Noname manuscript No.
(will be inserted by the editor)

TCP over Optical Burst-Switched Networks with Controlled Burst Retransmission

Qiong Zhang, Neal Charbonneau, Vinod M. Vokkarane, and Jason P. Jue

Abstract For optical burst-switched (OBS) networks in which TCP is implemented at a higher layer, the loss of bursts can lead to serious degradation of TCP performance. Due to the bufferless nature of OBS, random burst losses may occur, even at low traffic loads. Consequently, these random burst losses may be mistakenly interpreted by the TCP layer as congestion in the network. The TCP sender will then trigger congestion control mechanisms, thereby reducing TCP throughput unnecessarily. In this paper, we introduce a controlled retransmission scheme in which the bursts lost due to contention in the OBS network are retransmitted at the OBS layer. The OBS retransmission scheme can reduce the burst loss probability in the OBS core network. Also, the OBS retransmission scheme can reduce the probability that the TCP layer falsely detects congestion, thereby improving the TCP throughput. We develop an analytical model for evaluating the burst loss probability in an OBS network that uses a retransmission scheme, and we also analyze TCP throughput when the OBS layer implements burst retransmission. We develop a simulation model to validate the analytical results. Simulation and analytical results show that an OBS layer with controlled burst retransmission provides up to two to three orders of magnitude improvement in TCP throughput over an OBS layer without burst retransmission. This significant improvement is primarily because the TCP layer triggers fewer time-outs when the OBS retransmission scheme is used. [1]

Address(es) of author(s) should be given

[1] This work was supported in part by the National Science Foundation (NSF) under Grant ANI-01-33899 and CNS-0626798. Portions of this paper have appeared in IEEE/CreateNet BroadNets 2005 and IEEE GLOBECOM 2005.

Qiong Zhang is with Fujitsu Labs of America Inc., Richardson, TX. Neal Charbonneau and Vinod M. Vokkarane are with the Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA. Jason P. Jue is with the Department of Computer Science, University of Texas at Dallas, Richardson, TX.

Keywords: DWDM, OBS, Retransmission, and TCP.

1 Introduction

Optical burst switching (OBS) [1] is a promising switching technology that efficiently utilizes the raw bandwidth provided by dense wavelength division multiplexing (DWDM), and at the same time, avoids the need for optical buffering while handling bursty traffic. OBS is expected to support the dramatically increasing bandwidth demands of the Internet backbone. In an OBS network, a data burst consisting of multiple IP packets is switched through the network all-optically. A *burst header packet* (BHP) is transmitted ahead of the burst in order to reserve the data channel and configure the core switches along the burst's route. In the *just enough time* (JET) signaling scheme [2], the burst transmission follows an out-of-band BHP after a predetermined offset time. The offset time allows the BHP to be processed before the burst arrives at the intermediate nodes; thus, the data burst does not need to be delayed at the intermediate nodes. The BHP also specifies the duration of the burst so that each node knows when the resources being used by the burst will be released. Since most OBS signaling and reservation protocols, such as JET [2] and JIT [3,4], are implemented in an one-way unacknowledged manner, burst contentions may occur in the OBS network. If a burst reservation fails due to contention in the core, then the burst will be dropped, and higher layers (such as TCP) will need to handle the retransmission of lost data at a later time.

Due to the bufferless nature of OBS core network and one-way based signaling, OBS network will suffer from random burst losses even at low traffic loads. One problem that arises when TCP traffic traverses over OBS networks is that the random burst loss may be falsely interpreted as network congestion by the TCP layer. This problem is referred to as *false congestion detection*. For example, if a burst that

contains all of the segments of a TCP sending window is dropped due to contention at a low traffic load, then the TCP sender times out, which leads to false congestion detection. This false congestion detection is referred to as a *false time out* (FTO) in [5]. When the TCP sender detects this false congestion, it will trigger the *slow start* congestion control mechanism, which will result in the TCP throughput being reduced. Another example is when a random burst loss triggers TCP fast retransmission for the case in which segments in a TCP sending window are assembled into multiple bursts. The burst loss will be interpreted as light network congestion and will trigger one or more TCP-layer fast retransmissions. This false congestion detection is referred to as *false fast retransmission (FFR)* in [6]. The amount of time that it takes for the TCP layer to recover the segments in the lost burst through fast retransmission is referred to as *fast retransmission period*. The false congestion detection problem becomes even more severe when TCP packets from many TCP senders are assembled into a single burst and the burst is dropped in the OBS network. In this case, many TCP flows will detect packet losses and reduce their sending rate, which leads to the network being under-utilized.

Many TCP-based applications, such as web (HTTP), email (SMTP), peer-to-peer file sharing [7,8], and grid computing [9], account for majority of data traffic in the Internet; thus understanding and improving the performance of TCP implementations over OBS networks is critical. Several works have evaluated TCP throughput over an OBS network [10–12]. However, these works assume a constant random burst loss probability in the OBS network, and do not take into account TCP false congestion detection. In [18], the authors propose the modification of TCP protocols in order to avoid the false congestion detection problem. The work in [5] proposes several cross-layer schemes for detecting FTOs in the TCP layer and reacting with a fast retransmission for each FTO detection. However, these schemes require either that the TCP sender is capable of estimating the maximum number of packets assembled into a burst, or that the OBS nodes are able to send the TCP packet information in a burst back to the TCP sender. The requirement that OBS nodes must be aware of TCP segments leads to the complication of both the TCP and OBS network implementations. Each node in the OBS core network must keep track of each individual TCP flow, as well as corresponding information, such as the sequence numbers of TCP packets and the TCP sender's IP address. Hence, the proposed methods may not be practical.

In this paper, we propose a controlled burst retransmission scheme for OBS networks. In the OBS retransmission scheme, when a burst incurs contention at a core node, the core node sends an *automatic retransmission request* (ARQ) back to the ingress node. Once the ingress node is notified of the burst contention, the ingress node retransmits a duplicate of the lost burst. The retransmission scheme may retransmit the burst multiple times until either the burst reaches the egress node, or the burst retransmission process exceeds a preset delay constraint. If a retransmission results in the burst exceeding its delay constraint, then the burst will simply be dropped. By setting a proper delay constraint (considering the edge node's buffer size), the OBS layer retransmission can recover most of the bursts that are lost due to random burst contentions under low loads. If a burst is dropped during retransmission, then it is likely that the OBS network is indeed congested. The TCP layer then correctly detects network congestion and acts accordingly.

The burst retransmission scheme has several advantages. First, the burst retransmission scheme can avoid TCP false congestion detection, while maintaining layer independence. There is no need to maintain information for each individual TCP flow in the OBS layer and no need to modify the upper TCP layer. Second, the burst retransmission scheme significantly reduces burst loss probability, especially at low network loads. The tradeoff of the burst retransmission scheme is that it requires electronic buffering of bursts at ingress nodes. If an arriving burst can not be stored due to lack of buffers, the burst will not be retransmitted. The successfully retransmitted bursts suffer from additional retransmission delay and can cause congestion in the network at high loads. Based on extensive simulations we observe that retransmission exhibits this behavior only at very high loads. In order to handle this issue, we can assign different priorities to original bursts (high priority) and retransmitted bursts (low priority).

