming a data path.

The above burst contention resolution schemes have been proved to effectively increase the transmission reliability and throughput in OBS network domain at the expense of introducing extra overhead, delay, and computation complexity, which will be discussed in details later.

B. Characteristics of OBS Networks

The bufferless nature of OBS results in two major transmission characteristics that distinguish OBS networks from any other type of networks, which are, (1) random burst contention losses at low traffic loads and (2) assembling (Burstification) and signaling delay.

- **Assembling (Burstification) and Signaling Delay**

Several burst-assembly algorithms were proposed for OBS networks and can be divided into the following four categories [12][13][14]. The first category is time-based in which a timer is set after the beginning of each new assembly cycle. With this assembly algorithm, all IP packets arriving within a specific period of time are considered to be assembled into a burst assuming they have a common destination edge node. The second set of algorithms is the burst-length-based, where a burst-length threshold is set. With these schemes, the burst length threshold serves as the minimum burst length before the burst leaves the network. The third category combines the time and the burst length-based, where the bursts are assembled and sent either when the burst-length exceeds the desirable threshold or the timer expires. The last category includes adaptive-assembly algorithms, where a dynamic adaptive-threshold on the burst length is set in order to optimize the overall performance in the OBS domain for QoS sensitive traffic [24]. Both barebone OBS and BCR OBS experience the burstification and burst signaling delays.

- **Random Burst Contention Losses**

In IP-based networks with electronic processing, contended packets can be temporarily stored in the memory of intermediate IP routers instead of being dropped immediately. Except for extreme cases such as buffer overflow caused by persistent network congestion, packets can be delivered in a hop-by-hop manner until it reaches the destination. Nonetheless, this is not the case in OBS networks. As a compromise between OCS and OPS networks, OBS facilitates some extent of statistical multiplexing by using JIT or JET resource-reservation schemes while buffering the bursts at the OBS edges only. Due to the adoption of a one-way signaling mechanism and the lack of global scheduling, burst contention and resultant dropping could happen even when the network is lightly loaded [12]. This is referred to as *random burst contention* losses, which may impose a significant vicious impact on the TCP protocol stacks especially for those which take packet-loss events as the only indication for network congestion. In contract with random burst contention losses, the network could be subject to persistent network congestion, where a high utilization state of the OBS network lasts for much longer time.

C. Issues of TCP over OBS

It has been shown that the burstification process at OBS edge nodes influences the performance of TCP Reno [25]. TCP flows are classified into three categories: fast, medium, and slow. In fast TCP flow category, TCP flow is subject to high IP-access bandwidth such that the entire transmitted segments of the *cwnd* are assembled in a single burst. On the other hand, the medium and slow flows have a proportion of TCP segments in the *cwnd* assembled in a single burst due to limited IP-access bandwidth. For medium to slow flows, a performance penalty is introduced on TCP sending rate due to assembly delay, while the aggregation of multiple TCP segments due to burst assembly has given TCP a throughput performance gain [25][26]. When a burst is lost either due to persistent network congestion or random burst contention loss, fast TCP flows returns to slow start since the TCP segments of the entire *cwnd* are lost, which indicates severe network congestion [27]. However, the medium and slow flows enter the fast retransmission stage since some of the TCP segments of the current *cwnd* are lost [27].

- **Multiple TCP Segments Loss**

In TCP over OBS networks, TCP performance is significantly affected by random burst contention loss [25][26][27], which causes unnecessary *cwnd* cut. Since an OBS edge may correspond to millions of TCP senders [3], hundreds or thousands of TCP segments could be assembled and transported together in a single burst. At the occurrence of a burst loss, the TCP with fast flows will be significantly affected by falling into the false congestion detection or also called false TO (FTO) [27]. However, in the case of medium and slow flows, the probability of having multiple segments of a single TCP session assembled in a single data burst is low, a burst loss will less influence a single TCP flow. However, it results of affecting a large number of TCP flows which has a segment assembled in the lost burst. Depending on the flow speed [25], TCP usually suffers from FTO [27] or perform unnecessary *cwnd* cuts [25][27].

- **Packet Reordering**

The out-of-order burst delivery may occur due to the burst assembly mechanism, where the TCP segments of certain TCP session are assembled in two or more bursts that are sent in a different order from that of the TCP segments. Also, bursts may be sent in a correct order but received out-of-order due to burst drop, retransmission or delays from deflection. The frequency of the out-of-order delivery depends on the time and burst-length thresholds, the traffic shape, or the QoS criteria defined in the burst assembly algorithm. The out-of-order delivery may cause a TCP sender to improperly retransmit a TCP segment since the TCP receiver will simply issue TD *acks* by assuming that there exists a sequence of missing segments. Depending on the number of the 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 dramatically affect the sender in presence of a large number of

TCP segments assembled in a single burst. The higher rate the TD *acks* occur, the smaller the TCP sender *cwnd* remains, which significantly throttles the throughput. This problem is also referred to as the global synchronization problem.

- **Slow Convergence**

For high-bandwidth TCP flows that last for a long period of time and span over long distances, the TCP throughput may suffer from a very long convergence time due to the slow additive-increase congestion avoidance mechanism. The following gives an example. Let we have a 10Gbps TCP flow with 1.5KB TCP packet size. Let the RTT be 100 ms. Thus, it takes about an hour to converge to the rate of 10 Gbps assuming that the packet loss rate is less than 10^{-8}. The slow convergence problem is expected to be worsened in OBS networks due to a longer RTT of each data burst caused by the burstification process and BCL delay [28].

With the additive increase of *cwnd* for each successful round of burst delivery, TCP over OBS networks may take a significant amount of time in converging to the available high bandwidth. On the other hand, the multiplicative decrease for each segment loss at the TCP senders is a critical response to the network congestion, even if we exclude the possibility of any burst drop due to random burst contention. Note that it is necessary to maintain a large *cwnd* to achieve a high throughput, which can only happen when an extremely low packet loss rate is seen. This is not likely to happen in presence of random burst losses in the OBS networks.

D. Taxonomy of Solutions for TCP over OBS

In bufferless OBS networks, TCP senders should not blindly consider the loss of the data segments as an indication of network congestion; instead, TCP should attempt to collect further information and network states to distinguish a packet-loss event between due to random burst contention and persistent network congestion. TCP implementations of this category may require extra signaling efforts between the TCP senders and the OBS domain [27], and/or within the OBS domain [17][18][29]. In this paper, three categories of previously reported solution for improving TCP over OBS are briefly described as follows.

- **Link-Layer Solutions**

With link-layer solutions, the OBS domain is simply treated as the link layer underneath the IP edge routers. Thus, the OBS burst transmission is taken transparent to the upper TCP senders, where the TCP throughput is improved by adopting some mechanisms such as an adaptive-burstification algorithm [24], a forward-error-correction (FEC) mechanism, a burst-contention resolution schemes, and automatic repeated request (ARQ) [17][18], *etc*. The fundamental advantage of employing a link-layer solution is to hide the non-congestion burst loss from the TCP senders, such that the layered structure of network protocols is followed by keeping local burst transmission reliability. To achieve this link-layer transmission reliability, burst retransmission, deflection, and adoption of FDLs in the OBS domain are common

approaches. These approaches have contributed to reduce the *cwnd* fluctuation and consequently improve the TCP throughput at the expense of introducing additional delay and control overhead. In the rest of the article, we term the schemes functioning only within the OBS domain as *link-layer solutions* to reflect the fact that the OBS domain can be simply taken as the link layer transmission seen by the IP routers.

• **Congestion Detection without Explicit Notifications**

With the solutions for congestion detection without explicit notifications, the TCP senders try to evaluate the states in the OBS domain by way of some equations, measured RTT, and/or statistic tools. In this case, TCP can maintain and analyze a number of previous RTTs at the TCP senders in order to identify if a packet loss event is due to persistent congestion or random burst contention. Under this category, we identify the following sender side congestion-control schemes, Statistical AIMD, Threshold-based Vegas, Burst TCP with burst length estimation (BLE), and OBS with General Additive Increase Multiplicative Decrease (BAIMD). There are other existing TCP solutions such as TCP Sack, duplicated Sack, and Fack that can work over OBS networks without explicit signaling [26][27]. These schemes are excluded from our review study since they are not originally proposed for OBS environments. However, the throughput performance of these TCP implementations over OBS networks has been examined through simulation in [43].

In case of TCP fast flows, a burst that is successfully retransmitted, deflected, or buffered is subject to additional delay due to the contention-resolution mechanisms. The additional delay is inevitably added on top of the RTT of all the TCP segments assembled within the burst. In this case, the delay-based TCP sender (*e.g.*, TCP Vegas) detects sudden increase in RTT of those segments, and interprets the additional delay as an indication of network congestion. This is another form of false congestion detection by the delay-based TCP, which has been identified to fatally impair the TCP throughput due to unnecessary TO retransmissions followed by slow start at the TCP senders. Threshold-based TCP Vegas was proposed to cope with the above mentioned problem [49][50]. This scheme is considered a sender side TCP implementation that does not require any explicit notifications.

• **Congestion Detection with Explicit Notifications**

The solutions of congestion detection with explicit notification are basically cross-layer designed such that the states in the OBS domain can be leaked to the upper TCP senders. In this case, TCP senders can additionally be informed of the channel conditions and the actual cause of the packet loss, such as network congestion, packet corruption, or any possible hardware component failure. TCP senders will retransmit the lost segments without affecting their *cwnd* for packet losses due to any reason other than network congestion. This category folds the TCP implementations with explicit congestion or loss notifications such as OBS TCP (BTCP)

(Back/BNack) and TCP with explicit burst contention losses (TCP-BCL).

As a summary of this section, we have briefly introduced and classified the previously reported solutions for TCP in OBS networks. As shown in figure 4, the following three main categories were highlighted, namely, link-layer solutions, TCP congestion detection without explicit notifications, and TCP congestion detection with explicit notifications.
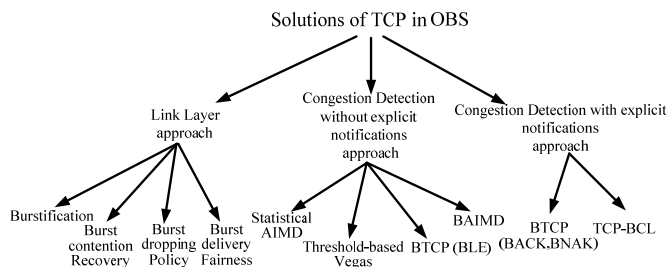

Figure 4. *The taxonomy of solutions for TCP in OBS networks*

III. OVERVIEW OF EXISTING SOLUTIONS

This section explicitly surveys all the three categories in the aspects of dropping-based (*e.g.*, Reno), delay-based (*e.g.*, Vegas), and the explicit notification (*e.g.*, TCP ELN/ECN) based TCP implementations.

A. Link-Layer Solutions

• **Solutions based on Burstification Processes**

It has been reported that the delay caused by the burstification process at the OBS ingress nodes will enlarge the TCP RTTs and may lead to TO in case the TO threshold is not well manipulated, which has a potential to impair the TCP throughput due to non-congestion caused *cwnd* cut [24][25][26]. In [25], the authors examined the performance of TCP Reno in presence of various burst assembly algorithms at OBS ingress nodes. The authors have shown that TCP with medium to slow speed suffers from significant performance degradation due to the fact that the assembly delay becomes very large with respect to the arrival of each TCP segment. On the other hand, the authors have shown that fast TCP flows are less influenced by the burstificaiton delay.

The authors in [24] proposed an adaptive assembly-period (AAP) algorithm and investigated the impact of some configuration parameters, such as burst-assembly time, burst size, and threshold on adaptive assembly queue length, on the performance of both TCP and UDP traffic over OBS networks. The authors showed that the adaptive assembly algorithm supports achieving the best TCP performance among all the investigated assembly schemes.

The study in [32] has taken TCP Reno flavour as a dominant TCP, where various burst assembly delays and burst sizes have been examined and simulated. The study concluded that TCP Reno has failed to deal with burst losses where each burst contains a large number of TCP segments assembled from a single TCP source. The study in [32] contributed with a

TCP over OBS simulation package for *NS-2* which has been later extend by other research institutions [40][48][49][50].

- **Burst Contention Loss Recovery**

As one of the link-layer solutions, BCR schemes can be employed in OBS domain in order to reduce random burst loss, thereby improving the transmission reliability of OBS networks. In many cases, the reported BCR schemes have successfully hided burst-loss events from the upper TCP senders at the expense of introducing extra transmission delay for the bursts that experience contention. Among the three approaches, FDLs facilitates the BCR by providing very limited buffering time for each contended burst. In general, 1 *ms* of buffering time requires a fiber with a length of 200 *km*. Thus, the resultant delay could be limited as well. On the other hand, burst retransmission and deflection may introduce a significant amount of delay. Therefore, in this survey study, we will focus on both burst retransmission and deflection.

With burst retransmission, the OBS edge node stores a copy of the launched bursts for possible retransmission. As the control packet traverses through the core nodes, the intermediate node that fails to reserve the resource will send an explicit notification to the OBS edge in order to report the reservation failure. Upon receiving the notification, the OBS edge retransmits a duplicate of the requested contending burst preceded by the corresponding control packet. The retransmitted burst will certainly experience an extra retransmission delay, which is the time elapsed between the initial control packet transmission and the last notification receipt for the corresponding burst.

In the case that the network is lightly loaded, the retransmission scheme has a good chance of successfully delivering the contending bursts without involving the TCP retransmission mechanisms. The studies in [17] and [18] have shown that the retransmission scheme can significantly reduce the burst loss probability compared with that using a barebone OBS network, especially at low traffic load. The authors have also shown that retransmitting lost bursts from ingress nodes can avoid TCP false congestion detection. However, when the network is heavily loaded, the retransmitted bursts may still get blocked and finally lead to TO at the TCP senders. In this case, the retransmission persistence (or the maximum number of retransmissions for a single burst) and the time threshold at the OBS edge nodes before stopping the retransmission are important issues subject to further research efforts.

With burst deflection, a data burst is routed through its primary path in case there is no burst contention. In the event of contention at the core node, the burst will be dynamically rerouted and directed to an alternative path segment starting at the core node where the burst encounters a contention. Since the primary path is usually the shortest path, the data bursts following the alternative path segment results in a longer propagation delay [19][20]. The study in [18] has investigated burst retransmission along with deflection routing, which showed that the deflection scheme can significantly reduce burst-loss probability at low traffic loads. The same as the case in burst retransmission, the extra delay induced by the deflection will cause some vicious effects on the delay-based TCP. In the case where the network is heavily congested, the deflection may worsen the situation by consuming more resources, which leads to a fatal impairment to the TCP throughput.

- **TCP with Burst Acknowledgement**

In [29], a TCP throughput analytical model is introduced by considering the burst-acknowledgment mechanism implemented in the OBS domain. The authors proposed an error recovery mechanism for electronic buffering at the edge nodes in order to improve the TCP throughput at the expense of taking extra memory space at the edge node for buffering a copy of all bursts within a certain time window. Once the amount of additional memory at the edge nodes becomes very large, the proposed scheme could be subject to a problem in practical implementation. Furthermore, introducing burst delivery acknowledgement violates the fundamental OBS burst transmission semantic. 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 B_{dp}. The required extra buffering space is $c \times (RTT_{burst} + B_{dp} \times (RTT_{burst} + k \times M))$, where RTT_{burst} includes the burst assembly time and the burst offset time. The scheme cannot prevent the TCP senders from receiving TO indications, where the *cwnd* could be decreased in response to the burst retransmission delay.

- **TCP Decoupling**

In [30], a modified TCP decoupling approach is introduced. This approach monitors the burst-contention probability at the OBS network bottleneck link by taking the advantage of the TCP self-clocking property, where the burst sending rate is controlled through the arrival time of TCP decoupling management packets. In this scheme, a virtual circuit (VC) is set up for each source to destination edge node pairs in OBS domain. The VC is controlled through the TCP congestion control located at the OBS edges such that the sending rate never exceeds the link capacity. The OBS edge node uses TCP *ack* packets to control the timing of the burst sending. Considering the simulation parameters provided in [30], the authors have demonstrated an improvement of TCP throughput by avoiding unnecessary burst losses, where the overall link utilization is increased from 50% to 62% and the packet dropping probability is 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 (through TCP agents) at the OBS network.

- **Retransmission-Count Based Dropping Policy (RCDP)**

In [33] and [34], a dropping policy, called Retransmission-Count based dropping policy, is introduced, which aims to improve TCP throughput. The basic idea of the dropping policy works at the OBS edge node by taking the number of burst retransmissions attempts into consideration, where the bursts that have been less retransmitted are dropped. It is known that the more frequently the burst retransmission takes place in the OBS domain, the less time remains for the TCP

timeout to be triggered. The authors proposed to add a retransmission count (*RC*) field in the burst 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 the bursts with lower *RC* values. The bursts with larger *RC* values are subject to higher priority of successful retransmission since their assembled TCP packets have already experienced relatively longer delay. Once a burst is dropped, the corresponding NAK along with a copy of the *RC* value is sent back to the OBS edge node, which sets the *RC* filed for the dropped burst to (*RC_in_NAK+1*) and retransmits the burst. The number of retransmission attempts follows a predefined retransmission policy. This study aims to increase the transmission chances of the TCP packets which have experienced the longest time in the OBS domain. However, the authors have not address the TCP fairness factor in the proposed dropping policy. Also, the preemption of reserved resources by the bursts with a larger *RC* values may lead to negative influences on the dropped flows which may have already launched the corresponding data burst.