The TCP over OBS network with burst retransmission has two levels of loss recovery. One level is TCP-layer packet loss recovery through fast retransmissions and time-out based retransmissions. The other level is through OBS-layer burst retransmissions. By setting a very high delay constraint, the OBS-layer retransmission scheme can potentially recover from all of the burst losses; however, if the delay constraint is too high, then the TCP layer may retransmit the lost packets before the retransmitted burst reaches the destination. We will determine how to choose an optimal delay constraint in order to minimize redundant packet retransmissions.

In this paper, we will evaluate and model the performance of the controlled burst retransmission scheme and investigate whether the burst retransmission scheme results in higher burst loss probability due to the additional load from the retransmitted bursts. The buffer size at ingress nodes is determined by the delay constraint, as well as the burst arrival rate. If an arriving burst cannot be stored due to lack of buffers, we assume that the burst will not be retransmitted. Hence, we will obtain the required buffer size in order to maintain a certain probability that an arriving burst can not be stored at an ingress node. We will also evaluate the TCP performance over an OBS network with OBS-layer burst retransmissions through analysis and simulation.

Researchers have proposed to use link-layer retransmission technique for TCP over wireless networks [13–16]. However, there is a major difference between the wireless link-layer retransmission and the OBS-layer retransmission. Contentions on wireless channels drop all contending packets, while contentions on OBS links drop only the burst that arrives latter and fails wavelength reservation. This difference significantly impacts the packet loss probability in the higher layer. Also, the typical data rates in optical networks are several orders of magnitude higher compared to typical data rates in wireless networks.

There are many contention resolution mechanisms that can also reduce random burst loss, thereby avoiding TCP false congestion detection. These mechanisms include fiber delay line buffering [17], wavelength conversion [19], segmentation [20], and deflection [21]. Wavelength conversion and fiber delay line buffering need expensive hardware, while segmentation has assembly overhead at edge nodes to create segments. Also, deflection suffers from potential loops and insufficient offset time [22]. Based on the existing literature, there is no single contention resolution mechanism that can clearly outperform the rest, since the performance of these techniques heavily depends on network characteristics, such as network topology, switch architecture, traffic conditions, and nodal degree. In this paper, we adopt the burst retransmission scheme since it is a simple and light-weight loss-recovery mechanism. Also, the proposed burst retransmission scheme can be used in conjunction with existing contention resolution mechanisms.

The paper is organized as follows. Section 2 provides background information on congestion control mechanisms for different TCP flavors. Section 3 discusses the controlled retransmission scheme in the OBS layer. In Section 4 presents the performance evaluation of TCP over an OBS network with OBS-layer burst retransmissions. Section 5 presents an analytical model for the burst loss probability for an OBS-layer burst retransmission scheme and an analysis of TCP throughput over an OBS network with OBS-layer burst retransmission. Section 6 presents numerical results from the analysis and simulation, and also compares the TCP performance over OBS networks without OBS-layer burst retransmission and with OBS-layer burst retransmission. Section 7 concludes the paper.

2 Background: TCP SACK and High Speed TCP

In this section, we describe TCP congestion control mechanisms implemented by two well-known loss-based flavors of TCP: SACK [25] and High Speed TCP [29].

TCP congestion control mechanisms include slow start, congestion avoidance, fast retransmission, and fast recovery. If a TCP segment is lost, there are two types of loss indications: *time out* (TO) and *triple duplicates* (TD). A TO loss is detected by a *retransmission time out* (RTO), when an acknowledgment for a segment is not received within a certain period of time. TCP interprets a TO loss as a loss due to heavy network congestion; hence, the TCP sender retransmits the lost segment and enters into a *slow start* phase. A TD loss is detected when a TCP sender receives three duplicate ACKs, which indicates that a packet is lost due to light network congestion; hence, the TCP sender enters into *fast retransmission* and *fast recovery* without waiting for RTO.

TCP SACK is one of the most widely deployed TCP flavors, while HS-TCP has been gaining a lot of attention for networks with large bandwidth-delay product (BDP).

TCP SACK is a conservative extension of TCP Reno. TCP Reno performs poorly when multiple packets are lost because its fast recovery assumes only a single packet loss. When there are multiple packets lost it may enter and exit fast recovery multiple times reducing the window drastically. SACK improves upon TCP Reno by using *selective acknowledgements*. An ACK contains a number of SACK blocks, where each SACK block reports a non-continuous set of packets that has been received and queued at the receiver side. After detecting a TD loss, the sender retransmits one lost segment and enters the fast recovery phase. The TCP sender selectively retransmits one or more lost segments that are reported by a SACK block for each partial ACK it receives. When an ACK acknowledges the highest sequence number sent when fast retransmission was triggered, TCP SACK exits the fast recovery phase and enters congestion avoidance. By giving the SACK information, the sender can avoid unnecessary delays and retransmissions as in Reno, resulting in improved throughput.

High Speed-TCP (HS-TCP) is a modification to TCP's window increase and decrease algorithm that allows it to run efficiently on networks with large BDP [29]. Standard TCP performs poorly on networks that require large congestion windows because of the slow growth during congestion avoidance and the large decrease in window size after a triple duplicate event. HS-TCP has two modes of operation, one is used when the packet loss rate is high, or, equivalently, when the congestion window is small, and the other is used when the congestion window is greater than a threshold value. While the congestion window is under this threshold value, Low_Window, HS-TCP behaves exactly like standard TCP; it increases the sender's window by one segment each RTT in congestion avoidance and reduces the window by half after a triple duplicate acknowledgement. When the window is greater than this threshold value, the increment and decrement amount become functions of the current window size. As the window gets larger, the window increment per RTT also gets larger while the decrement after a triple duplicate gets smaller. This modification allows HS-TCP to quickly recover after a loss in networks with large BDPs but still be fair to standard TCP in normal networks. This mod-

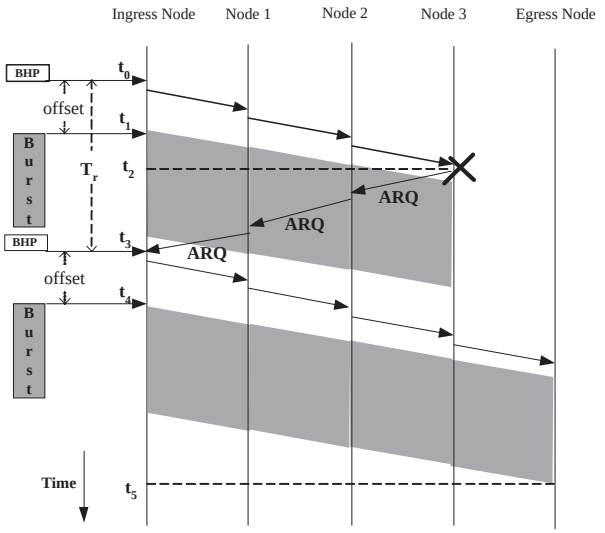


Fig. 1 OBS retransmission scheme.

ification can be made to any of the standard TCP flavors as it only modifies the window increase and decrease behavior. HS-TCP behaves the same as standard TCP during the slow start phase and after a timeout is experienced. In the paper we use TCP SACK and HS-TCP with SACK.

3 Controlled Retransmission in OBS Networks

In this section, we propose a burst retransmission scheme for OBS networks, and we also discuss design issues for the retransmission scheme. In the rest of the paper, the burst dropped due to contention will be referred to as the *contending burst*.

3.1 OBS Retransmission Scheme

The basic idea of burst retransmission is to allow contending bursts to be retransmitted in the OBS layer rather than having higher-level protocols, such as TCP, recover lost data. In this scheme, BHPs are sent out prior to data burst transmission in order to reserve resources. After an offset time, the burst is transmitted. The ingress node stores a copy of the transmitted burst for possible retransmissions. As the BHP traverses through the core nodes, if the channel reservation fails at a core node due to burst contention, the core node will send an ARQ on the control plane to the ingress node in order to report the reservation failure. Upon receiving an ARQ, the ingress node retransmits a duplicate of the requested burst preceded by a corresponding BHP.

We illustrate a retransmission scenario in Fig. 1. In this figure, the BHP is transmitted at time t_0, while the burst is duplicated and stored at the ingress node before being transmitted. The burst is transmitted at time t_1 after some offset time. At t_2, the burst reservation fails at Node 3, triggering Node 3 to send an ARQ back to the ingress node. The ingress node receives the ARQ at t_3, then sends a new BHP and retransmits a duplicate burst at t_4 after some offset time. Assuming the retransmission is successful, at t_5 the burst arrives at the egress node. A duplicate burst may be retransmitted multiple times until the burst successfully reaches the egress node.

We observe from Fig. 1 that the retransmission scheme results in an extra delay, T_r, referred to as *retransmission delay*. The retransmission delay is the time elapsed at the sender between the initial BHP transmission of a burst and the last ARQ receipt for the corresponding burst, i.e., $T_r = t_3 - t_0$. The retransmission delay can be bounded by a delay constraint, notated as δ. Once the ingress node receives an ARQ for a contending burst, the ingress node calculates T_r for the contending burst and decides if it is necessary to retransmit the burst. If $T_r \geq \delta$, the ingress node ignores the ARQ and does not retransmit the burst. After δ, the duplicate copy in the electronic buffer will be purged.

If the network is lightly loaded, the retransmission scheme has a good chance of successfully retransmitting contending bursts. Hence, the retransmission scheme improves the loss performance in an OBS network. Since the contending burst can be retransmitted and received at the egress node prior to TCP timeout, the retransmission scheme can avoid FTOs at the higher TCP layer. Also, the burst retransmission scheme can reduce the fast retransmission period if packets in a lost burst can successfully reach the destination through burst retransmission before the TCP layer recovers the lost segments through fast retransmission. If the network is heavily loaded, the retransmitted bursts have a lower probability of being successfully received. The ingress node can continue to attempt retransmission until the retransmission delay exceeds the delay constraint, in which case the burst is dropped and no longer retransmitted when a contention occurs. Hence, burst losses in an OBS network with controlled burst retransmission can potentially provide a more accurate indication of network congestion to the TCP layer.

Compared to an OBS network without burst retransmission, an OBS network with burst retransmissions will have a higher traffic load in the network, leading to higher burst contention probability. However, the burst is allowed to experience multiple contentions, which leads to a lower effective burst loss probability, especially at low network loads. In Section 6 we will investigate this issue.

3.2 Buffer Requirements and Burst Identity

In the OBS retransmission scheme, each ingress node must store a copy of each transmitted burst for possible retransmission, prior to transmitting the burst. Therefore, electronic

buffering at each ingress node is necessary. Since the retransmission scheme only reports burst contentions (negative acknowledgement) but not the successful receipt of the bursts, the ingress node does not know when to purge the buffer of unwanted bursts. Given the delay constraint, δ, the ingress node can purge the bursts that have been in the buffer for δ units of time.

If we assume that burst arrivals at the buffer have a Poisson distribution, that the duration of the burst staying in the retransmission buffer is δ, and that the buffer can store k bursts, we can model the retransmission buffer as a $M/G/k/k$ queuing system. Let P_b be the buffer blocking probability, or the probability that a burst could not be stored in the buffer, and let k be the number of bursts that the buffer needs to store in order to satisfy a given buffer blocking probability. Using the Erlang-B formula, we can obtain the relationship between the number of bursts to be stored and P_b, and can then estimate the required buffer size for a certain buffer blocking probability.

Another issue is that each ingress node must assign a unique identity to each new burst in order to identify the burst duplicate at the ingress node. Burst identification can be provided by a unique sequence number for each burst traversing between an ingress-egress pair. Each ARQ message contains the sequence number of the contending burst. Once the ingress node receives the ARQ message, the ingress node can identify which burst to retransmit.

The authors in [36] provide an alternative controlled burst retransmission framework. In addition to a delay constraint and upper bound on the maximum number of retransmissions per burst, they provide two more parameters, α_1 and α_2, used to control the probability that a burst will be retransmitted given an ARQ was received. They investigate suitable parameters to find a good tradeoff between the higher network loads (and hence higher contention probability) and lower effective loss. However, they do not analyze impact of burst retransmission on TCP traffic. In the following section we evaluate the performance of TCP over OBS with controlled burst retransmissions.

4 TCP over OBS with Controlled Retransmission

In an OBS network with burst retransmission, the value of the delay constraint, δ, is critical. The delay constraint not only determines the buffer capacity necessary at the ingress nodes, but also has an effect on the higher layers, such as TCP. In this section, we attempt to determine a reasonable delay constraint independent of the different TCP flows.

Fig. 2 shows a general scenario of a TCP flow over an OBS network. We denote the access network delay as T_a, the burst assembly and disassembly delay as T_b, and the propagation delay incurred in the OBS network as T_p. In the

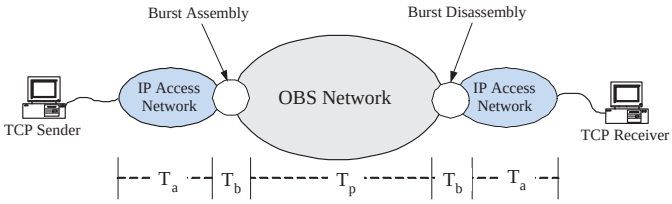


Fig. 2 TCP over OBS network.

TCP layer, we denote the *round trip time* as RTT, where $RTT = 2(T_p + 2T_b + 2T_a)$.

We consider two kinds of TCP flows, namely fast flow and medium flow [10]. For a fast flow, all the segments in a TCP source's sending window are assembled into a single outgoing burst. Thus, if the burst is dropped, even after OBS retransmission attempts, then the TCP source will time out. For a medium flow, the number of segments of a TCP source included in a burst is more than one and less than the sender's window size. Thus, if the burst is dropped, even after OBS retransmission attempts, then the TCP source will have a high probability of entering the fast recovery phase.