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

B. Solutions with Explicit Notifications

- **Burst TCP (BTCP)**

The study in [27] investigated TCP false timeout detection due to random burst-contention loss under wide range of traffic loads. Two solutions were proposed that use explicit notification. The first approach of [27], called BTCP with burst acknowledgement (Back), each TCP packet is acknowledged by a TCP agent located at OBS edge nodes. This approach can effectively prevent TCP from detecting false TO; however, the end-to-end TCP semantics is violated since *ack*s reach TCP senders before the actual completion of packet delivery. The second proposed approach in [27], called BTCP with burst negative acknowledgement (Nack), 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 explicitly exchanged between the TCP senders and the OBS core nodes. In general, how to reduce the extra control overhead and the implementation complexity is a challenge when attempting to deploy the abovementioned solutions.

- **TCP with Burst Contention Loss Notifications**

The paper in [43] has addressed the issues in the design of various TCP flavours in the OBS environment. TCP Reno, New Reno, Sack, Duplicate Sack [44], Fack, Forward Retransmission Timeout Recovery (F-RTO) [45], Eifel [46], delayed congestion response (DCR) [47], BTCP [27], and BAIMD [40] have been examined in terms of throughput performance under a wide range of burst drop probabilities. Table I summarizes the experiment results. The measured

performance evaluation of both DCR and F-RTO are based on TCP Sack.

Schemes	Suspicious TOs	Packet reordering	Multiple packet losses in a round
Reno (Eifel)	√		
New Reno (Eifel)	√		
Sack (Eifel)	√		√
DSack		√	√
Fack			√
DCR	√	√	√
F-RTO		√	√
BTCP	√	√	√
BAIMD	√	√	√

Table I: *TCP implementation performance over OBS*.

The simulation results have shown that the above TCP flavours have failed to well maintain a good throughput level in the presence of burst losses which contains multiple segments from a single TCP flow. Therefore, the Explicit Burst Contention-Loss Notification (BCL)-based TCP is introduced. The scheme aims to solve the false congestion detection problem in TCP over OBS networks and avoid the unnecessary *cwnd* reduction by adjusting TCP *cwnd* based on the utilization and burst dropping information carried in the explicit notification messages. This scheme considered the first study that integrates the explicit notification platform with the GAIMD framework over OBS networks.

In this paper the authors proposed a novel mechanism for detecting network congestion in bufferless OBS networks. In OBS networks, congestion occurs at edge nodes when receiving packets in a much higher arrival rate (*i.e.*, bursty impulse) than that the edge node is capable of dealing with. This causes the edge node to drop bursts due to buffer overflow. The authors refer the congestion at the network edge node as *edge congestion*. In addition to the edge congestion, the *path congestion* is defined, which leads to congestion in the network core nodes.

The authors proposed two possible approaches for detecting congestion at the network core (*i.e.*, path). The first approach is to delegate the congestion detection process to the core nodes. The edge nods receives explicit signals from the core switches indicating link congestion. Similar approach is proposed in BTCP with BNack [27]. It is notable that this approach may not be very practical since it increases the signaling and computation overhead at the core nodes. Therefore, the authors introduce a new mechanism for detecting congestion status along an OBS route (path) through the statistics gather at the corresponding edge node. This approach does not introduce any additional signaling effort at the core nodes. The path congestion is measured on how congested the route in the OBS domain is, which will be taken as an important index for the upper-layer TCP senders using the route to adjust their congestion windows.

In this scheme, each OBS ingress edge node maintains long-term and sort-term statistics for each route initiated at it. Whenever a burst-loss occurs, the ingress node determines if the route is in congestion by correlating the long-term and short-term statistics. In specific, let the parameter M be the number of transmitted burst along the OBS route, which is used to obtain the long-term statistics. Note that M should be sufficiently large in order to fully represent the intrinsic characteristics of the network topology, routing policy, and traffic pattern, *etc*. The outcome of the M burst deliveries is kept as a vector with each entry 0 or 1, which represents burst-drop event or successful delivery, respectively. Let the parameter N be the number of transmitted burst for the short-term burst drop rate, which is generally small.

The main idea of the proposed scheme is to position the average short-term burst drop rate (denoted as avg_b_N) in the spectrum of long-term burst drop rate. To achieve this, the outcomes of the M burst deliveries are divided into $\left\lfloor \dfrac{M}{N} \right\rfloor$ segments each contains the outcomes of consecutive N burst deliveries. Thus, it is possible to obtain a vector denoted as θ of a size $1 \times \left\lfloor \dfrac{M}{N} \right\rfloor$, where each entry, denoted as θ_i for $i = 1$ to $\left\lfloor \dfrac{M}{N} \right\rfloor$, stores the burst drop rate of the *i-th* small segment of burst deliveries. With this vector, it is possible to obtain the average burst drop rate in the M burst deliveries (denoted as avg_b_M) and the variance of the vector (denoted as var_b_M). In case avg_b_N is much larger than avg_b_M, it is highly possible that the route in the OBS core is subject to random burst contention, and the corresponding TCP senders should not take it serious in the response of *cwnd* adjustment. On the other hand, with comparable avg_b_N and avg_b_M, we can expect that the current burst drop should be more likely an indication of congestion along the route in the OBS domain.

In specific, to quantify the relationship between avg_b_N and avg_b_M, the authors assume that the burst-drop rate of each entry in θ can be positioned in histogram distribution. Thus, it will be easy to position avg_b_N in the spectrum formed by the vector θ. A policy-based is created to define the relationship between avg_b_N and the spectrum formed by θ in order to define the confidence with which a burst-loss event is due to network congestion is defined. A BCL is sent to the corresponding TCP senders if the burst loss is judged as due to contention at low traffic load. Thus, TCP-BCL has the TCP senders take every segment loss event as due to congestion when a BCL is not received. Figure 5 illustrates the proposed TCP-BCL congestion-control scheme. The proposed path congestion detection distinguishes TCP-BCL scheme from the BTCP in [27] since TCP-BCL only signals the dropped bursts at low link utilization, while BTCP (Back/BNack) reports the loss of every burst loss from core nodes. Hence, the number of notifications in TCP-BCL will be much less than that in BTCP. Furthermore, this design can significantly reduce the intra-domain signaling overhead as the edge node is the one responsible for detecting network congestion. This comes with the expense of introducing extra computation and signaling performed at the edge nodes.
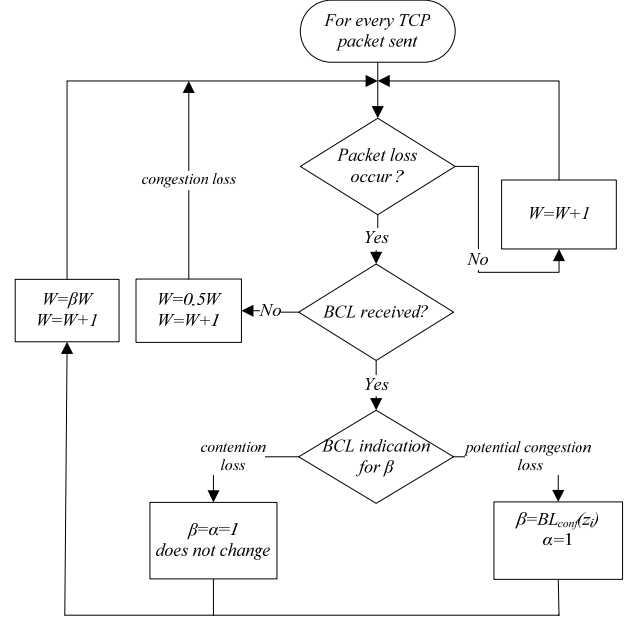
For every TCP packet sent

Packet loss occur ?

congestion loss

$W=W+1$

No

Yes

BCL received?

$W=\beta W$
$W=W+1$

$W=0.5W$
$W=W+1$

No

Yes

BCL indication for β

contention loss

potential congestion loss

$\beta=\alpha=1$
does not change

$\beta=BL_{con}(z_i)$
$\alpha=1$

Figure 5. *Flowchart of TCP-BCL over OBS*

C. Solutions without Explicit Notifications

- **Burst TCP with Burst Length Estimation**

The study in [27] proposed solving false TCP TOs using a burst length estimation approach. This approach, 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 denoted as *burst_wd* is also maintained. When the TCP sender detects TO, it first compares its *cwnd* with the *burst_wd*. If the $cwnd \le burst\_wd$ and $burst\_wd > 3$, then the TCP sender considers this TO as a false TO. It half the *cwnd* and perform fast retransmission for the missing segments. In case $cwnd > burst\_wd$ or $burst\_wd \le 3$, the TCP sender considers this TO event as a true TO and initiate normal TCP retransmission procedure. This approach does not require any coordination or explicit information exchange between the TCP senders and the OBS network. However, the accuracy of the estimated *burst_wd* remains an open research challenge. It is possible that a true TO can be taken as a false TO, which endanger the network's stability. Furthermore, this approach can not distinguish the loss of multiple packets in a congested IP-access network from the loss of multiple packets in a burst loss caused by random burst contention loss.

- **Burst AIMD (BAIMD)**

The BAIMD [40] scheme is based on the framework of Generalized Additive Increase Multiplicative Decrease (GAIMD) [41][42] for *cwnd* adjustment. Two parameters are defined: α and β, which serve as the additive incremental and the reduction ratio for the *cwnd* at each TCP sender. Unlike conventional TCP, where α and β are constantly set to be 1 and 0.5, BAIMD dynamically determines the two parameters

in such a way that the *cwnd* is increased by α segments for each acknowledged packet in a round and is decreases by β ($0.5 < \beta < 1$) as for any packet-loss event. It is clear that BAIMD is more general than AIMD with much better design granularity and flexibility.

With BAIMD, the sender is not explicitly notified about the used burst assembly mechanism and the reasons of the burst losses. Each sender treats a packet-loss event as a congestion loss. Obviously, this could lead to an overestimation of the network congestion by irrelevantly cutting the *cwnd* by a half for every TCP segment drop. To compensate this overestimation, BAIMD senders use β larger than 0.5. As such, the summarized effect of the burst-drop event is estimated when both the values of α and β are dynamically determined in the BAIMD senders using burst-level states (*i.e.*, the estimated traffic load in OBS network). 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 be $0.5 < \beta < 1$. 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 being simple where 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 signaling at the TCP senders and OBS edge nodes. Most notably, BAIMD senders use 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 substantially increases. On the other hand, if the 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 6 illustrates the functionality of the BAIMD scheme.
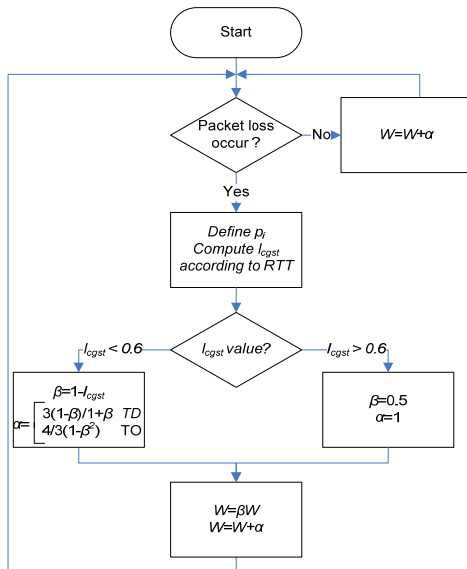

Figure 6. *Flowchart of the BAIMD congestion control scheme*

BAIMD estimates the multiplicative factor β through estimating the traffic load at the OBS layer. BAIMD defines certain threshold-load value denoted by 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 used threshold is $l_{cgst} \geq 0.6$), the BAIMD senders behave similar to conventional TCP senders with $\beta = 0.5$. Otherwise, β follows $\beta = 1 - l_{cgst}$ where, $0 < l_{cgst} < 0.6$. Once β is computed, BAIMD uses the GAIMD congestion control mechanism to obtained the sending rate α as follows [76][77][78]. In the presence of TD loss, $\alpha = 3(1-\beta)/(1+\beta)$, while in the case of TO loss, $\alpha = \frac{4}{3}(1-\beta^2)$.

- **Statistical AIMD (SAIMD)**

In [48], the authors introduced a new 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 taken in the developed policy for TCP congestion window adjustment.

The SAIMD scheme adopts the framework of BAIMD to enhance the responsiveness of TCP upon any burst loss event that is not caused by congestion. In SAIMD, when a burst consisting of many TCP segments from single or multiple TCP senders is lost, the corresponding TCP senders are notified of the packet loss through receiving either TDed acks or TO. In either case, instead of halving the *cwnd* or even throttling to the slow start stage, TCP senders reduce their *cwnd* by the multiplicative factor β. The factor β is dynamically determined by positioning the short-term RTT statistics in the spectrum of long-term historical RTTs. Here, the "statistics" refers to mean, standard deviation, and correlation function in this study, and will be further detailed as follows.

Two parameters in the proposed scheme, denoted as M and N, were introduced. The parameter M is the number of consecutive RTTs measured for the long-term statistics. M should be sufficiently large such that the derived statistics (*i.e.*, the mean and standard deviation) can fully 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 how likely the packet loss is due to network congestion or due to random burst contention loss at a light-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 since the high utilization of network resources

remains only a short time period in the N RTTs. A larger avg_RTT_N can be considered that a packet loss event is more likely due to network congestion rather than random burst contention.
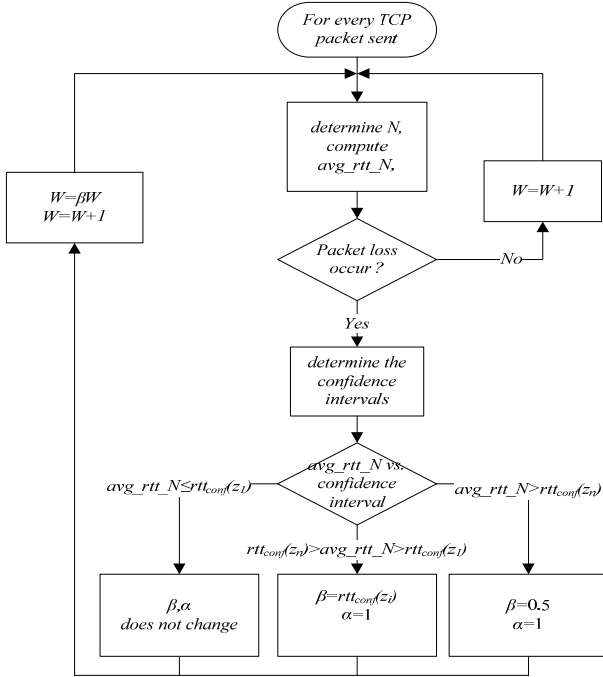

Figure 7. *Flowchart of the SAIMD congestion control scheme*

The relationship between avg_RTT_M and avg_RTT_N is based on the following observations: (1) in TCP over OBS networks, packet losses can be caused by random burst contention in OBS core networks or network congestion along the route of IP access networks and OBS core networks. The difference between random burst contention and network congestion is that network congestion suffers from high resource utilization for a longer period; (2) in the high resource utilization state, the RTT of each packet delivery will be much higher than that in the low-utilization state. This is due to the fact that high-utilization will cause longer queuing delay in IP access networks. Also, in an OBS core network with contention resolution schemes, such as burst retransmission [17][18] and deflection [19][20] schemes, bursts will have a high probability of being retransmitted or deflected, which results in a longer average burst delay in the OBS network.

Since the N RTTs are expected to provide sufficient information about the short-term network status when a packet is lost, the scheme uses the autocorrelation for selecting a proper value of N. If N is chosen too small or too larger, the short-term network status may not be accurately represented. The scheme defines an autocorrelation threshold value (γ) that determines N. In order to well-represent the short-term network status, the N RTTs should have a strong correlation with each other. Hence, the value of γ should be close to 1. In SAIMD, γ is taken as 90%. Figure 7 shows the congestion control of the proposed SAIMD scheme.

Compared with the conventional AIMD based TCP scheme, the SAIMD causes additional overhead for maintaining the M RTTs along with the efforts in computing the autocorrelation and confidence intervals for the N RTTs. The cost is nonetheless a trade-off with the long convergence time in recovery from slow-start caused by false congestion detection. This is considered with essential importance for those high-bandwidth TCP flows which may take hours or days to recover from a slow-start. 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 a constant regardless of M and N. In addition, the proposed SAIMD scheme can mainly be used for long and high-bandwidth TCP flows instead of short TCP such as HTTP web services; thus, the resultant additional overhead to the whole network is expected to be trivial.

- **Threshold–based TCP Vegas**

In [49][50] the authors analyzed the performance of the delay-based TCP (*i.e.*, Vegas) over OBS with burst retransmission and deflection. The study has shown several issues observed when the conventional TCP Vegas congestion control mechanism is adopted in OBS networks. When a fixed source-routing strategy is used, the packet delay experienced in the OBS domain is primarily the sum of burst assembly delay and link propagation delay, which do not vary when the traffic load changes in OBS networks. Hence, the conventional TCP Vegas cannot effectively detect network congestion in OBS domain. Furthermore, if all packets in the congestion window of TCP Vegas are assembled into a single burst, TCP Vegas may suffer from the false congestion detection problem, which fatally impairs the TCP throughput due to unnecessary TO retransmissions followed by slow start at TCP Vegas senders.