We first consider a TCP fast flow over an OBS network with burst retransmission. For a TCP fast flow, if a burst experiences contention and if the burst retransmission is successful, the TCP receiver will acknowledge all the packets contained in the burst. The OBS burst retransmission does result in additional retransmission delay; however, as long as the acknowledgments for the corresponding packets arrive at the TCP sender prior to RTO, slow start will not be triggered. Hence, the delay constraint δ can be set to $(RTO - RTT)$ such that the packets in the successfully retransmitted burst are acknowledged earlier than RTO. In TCP, the value of RTO is generally several times the RTT. If we assume $RTO = 2RTT$ when deciding the value of δ, then we have $\delta = RTT = 2T_p + 4T_b + 4T_a$, which is suitable for most TCP flows.

For a TCP medium flow, a burst contention may trigger fast retransmission even when OBS-layer retransmission is employed. In this case, packets from a given TCP flow may be spread across multiple bursts. Since the retransmitted burst incurs an extra retransmission delay, bursts that are sent after the contending burst may actually reach the egress node prior to the retransmitted burst. The earlier arrival of these other bursts will result in the generation of duplicate ACKs, leading to the triggering of fast retransmission at the source. Once fast retransmission is triggered, the TCP sender will retransmit a lost packet. At the same time, the OBS layer also attempts to recover the packet losses through burst retransmission. Therefore, there will be redundant retransmissions of packets in the network. The false fast retransmissions can be eliminated by reordering at the egress node as in [14].

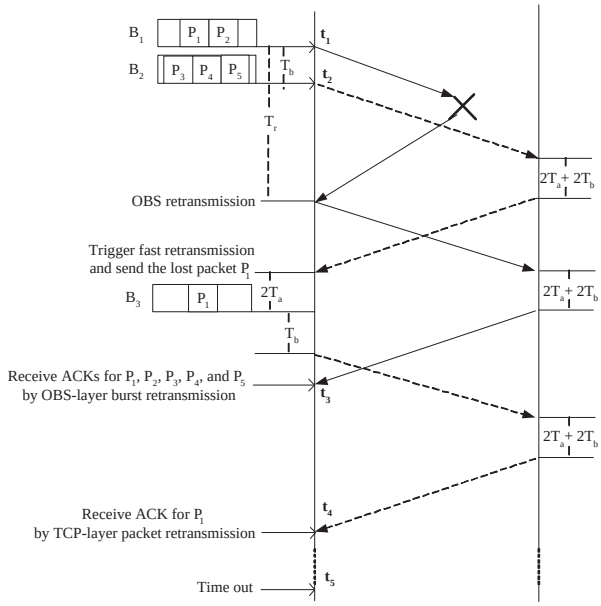


Fig. 3 Two levels of packet loss recovery for a TCP medium flow.

Fig. 3 presents the two levels of packet loss recovery for a TCP medium flow. Packets P_1, P_2, P_3, P_4, and P_5 belong to the same sending window. Packets P_1 and P_2 are assembled into burst B_1, and packets P_3, P_4, and P_5 are assembled into burst B_2. The five packets are acknowledged at time t_3 due to the OBS-layer burst retransmission scheme. However, due to the retransmission delay, the TCP sender triggers fast retransmission, and retransmits the lost packet P_1. The acknowledgement of packet P_1 that is retransmitted by the TCP layer during fast retransmission is received at time t_4. Hence, if $t_3 \leq t_4$, then the TCP layer will not retransmit additional redundant packets (i.e. packet P_2) and the TCP fast retransmission period will be reduced. We have $(T_r + 2T_p + 2T_a + 2T_b) \leq (4T_p + 6T_a + 6T_b)$, thus $T_r \leq (2T_p + 4T_a + 4T_b)$. The delay constraint δ can then be chosen to be $(2T_p + 4T_a + 4T_b)$ for a TCP medium flow, which is same as for a TCP fast flow. Since the value of T_a may be variant for different TCP flows and the value of T_b may differ when different burst assembly mechanisms are used, we only use the knowledge of OBS core networks to perform the burst retransmission. Therefore, we obtain $\delta = 2T_p$ without considering the values of T_a and T_b.

Note that the delay constraint δ may be different for different ingress-egress pairs, depending upon the propagation delay between the ingress-egress pair. Since the routing in OBS networks is usually source routing, in the case that the network topology changes, the ingress node will be able to update the delay constraint for different paths.

5 Performance Analysis

In this section, we develop an analytical model for evaluating the end-to-end burst loss probability for OBS networks with burst retransmission and an analytical model for evaluating TCP throughput when TCP is implemented over an OBS network with burst retransmission.

5.1 Burst Loss Probability of OBS Retransmission

Given the retransmission buffer blocking probability, P_b, we analyze the end-to-end burst loss probability over an OBS network. With burst retransmissions, a burst contention may not result in actual burst loss. Hence, we define *burst contention probability* as the probability that a burst is dropped due to contention at an intermediate node. We also define *burst loss probability* as the probability that a burst does not successfully reach its destination, even after possible retransmissions.

Since the delay constraint is associated with the propagation delay of an ingress-egress pair (s, d), let the delay constraint for ingress-egress pair (s, d) be δ_{sd}, and the propagation delay for an ingress-egress pair (s, d) be $T_{p_{sd}}$. We assume that each retransmission takes an average time of $T_{p_{sd}}$. We can then approximate the maximum number of retransmissions for a burst that is able to be retransmitted as,

$$r_{sd} = \lfloor \frac{\delta_{sd}}{T_{p_{sd}}} \rfloor. \quad (1)$$

We assume that the input load to each ingress-egress pair (s, d) is ρ_{sd}, which does not include the additional load due to retransmitted traffic. Bursts that are blocked by the retransmission buffer will not be retransmitted, the traffic load that is unable to be retransmitted is $P_b\rho_{sd}$, and the traffic load that is able to be retransmitted is $(1 - P_b)\rho_{sd}$.

Let p_c^{sd} be the steady-state end-to-end burst *contention* probability between the ingress node s and egress node d. For each ingress-egress pair (s, d), all dropped bursts that are able to be retransmitted along the route can be retransmitted a maximum of r_{sd} times by the corresponding ingress node. In the worst case, when a burst is retransmitted the maximum number of times, the total load of the ingress-egress pair (s, d) that includes retransmitted traffic, is given by:

$$\phi_{sd} = \rho_{sd}P_b + \sum_{k=0}^{r_{sd}} (\rho_{sd}(1 - P_b)(p_c^{sd})^k). \quad (2)$$

In order to obtain the end-to-end burst contention probability p_c^{sd}, we first compute the burst contention probability on each link. Let the burst contention probability at steady state on link l_{ij} be p_c^{ij}. Given the routes of the ingress-egress pairs, $route(s, d)$, the total load ϕ_{ij} on link l_{ij} is

$$\phi_{ij} = \sum_{\{\forall(s,d)|l_{ij} \in route(s,d)\}} \phi_{sd}. \quad (3)$$

We assume that the burst arrival at each ingress-egress pair (s,d) is Poisson [28]. Hence, the burst contention probability on link l_{ij} can be calculated by using the Erlang-B formula:

$$p_c^{ij} = ErlangB(\phi_{ij}, m). \tag{4}$$

where ϕ is the traffic load on link l_{ij} and m is the number of wavelengths on a fiber link.