When TCP Vegas runs over OBS networks with burst retransmission or deflection scheme, false congestion detection problem can be mitigated since both schemes incur extra delay for bursts that are retransmitted. Hence, TCP Vegas will detect the increases in RTTs for packets in bursts that are retransmitted, which may result in TCP Vegas reducing *cwnd*, leading to lower TCP throughput. If the increases in RTTs are caused by burst retransmission in a lightly-loaded OBS network, TCP Vegas should not reduce its *cwnd*. Hence, a modified TCP Vegas which able to tell whether the increase in RTTs is due to network congestion or due to retransmission in lightly-loaded OBS networks is introduced and evaluated in [49][50].

Based on the above observations, the authors proposed a threshold-based TCP Vegas scheme by introducing a new parameter, T, referred to as the threshold, to assist TCP Vegas to distinguish between network congestion and burst contentions at low traffic loads. In the proposed scheme, the TCP Vegas measures RTT for each packet sent and keeps track of the minimum measured RTTs of 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, TCP sender will stay with the *Actual* throughput calculated based on $MinRTT(i-1)$, even if the measured RTTs are increased. If the number of TCP packets whose RTTs are larger than their $MinRTT(i-1)$ exceeds the threshold T, it means that the network is congested. Hence the threshold-based Vegas recognizes the network congestion and calculates the *Actual* throughput as usual.

The fundamental object of the threshold-based TCP Vegas is to reduce the sensitivity of TCP Vegas to the increases of RTTs caused by burst retransmission in the OBS domain. Instead of changing *cwnd*, the sensitivity of TCP Vegas can also be reduced by decreasing α or increasing β. However, changing α and β makes TCP Vegas difficult to estimate the available bandwidth in the networks. Figure 8 summarizes the proposed threshold-based TCP Vegas.
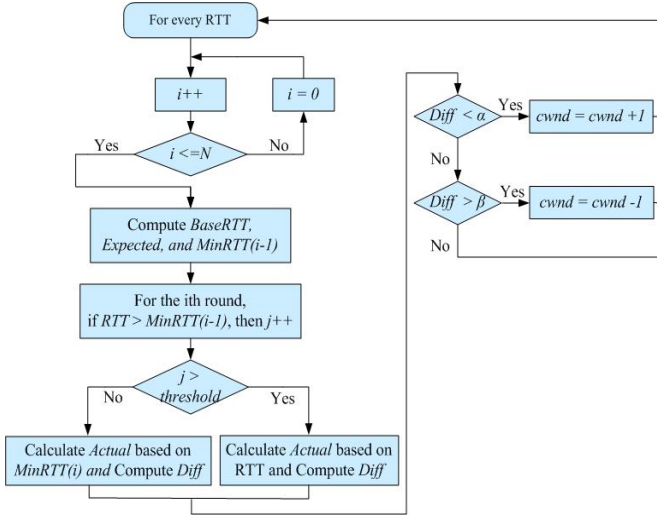

Figure 8. *The threshold-based TCP Vegas congestion control scheme.*

In the threshold-based TCP Vegas, the number of packets consecutively sent (denoted as N) and the threshold T should be chosen much larger than the number of packets from a single TCP Vegas connection that are assembled into a burst, such that TCP Vegas is able to detect the frequency of retransmission in the OBS domain based on a number of bursts. Usually, the packets from a TCP connection assembled in the same burst have the same measured RTT. By analyzing the variation pattern of packet RTTs, TCP Vegas can obtain the number of packets from a TCP Vegas connection that are assembled into a burst. Also, the relationship between the T and N affects the TCP performance. When T is chosen closer to N, there would be less remaining packets in the N consecutively packets to react to the detected congestion, which results in an ineffective response to the network congestion. Hence, we take $N = iT$, where $i > 1$. Defining

proper values of T and N are subject to many factors such as the number of TCP flows, the TCP flow speed, and the burst assembly thresholds. An optimal values of T and N are derived in [63] using the fixed-point feedback mechanism proposed in [31].

Table II summarizes the features of the surveyed schemes in this review paper noting that BCR stands for solutions that works on OBS with burst contention resolution schemes.

IV. OPEN RESEARCH PROBLEMS

Although TCP has been subject to extensive research efforts in the past decades, TCP over OBS is a relatively unexplored research area with limited number of studies that tackled some of the unique features in such a network scenarios. Through close analysis on the reported TCP enhancements listed in Table I and Table II, we observed that the most important characteristics and abilities that a congestion-control mechanism in TCP over OBS lie in the following three folds: (1) able to handle multiple TCP segment loss events in one round trip, (2) able to identify false TOs and burst losses under low traffic load situation, and (3) able to compensate the out-of-order delivery in the OBS domain. In this section, we identify the open issues in the area of TCP over OBS in the following subsections.

A. Integrating Link-Layer Solutions with TCP Performance

Although with less efforts for TCP enhancements over OBS, there have been extensive studies reported in the OBS network that aim to reduce burst dropping, provide QoS in burst transmission, and 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 [24], burst retransmission [17][18], burst deflection [19][20], and FDLs [21], have successfully contributed in reducing burst loss probability at the expense of introducing extra delay and design complexity. It is important to integrate link-layer solutions with any mechanism for TCP throughput and reliability enhancements before OBS can be practically deployed in the Internet.

We found that reducing the burst-dropping probability may not in all cases result in significant enhancement in the TCP throughput. For example, burst retransmission in OBS networks can greatly improve the TCP performance; 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 exist while tackling this problem is to properly define the TO threshold in 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 successively assembled packets in single burst. A TCP snooping mechanism similar to some proposed schemes in wireless networks such as ATCP [56], ELN [10], JTCP [57], TCP Casablanca [58], and TCP Peach [59], can be developed to evaluate the persistence of burst retransmission by estimating the RTT. This enhancement aims to compromise the risk of having a false TO with the gain by performing burst retransmission or burst deflection routing.

The research on TCP over OBS presented in BTCP [27], BAIMD [40], TCP-BCL [43], and SAIMD [48] have successfully resolved the vicious effects of false congestion detection due to the bufferless characteristic in the OBS networks. They improved the TCP sender reaction to each packet-loss event relevantly. However, each scheme suffers from some overhead as well, such as a high computation complexity and/or extra signaling in the OBS networks. Also, some of the above schemes may fail to overcome the problem of losing large number of packets assembled in a single burst. Furthermore, it is necessary to evaluate the convergence rate of TCP in presence of either a constant RTT in the barebone OBS or different values of RTT occurred due to the deployment of burst contention resolution schemes.

B. TCP Convergence Rates for Large Bandwidth-Delay Product Networks

There exists a significant lack of performance evaluation on the TCP modifications proposed for a network environment with a large bandwidth-delay product over the OBS networks. Fast TCP [8][9], Binary increase congestion control (BIC) TCP [51], Explicit Control Protocol (XCP) [52], TCP with Simple Available Bandwidth Utilization Library (SABUL) [53], High Speed (HSTCP) [54], and Scalable TCP (STCP) [55], are among the most famous promising solutions. XCP has shown stability and efficiency using ECN through extending ECN and Core Stateless Fair Queue (CSFQ), which make XCP aware of the per-flow state and buffer size status. XCP uses Multiplicative Increase Multiplicative Decrease (MIMD) to control the congestion window, which increases the transmission by Δ (*i.e.*, Δ = squares the bandwidth - the queue size). On the other hand, it uses AIMD to control the fairness. If $\Delta > 0$, then divide by Δ equally between the co-existing flows otherwise, divide Δ between flows proportionally to their current rates [52]. SABUL introduces a hybrid approach by merging the rate-based transmission via UDP and reliable retransmission via TCP, where a UDP channel is adopted for transmitting data at high rates, while a TCP channel is used to resend the missing data segments to ensure reliability [53]. 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 [54]. A simulation-based study for HSTCP over OBS is presented in [28]. Using small burst assembly delay and moderate burst dropping, the study shows that HSTCP throughput is severally affected. Scalable TCP (STCP) uses MIMD approach with sending rate 0.01 and multiplicative rate as 1/8. 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 TCP's *cwnd* by 0.01 for each acknowledged packet (not in the fast recovery stage), while cuts the *cwnd* by 0.875 for each packet-loss event [55].

There exist several proposed schemes that solve the TCP slow-convergence problem over high-speed networks. There is a need to evaluate the 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 SAUBL. It is known that the above schemes can archive faster convergence of TCP throughput in large-bandwidth high-delay networks by quickly enlarging their *cwnd*. However, with large *cwnd*, the number of *ack*s is significantly decreased. In the presence of random burst losses that contain a large number of *ack*s packets results into dramatic damage in the TCP *ack*-clocking and forces TCP to fall into false detection of network congestion. Thus, retransmit an unnecessary large number of packets. Furthermore, the *ack* losses are expected to affect large number of TCP senders since the burst can assemble many *ack*s packets due to their small size. Up to our knowledge, the effect of *ack* packet losses over OBS networks has not been addressed in the literature.