We can then obtain the end-to-end burst contention probability of every ingress-egress pair (s,d) based on the burst contention probability on each link. We have:

$$p_c^{sd'} = 1 - \prod_{\{\forall(i,j)|l_{ij}\in route(s,d)\}} (1 - p_c^{ij}). \tag{5}$$

We iterate until p_c^{sd} and $p_c^{sd'}$ converge.

We now calculate the end-to-end burst *loss* probability for an ingress-egress pair (s,d), p_d^{sd}. A burst is lost, if it is able to be retransmitted and its $r_{sd}th$ retransmission fails in the retransmission scheme, or if it is unable to be retransmitted and experiences contention with probability p_c^{sd}. Hence, we have:

$$p_d^{sd} = (1 - P_b)(p_c^{sd})^{r_{sd}} + P_b p_c^{sd}. \tag{6}$$

Then, the end-to-end burst loss probability over an entire network is given by:

$$p = \frac{\sum_{\forall(s,d)} \rho_{sd} p_d^{sd}}{\sum_{\forall(s,d)} \rho_{sd}}. \tag{7}$$

5.2 TCP over OBS with Controlled Burst Retransmission

In this section, we analyze the TCP throughput over an OBS network with burst retransmission. In the existing analysis of TCP throughput over an OBS network without burst retransmission, burst loss probability is equal to burst contention probability. When OBS retransmission is employed, a burst is only considered lost if it experiences contention and it is not successfully retransmitted. Thus, the burst loss probability differs from the burst contention probability. In the analysis, we must take into account both burst loss probability and burst contention probability. We analyze the TCP SACK throughput for a TCP fast flow and a TCP medium flow.

As defined in [27], a TCP sending *round* refers to the period during which all packets in the sending window are sent and the first ACK for one of the packets in the sending window is received. We assume that the time needed to send all the packets in the sending window is less than RTT. Hence, the duration of a round is equal to RTT. We also assume that the number of packets that are acknowledged by a received ACK is one ($b = 1$ in [11]).

We introduce the following notation for a TCP flow:

p_c: burst contention probability.
p_d: burst dropping probability.
B: TCP throughput.
W_m: TCP maximum window size (in packets).
Z^{TO}: duration of a sequence of TOs.
H: # of TCP segments sent in Z^{TO}.

5.2.1 TCP fast flow

Our analysis of a TCP fast flow is similar to that in [10]. However, in our analysis for the case with OBS retransmission, the successfully retransmitted bursts are treated differently from the bursts that do not experience any contention. The retransmitted bursts suffer from an extra retransmission delay, which has a negative effect on the TCP throughput.

Since a TCP fast flow does not trigger TD, multiple successful sending rounds are only followed with one or multiple lossy rounds. Therefore, as in [10], a given time out period includes a sequence of successful rounds and a sequence of lossy rounds. In this time out period, let X be the number of successful rounds, Y be the number of segments sent before the first lossy round, and A be the duration of the sequence of successful rounds. We can then calculate the TCP throughput as given below:

$$B^f = \frac{E[Y] + E[H]}{E[A] + E[Z^{TO}]}. \tag{8}$$

The sequence of successful rounds consists of a portion of rounds in which the burst does not experience contention and a portion of rounds in which the burst experiences contention, but is successfully retransmitted. Hence, we obtain the probability of a successful round in which a burst experiences contention but is successfully retransmitted as

$$p_{sr} = \frac{p_c - p_d}{1 - p_d}. \tag{9}$$

The probability of a successful round in which there is no burst contention can be calculated as

$$p_{nc} = \frac{1 - p_c}{1 - p_d}. \tag{10}$$

We assume that each retransmission of a burst takes an average time of T_p. Then, the average number of retransmissions for a retransmitted burst, given that the burst needs to be retransmitted at least once and the retransmission is successful, is

$$E[r] = \sum_{i=1}^{\lfloor \delta/T_p \rfloor - 1} i p_c^{i-1}(1 - p_c) + \lfloor \frac{\delta}{T_p} \rfloor p_c^{(\lfloor \frac{\delta}{T_p} \rfloor - 1)}. \tag{11}$$

Hence, the average round trip time experienced by a successfully retransmitted burst is

$$RTT_r = RTT + E[r]T_p. \tag{12}$$

We then obtain $E[A]$ as

$$E[A] = p_{sr} E[X] RTT_r + p_{nc} E[X] RTT. \tag{13}$$

Based on the equations (14), (16), (18), and (28) in [10], we have

$$E[Z^{TO}] = RTO\frac{f(p_d)}{1-p_d}, \tag{14}$$

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

$$E[H] = \frac{p_d}{1-p_d}, \tag{15}$$

$$E[X] = \frac{1-p_d}{p_d}, \tag{16}$$

and

$$E[Y] = \begin{cases} \frac{1}{p_d^2} & p_d > \frac{1}{W_m} \\ \frac{W_m}{p_d} & otherwise. \end{cases} \tag{17}$$

Since only burst losses result in TOs for a fast flow, the burst loss probability in an OBS network with burst retransmission is applied in the above equations.

By substituting equations (13), (14), (15), and (17) into (8), we have

$$B^f = \frac{p_d^3 - p_d + 1}{p_d[(1-p_d)(p_c-p_d)RTT_r + (1-p_d)(1-p_c)RTT + p_d f(p_d)RTO]}, \tag{18}$$

when $p_d > \frac{1}{W_m}$. And

$$B^f = \frac{p_d^2 + W_m - W_m p_d}{(1-p_d)(p_c-p_d)RTT_r + (1-p_d)(1-p_c)RTT + p_d f(p_d)RTO}, \tag{19}$$

when $p_d \leq \frac{1}{W_m}$.

5.2.2 TCP SACK medium flow

In this section, we analyze the TCP SACK throughput for a TCP medium flow over an OBS network with burst retransmission. TCP SACK triggers fast retransmission when a burst contention occurs, and exits fast retransmission if all the TCP packets that were in the sending window when fast retransmission was triggered are acknowledged. At moderate to high loads, multiple bursts that contain packets belonging to this sending window might experience contention; however, only the first contending burst will trigger a TD event. In other words, some of the burst contentions may not trigger TD events. The analysis in [11] assumes that a burst loss always triggers a TD event. In our analysis, we must obtain the probability of triggering a TD event based on both the burst loss probability and the burst contention probability. Particularly, as OBS retransmission leads to higher burst contention probability, it becomes more important to obtain the probability of triggering a TD event.

Since a TCP medium flow may trigger both TO and TD events, a sequence of TD events may follow a sequence of TO events. The analysis of a sequence of TOs is same as that of fast flow. We focus on the analysis of a sequence of TD. A TD period is defined as the period between two consecutive TD events.

Let us define the following notation as in [11]:

S: # of packets from a TCP flow assembled into a burst.
TDP: duration of a TD period.
Y: # of TCP segments sent during the TD period.
W_X: sending window size of the last round in the TD period.
W_b: average number of bursts containing packets from a window of size $E[W_X]$.
Q: probability that a loss indication ending a TDP is a TO.
p_{TD}: probability of triggering a TD event.

We obtain the TCP throughput by,

$$B^m = \frac{E[Y] + QE[H]}{E[TDP] + QE[Z^{TO}]}. \tag{20}$$

We first need to obtain p_{TD}. The packets from a window of size $E[W_X]$ will be distributed over an average of W_b bursts, where $W_b = \frac{E[W_X]}{S}$. The first burst to experience a contention will trigger a TD event. However, none of the following $W_b - 1$ bursts, regardless of whether or not they experience contention, will trigger a TD event, since TCP SACK exits fast recovery only after all the packets in the window have been acknowledged. Since burst contentions in an OBS network are independent events, the average number of remaining bursts that experience contention but do not trigger TD is $(W_b - 1)p_c$. Note that the initial burst in W_b that triggered the TD event is not included. The ratio of the number of contentions that trigger TD to the number of contentions that do not trigger TD is equal to the ratio of the probability of a TD event to the probability of a non-TD contention event. Thus, we have

$$\frac{1}{(W_b - 1)p_c} = \frac{p_{TD}}{p_c - p_{TD}}. \tag{21}$$

We then have

$$p_{TD} = \frac{p_c}{(\frac{E[W_X]}{S} - 1)p_c + 1}. \tag{22}$$

We use equation (9) in [11] to obtain $E[W_X]$, where

$$\frac{3}{8}E[W_X]^2 - E[W_X] - \frac{S}{p_{TD}} = 0. \tag{23}$$

Note that we replace the burst loss probability in the original equation by p_{TD}. By substituting (22) into (23), we have

$$E[W_X] = \frac{8}{3} + \frac{4}{3}\sqrt{4 - \frac{3}{2}(S - \frac{S}{p_c})}. \tag{24}$$

For the case in which burst contention and burst loss probabilities are very low, the sending window size remains at W_m for a long time, and the analysis is the same as in [11]. The TCP throughput can be approximated as

$$B^m \approx \frac{W_m}{RTT}. \tag{25}$$

We now focus on the case in which burst contention probability is moderately high, and the sending window size

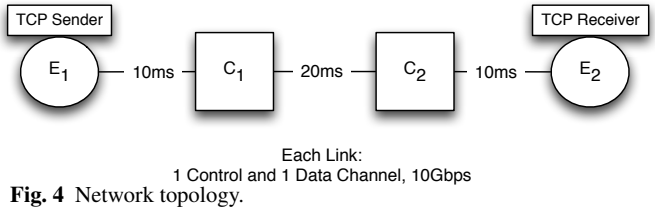


Fig. 4 Network topology.

rarely reaches W_m. Based on equations (5), (7), and (13) in [11], we have

$$E[Y] = \frac{3}{2}E[W_X] + \frac{1-p_{TD}}{p_{TD}}S, and \tag{26}$$

$$E[TDP] = RTT(\frac{1}{2}E[W_X] + 1). \tag{27}$$

Since TO events can only be triggered when all bursts in a sending window are dropped, from (25) in [11], we have

$$Q \approx p_d^{(\frac{E[W_X]}{S}-1)}. \tag{28}$$

By substituting equations (14), (15), (26), (27), and (28) into (20), for a small value of p_c and $p_d \leq p_c$, we can approximate B^m as,

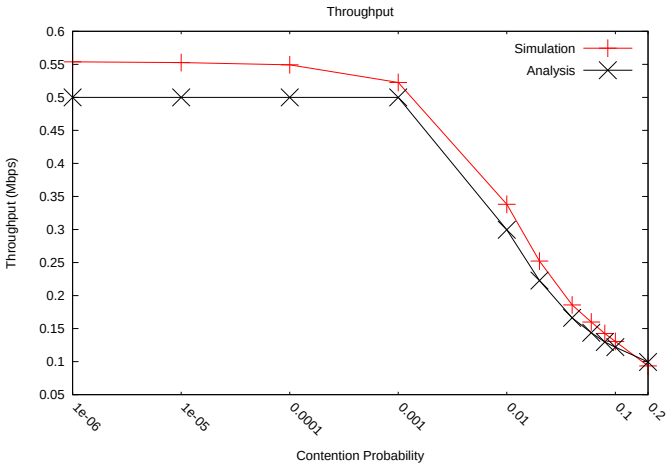
$$B^m \approx \frac{1}{RTT}\sqrt{\frac{3S}{2p_c}}. \tag{29}$$

6 Numerical Results

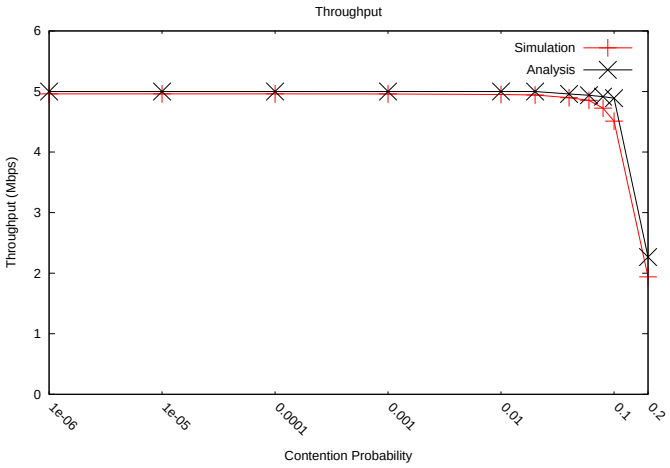
We develop simulations to verify the analytical results and to evaluate the performance of TCP over OBS using controlled burst retransmission. We first use a network with only TCP flows and a constant burst loss probability to compare with the analytical model. We then simulate a more realistic network scenario wherein TCP flows encounter loss due to background constant-rate UDP traffic. The simulations with background CBR-based UDP traffic allow us to sustain a constant input rate since TCP will reduce its input rate in response to dropped bursts. All simulations were performed using ns2 [33] simulator with the OWns [34] module with TCP and background UDP traffic.

6.1 Analytical Results

In this section, we verify the analytical model developed in Section 5. The topology used for verifying the analytical model using simulations is shown in Fig. 4. We limit the network complexity to obtain precise performance results. A single TCP SACK flow at the ingress (E_1) sends a 1GB file to the egress (E_2). In both the simulation and analytical model T_p is $40ms$ and T_b is $10ms$. The delay constraint is $2T_p$ and the packet size is 1KB. For the simulations, we only drop bursts containing TCP segments, not bursts containing acknowledgements, and random burst dropping is introduced at the left-most core node (C_1).



(a) Average throughput of TCP medium flow.

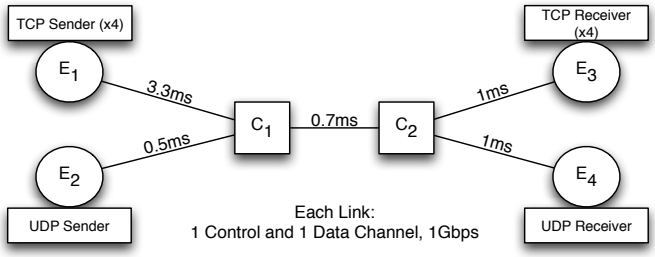


(b) Average throughput of TCP fast flow.