C. Performance Modeling for TCP over OBS Networks

The previously reported TCP over OBS performance modeling technique follows the packet-oriented approach [17][25][26][43][48][50]. These studies use Markov modulated Poisson arrival of the traffic [64]. In early 90's, the fluid modeling technique proposed in [60] has added a new dimension for modeling large number of TCP flows. However, the fluid modeling approach requires maintaining few strict assumptions, such as (1) having very large number of TCP flows, (2) with Poisson arrival of loss events, and (3) with strong correlation between losses in one RTT while being independent among the other RTTs. Regarding the first assumption, there is no sufficient evidence that the number of TCP flows is sufficiently high 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 partially be justified as that in the barebone OBS the RTT is more or less fixed. This is because the third assumption can only hold for the TCP flows which can emit the TCP segments of their entire *cwnd* while being assembled in one burst (e.g., fast flows [25]). We conclude that using the fluid model for evaluating TCP throughput performance over OBS requires significant improvements before being considered accurate.

The synchronization modeling approach proposed in [61] benefits from the *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 time to converge. The synchronization approach has been used for modeling Fast TCP and obtaining its stability by [61] and [62]. In order to obtain sufficient analysis for TCP performance while considering TCP stability, scalability and responsively, the synchronization modeling approach needs to capture the bufferless nature of OBS links (*i.e.*, fixed TCP RTTs), the burst aggregation factors, and the burst-loss distribution.

V. CONCLUSIONS

The research efforts of modifying and extending conventional TCP implementations have been extensively reported in the past years; however, relatively limited knowledge has been

gained in terms of the impacts on in the upper layer applications when OBS is adopted underneath in presence of burst dropping at different traffic loads. This article provides a comprehensive review and tutorial on the previously reported congestion control schemes for TCP over OBS networks. With the taxonomy identified in this review article, these schemes are classified into three categories, including the link-layer solutions, the congestion detection mechanisms with and without explicit notifications between the OBS edge nodes and the TCP senders. We have enumerated and discussed each category in details, aiming to provide a complete picture on the development of TCP enhancements over the OBS networks. We have also presented some future research directions for TCP enhancements in presence of various OBS characteristics, including the scenarios of large bandwidth-delay products, the schemes for protecting the TCP *ack* packets, and the extensions and elaborations of the existing modeling techniques (*e.g.*, fluid and synchronization modeling), in order to capture the burst transmission characteristics for TCP over OBS networks.

Schemes	Solution Category	OBS Type	Devices Involved			Explicit Notifications	Problem Addressed		
			TCP source	OBS edge	OBS core		Random burst Loss	Slow Convergence	Packet Reordering
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 Signaling	barebone/BCR	√						
BTCP with Burst ack	Signaling	barebone/BCR	√	√		√	√		
BTCP with burst Nack	Signaling	barebone/BCR	√	√	√	√	√		
Burst AIMD	without Signaling	barebone/BCR	√	√	√		√	√	
TCP-Burst Contention Loss	Signaling	barebone/BCR	√	√	√	√	√	√	√
Statistical AIMD	without Signaling	barebone/BCR	√				√	√	√
Threshold-based Vegas	without Signaling	BCR	√				√		√

Table II: *Overview of the surveyed OBS solutions for TCP.*

REFERENCES

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

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

[3] Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T. & Diot, C. 'Packet-Level Traffic Measurements from the Sprint IP Backbone'. IEEE Network, 17(6), November-December, pp. 6-16, 2003.

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

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

[6] 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.

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

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

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

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

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

[12] 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.

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

[14] 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.

[15] 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*, pp. 26-27, 1997.

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

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

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

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

[20] 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.

[21] 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.

[22] 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), vol. 5, pp. 2673 – 2677, 2002.

[23] 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", *IEEE International Conference on Communications* (ICC), vol. 3, pp.1543 – 1547, 2004.

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

[25] A. Detti and M. Listanti," Impact of segment agreegation on TCP Reno flows in optical burst switching networks, " in the proeeccedings of IEEE Infocom, 2002.

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

[27] X. Yu, C. Qiao, and Y. Liu, "TCP implementation and false time out detection in OBS networks," in the proceedings of IEEE Infocom, 2004.

[28] 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.

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

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

[31] C. Cameron, J. Choiy, S. Bilgrami, et al, "Fixed-Point performance analysis of TCP over optical burst switched networks", in the proceedings of ATNAC, 2004.

[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] L. Kim, S. Lee, and J. Song, "Dropping Policy for Improving the Throughput of TCP over Optical Burst-Switched Networks", *ICOIN*, pp. 409-418, 2006

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

[35] J. White, R. Tucker, and K. Long, "Merit-based scheduling algorithm for optical burst switching," *International conference on optical networks*, pp. 75-77, Korea, 2002.

[36] I. Ogushi, S. Arakawa, M. Murata, and K. Kitayama "Parallel reservation protocols for achieving fairness in optical burst switching," in the proceedings of *IEEE Workshop on High Performance Switching and Routing*, pp. 213-217, 2001.

[37] A. Kaheel, and H. Alnuweiri, "Batch scheduling algorithms for optical burst switching networks", *Proceedings of IFIP Networking,* pp. 90-101, 2005.

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

[39] J. Li, C. Qiao, J. Xu, and D. Xu, "Maximizing throughput for optical burst switching networks", *Proceedings of IEEE Infocomm*, pp. 1853 – 1863, 2004.

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

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

[42] 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.

[43] B. Shihada , and P-H. Ho, "A Novel TCP with Dynamic Burst-Contention Loss Notification over OBS Networks", Elservier Journal of Computer Networks, Vol. 52/2, pp. 461-471,2008

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

[45] 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.

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

[47] 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.

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

[49] B. Shihada, Q. Zhang, and P-H. Ho, "Threshold-based TCP Vegas over Optical Burst Switched Networks", 15th IEEE International Conference on Computer Communications and Networks (ICCCN'06), Virginia, USA, October 2006.

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

[51] L. Xu, K. Harfoush, and I. Rhee, "Binary incrase congestion control (BIC) for fast long-distance networks", IEEE infocom, Hong Kong, China, 2004.

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

[53] 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*, 2004S.

[54] Floyd, "High-Speed TCP for large congestion windows", RFC 3649, Experimental, December, 2003

[55] T. Kelly, "Scalable TCP: improving performance in highspeed wide area networks", ACM SIGCOMM, vol. 33, no. 2, pp. 83-91, 2003.

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

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

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

[59] 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.

[60] V. Misra, W.-B. Gong, and D. Towsley, "Fluid based analysis of a network AQM routes supporting TCP flows with an application to RED," Proc. ACM SIGCOMM, pp. 151-160, 2000.

[61] D. Wei, "Congestion control algorithms for high speed long distance TCP connections", Master thesis, Caltech, 2004. http://www.cs.caltech.edu/~weixl/research/msthesis.ps

[62] J. Wang, D. Wei, S. Low,"Modeling and Stability of Fast TCP", *IEEE Infocom*, pp. 938-948, 2005.

[63] B. Shihada, Q. Zhang, and P-H. Ho, "Performance Evaluation of Threshold-Based TCP Vegas over Optical Burst Switched Networks", submitted to *IEEE Journal on Selected Areas in Communications,* January 2007.