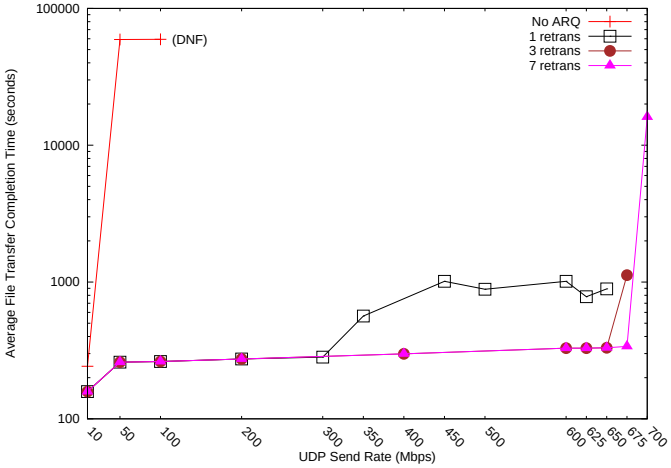
Fig. 5 TCP throughput vs. burst contention probability for analysis and simulation results.

In Fig. 5(a) we compare the results of the analytical model to the simulations for TCP medium flow. In both the simulation and analytical model we set the maximum window size, W_m, to 50 packets. In the simulations we allowed a burst to contain a maximum of 5 TCP segments, so S is set to 5 in the analytical model. In Fig. 5(b) we compare the results for TCP fast flow. Here W_m is set to 500 packets in the simulation and analytical model.

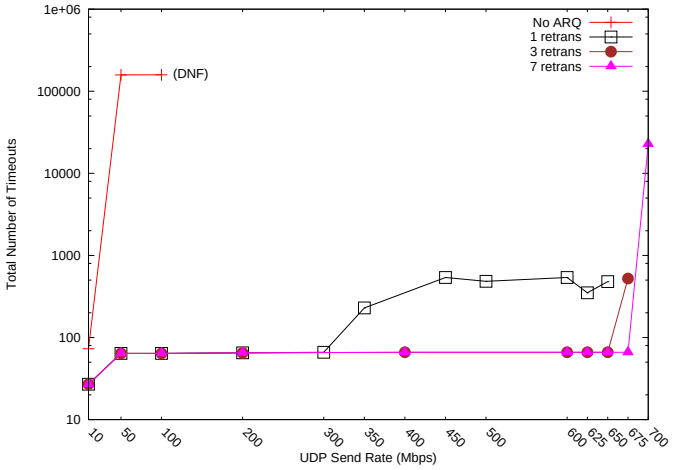
From the figures, we can see that the simulation results match the analytical results. We chose to stop the comparison for contention probability levels greater than 20%. At these higher contention probability levels we cannot create medium flows in the simulations because the congestion window becomes too small. In Fig. 5(a), our model slightly overestimates the burst loss leading to underestimating the throughput of medium flows. The model is very accurate for fast flows shown in Fig. 5(b).



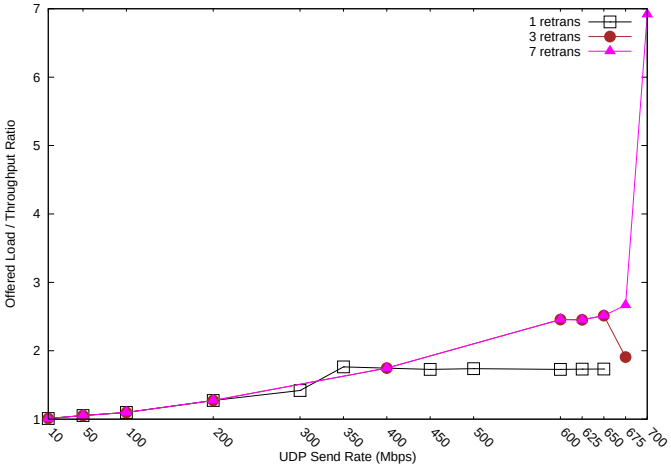
(a) Topology for simulations with background CBR-baesd UDP traffic.



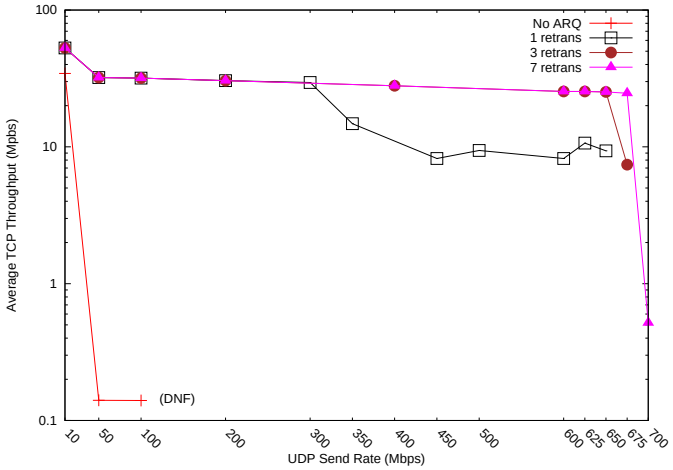
(b) Average file transfer completion time.



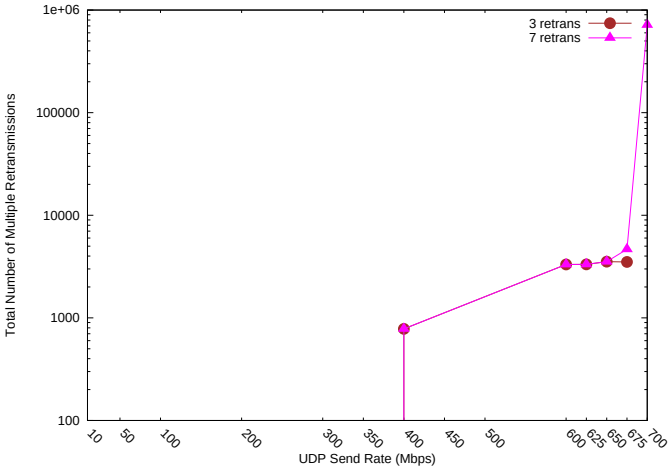
(c) Total number of timeouts.



(d) Offered load to actual throughput ratio.



(e) Average TCP throughput per flow.



(f) Total number of multiple retransmissions.

Fig. 6 Simulation results with background CBR-based UDP traffic.

6.2 Performance of OBS with controlled ARQ and background CBR traffic

In this section we ran simulations with background UDP traffic to determine if there is a limit on the effectiveness of burst retransmission, where the retransmission starts causing too much congestion in the network. Here, we use CBR-based UDP flows to maintain sustained input rates while evaluating burst retransmission.