[64] T. Karagiannis, M. Molle, M. Faloutsos, A. Broido, "A Nonstationary Poisson View of Internet Traffic", *IEEE Infocom*, 2004.
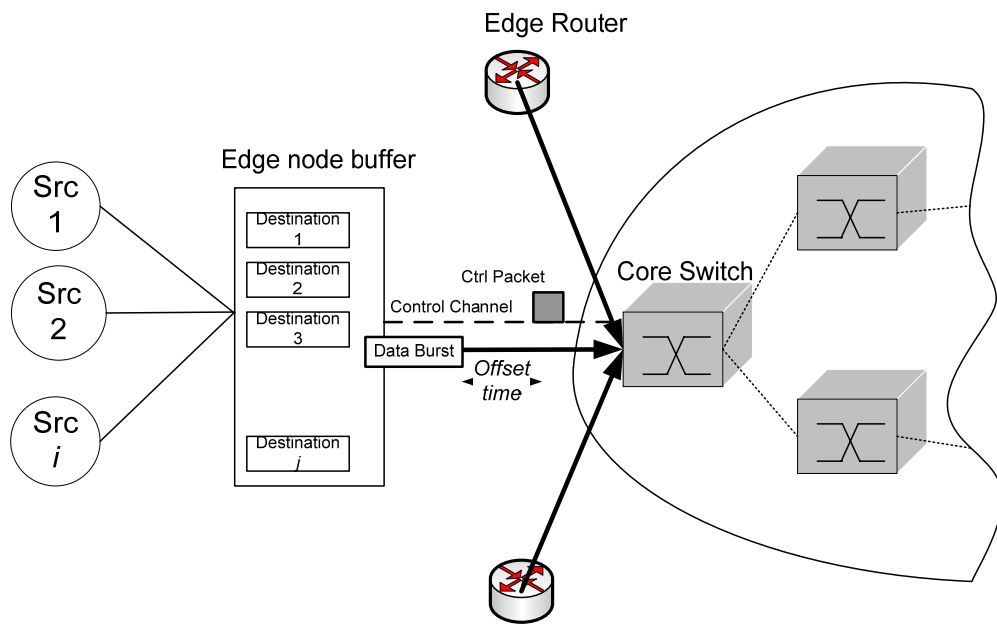
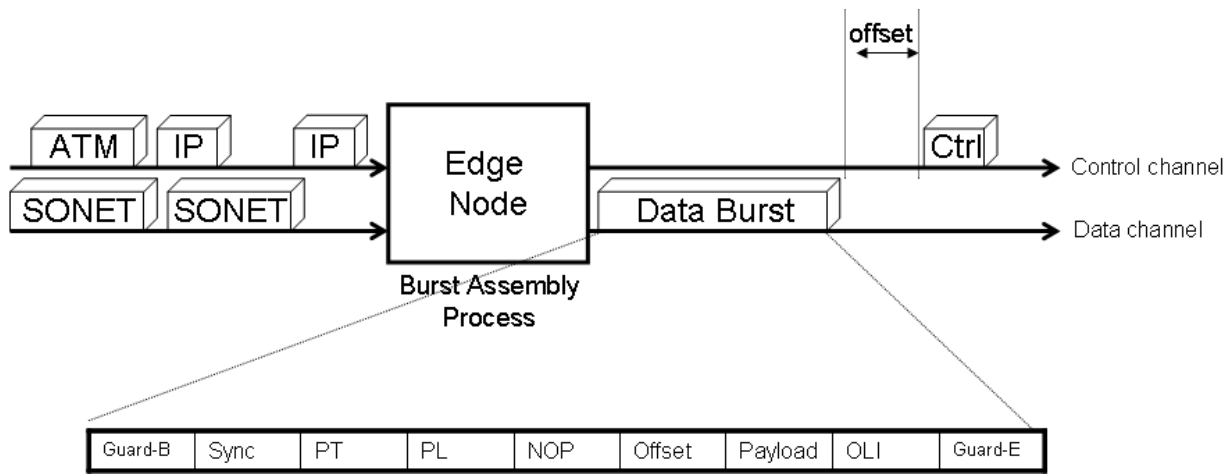Figure 1. An illustration of OBS networks
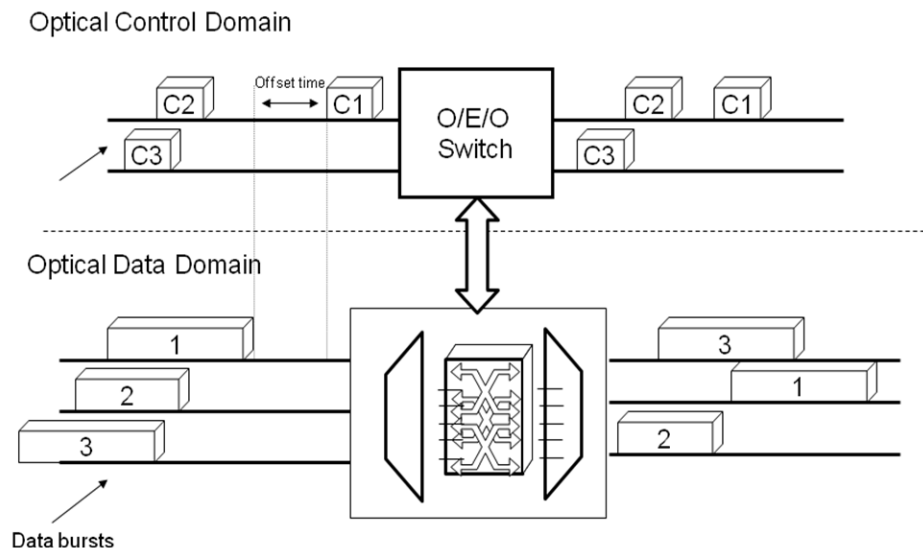
Figure 2. OBS edge node architecture

Figure 3. OBS core switch architecture

Solutions of TCP in OBS
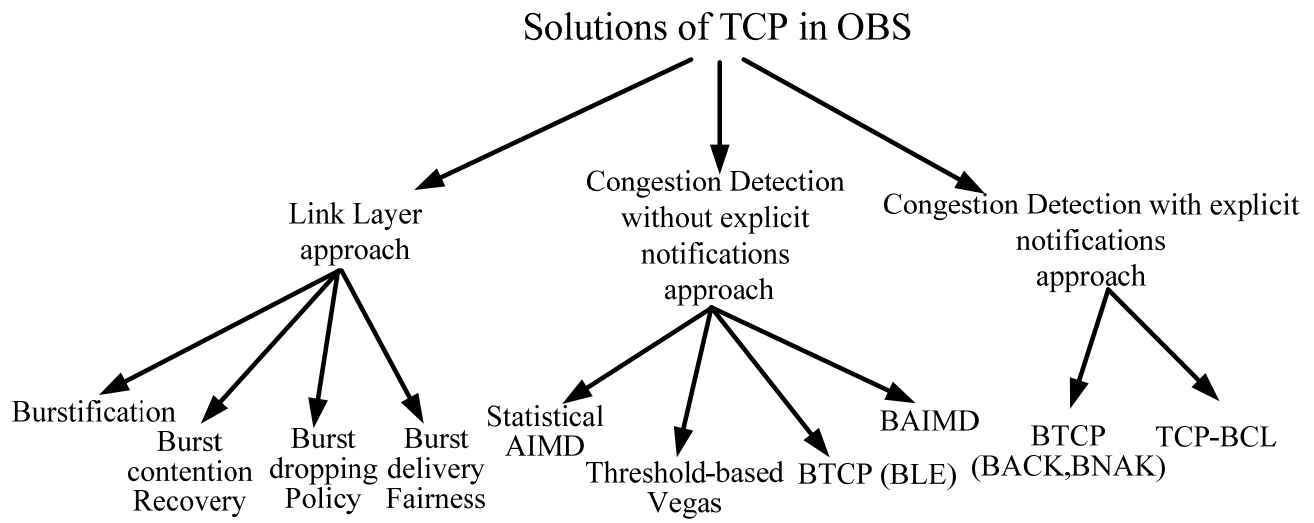
Link Layer approach

Congestion Detection without explicit notifications approach

Congestion Detection with explicit notifications approach

Burstification

Burst contention Recovery

Burst dropping Policy

Burst delivery Fairness

Statistical AIMD

Threshold-based Vegas

BAIMD

BTCP (BLE)

BTCP (BACK,BNAK)

TCP-BCL

Figure 4. The taxonomy of solutions for TCP in OBS networks
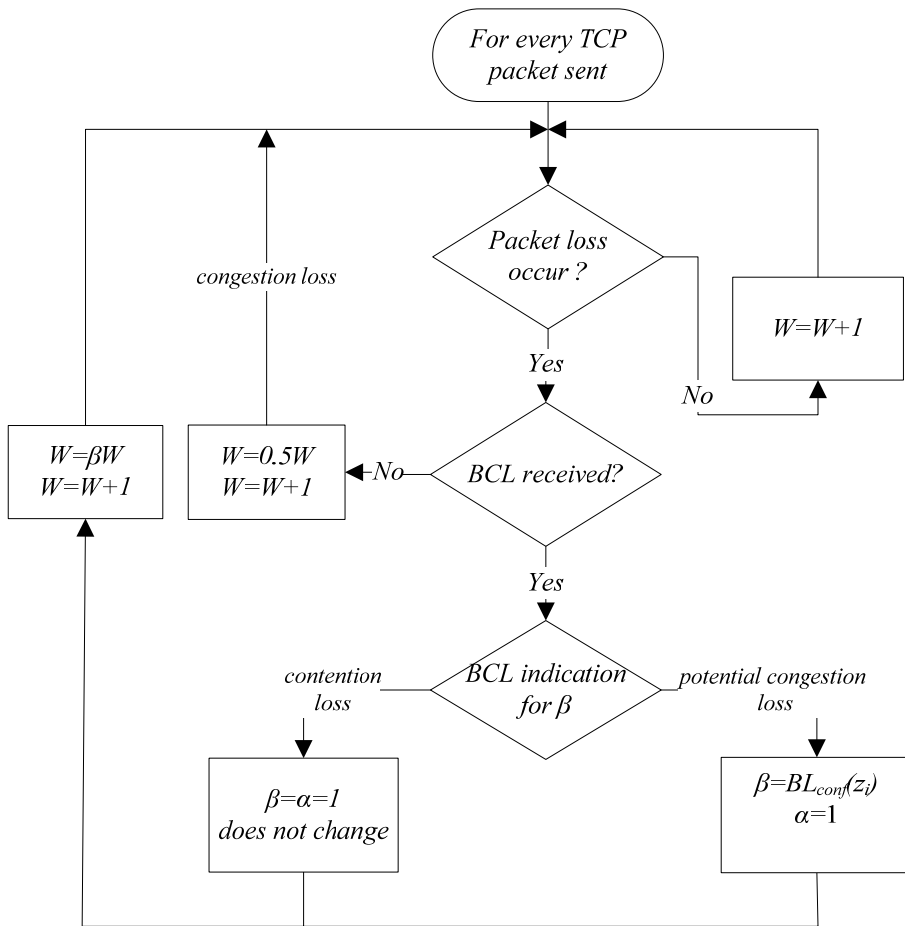
Figure 5. Flowchart of TCP-BCL over OBS
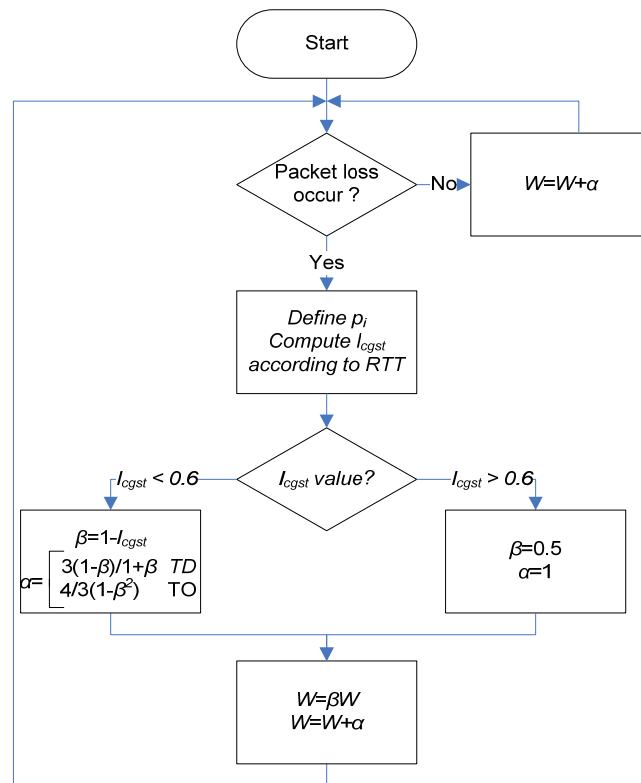
Figure 6. Flowchart of the BAIMD congestion control scheme

For every TCP packet sent

determine N, compute avg_rtt_N,

$W=\beta W$
$W=W+1$

$W=W+1$

Packet loss occur ?

No

Yes

determine the confidence intervals

avg_rtt_N vs. confidence interval

$avg\_rtt\_N \leq rtt_{conf}(z_1)$

$avg\_rtt\_N > rtt_{conf}(z_n)$

$rtt_{conf}(z_n) > avg\_rtt\_N > rtt_{conf}(z_1)$

β, α
does not change

$\beta = rtt_{conf}(z_i)$
$\alpha = 1$
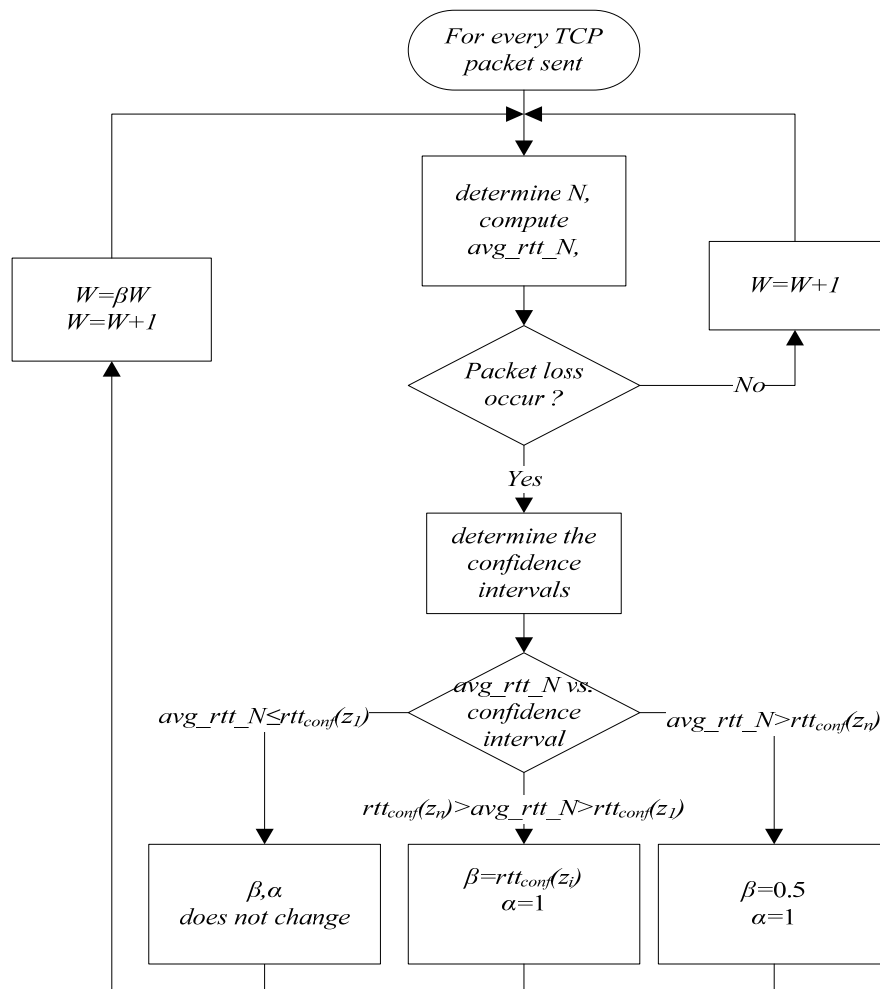
$\beta = 0.5$
$\alpha = 1$
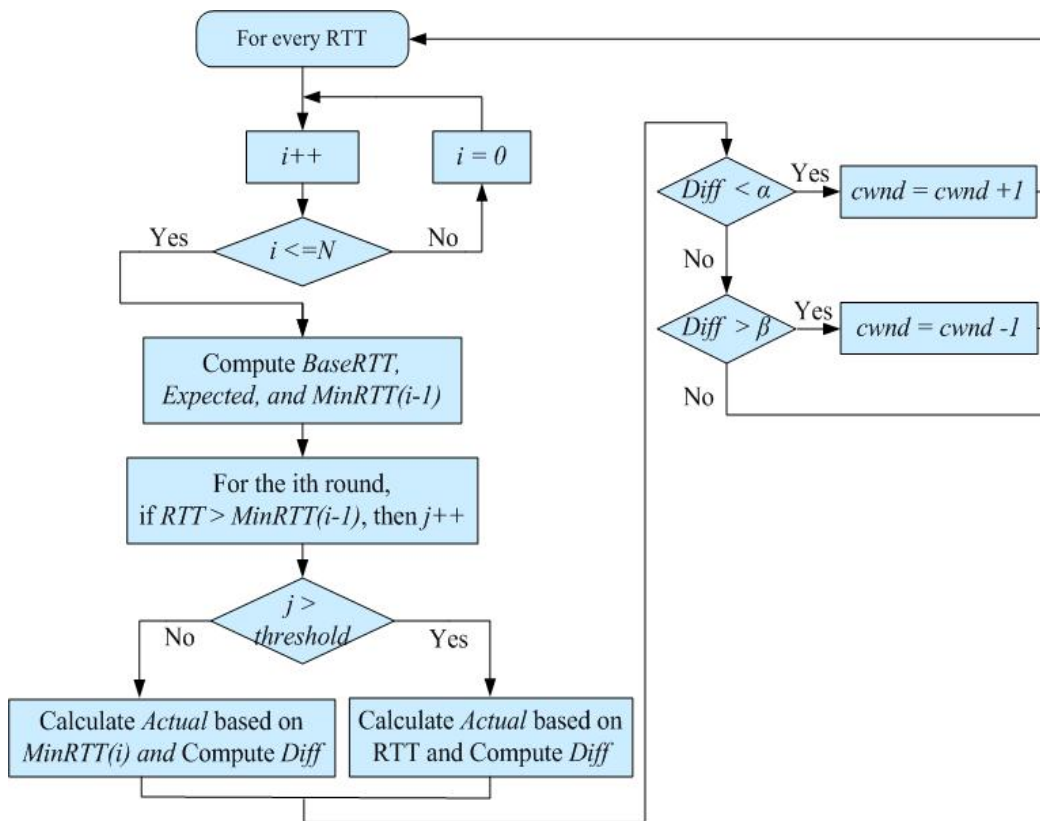
Figure 7. Flowchart of the SAIMD congestion control scheme

Figure 8. The threshold-based TCP Vegas congestion control scheme.