The machine used to perform the simulations ran ns2 version 2.33 with the OWns module v0.9a to simulate OBS networks. The machine has a Intel Core2 Quad Core processor at 2.66GHz with 8GB RAM. It runs Fedora Core 9 (Sulphur), 64-bit, with Linux kernel version 2.6.25.

We use the simulation topology as shown in Fig. 6(a). We only use one data channel with 1Gbps bandwidth. The burst assembly mechanism uses a maximum burst size of 1MB with a burst assembly time (BAT) of 10ms. There are four TCP flows on ingress E_1 that send to receivers on egress E_3. An FTP traffic generator is used to send a 1GB file over each of the flows. The TCP flows use TCP SACK. TCP's advertised window at the receiver is set to 1000 packets and remains constant throughout the simulation. A single UDP sender is on ingress E_2 sending data to E_4 so the TCP and UDP flows share the bottleneck link. A constant bit rate traffic generator is used on top of UDP. We vary the UDP send rates in the simulations. We implement controlled burst retransmission by setting a hard limit on the maximum number of retransmissions for these simulations, instead of basing it on the T_p value.

In Fig. 6(b) we plot the average file transfer completion times versus UDP traffic load. We can see that even at low UDP send rates, TCP's performance without burst retransmission is drastically impacted. In this case we have fast flows (TCP sender's window is less than 220 packets for all simulations), so when a burst is dropped, TCP loses an entire window of data and will enter the slow-start phase. From the same figure we observe up to three orders of magnitude improvement with controlled burst retransmissions. We see that burst retransmission is able to prevent a large number of false timeouts at the TCP sender through burst retransmissions and therefore improve performance. The number of timeouts is plotted in Fig. 6(c). This figure shows that the number of timeouts is closely related to the average completion time since the graphs are almost identical in structure. This is a result of having TCP fast flows, where a burst loss almost always results in a timeout, which drastically affects the TCP sender's send rate. There are very few fast retransmissions because of the fast flows. If the burst sizes were smaller, the TCP sender's window would be split across multiple bursts leading to more fast retransmissions. We explore this further in subsection 6.3.

In Fig. 6(d) we examine the extra offered load on the network due to retransmissions. We measure the ratio of the offered load to the actual throughput at the destination, where the offered load is the combination of original and retransmitted TCP traffic. From Fig. 6(e) we can see that the average TCP throughput decreases slightly for increasing UDP send rates with burst retransmission, but the ratio of offered load to actual throughput in Fig. 6(d) is increasing during the same period. For example, a ratio of 2 implies that 50% of the traffic is retransmitted traffic. This is a result of the extra traffic created by retransmission.

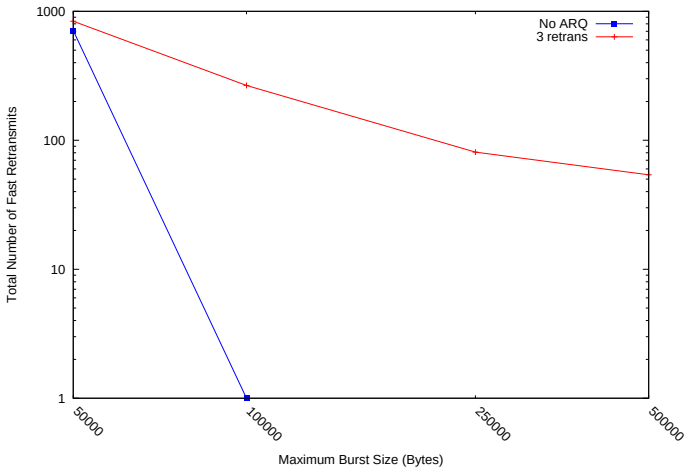
Fig. 6(f) shows the total number of multiple retransmissions for these simulations. The total number of multiple retransmissions is a count for the total number of retransmission attempts made after the first one across all bursts. This metric allows us to assess the effectiveness of attempting more retransmissions beyond the first. Up until a UDP send rate of 400Mbps, only a single retransmission is able to recover from any dropped bursts.

With a UDP send rate of 675Mbps and a maximum of three retransmissions we can see that both TCP throughput (Fig. 6(e)) and the ratio (Fig. 6(d)) decrease. Observing Fig. 6(f) we can also see that the total number of multiple retransmissions from 650Mbps to 675Mbps did not change for three retransmissions, suggesting that a maximum of three retransmissions is not sufficient to recover from the loss with UDP sending at 675Mbps. The ratio drops at 675Mbps because three retransmissions is no longer enough to recover from losses, so both the throughput and offered load decrease when TCP cuts its send rate.

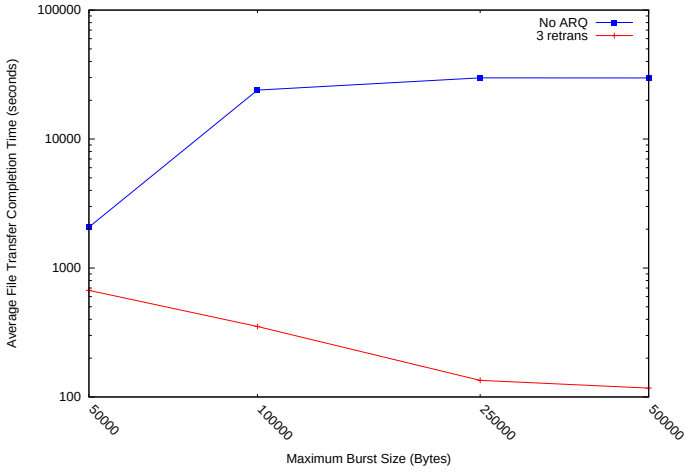
For the case of UDP sending at 700Mbps with a maximum of seven retransmissions we see that with a decrease in TCP throughput (Fig. 6(e)), there is an increase in the load-throughput ratio (Fig. 6(d)). Also, looking at Fig. 6(f) we can see the total number of multiple retransmissions is still increasing. This suggests that the problem is not that there are not enough retransmissions to recover from losses, but that the extra bursts created by those retransmissions are just causing more contentions, which is just hurting performance instead of helping. We believe the two important cases are shown in our plots, those being seven maximum retransmissions at 700Mbps UDP send rate where the retransmissions overload the network and three maximum retransmissions at 675Mbps UDP send rate where the number of retransmissions is not enough to recover from all loses. This confirms that it is very important to implement a controlled version of ARQ.

The ARQ scheme only begins causing too many contentions when the load is very high (700Mbps send rate, note: the link rate is 1Gbps) and when a large number of retransmission attempts are allowed. For low to medium network loads, there is clearly a big performance advantage. We also ran simulations with a maximum of two retrans-

missions and found that it performed the same as three up until 650Mbps send rate. This suggests that a maximum of two retransmissions should be sufficient to recover from loss in most scenarios. In order to ensure that the ARQ scheme does not negatively impact the network performance at high loads we can implement simple priority-based scheduling in the core. Original bursts can be given high priority while retransmitted bursts can be given low priority.



(a) Total number of fast retransmissions.



(b) Average completion times sending 500MB file.

Fig. 7 Total fast retransmits and average completion time with HS-TCP and 500MB file.

We note that given the specifications of our simulation machine, we did not plot further points in our graphs because of the very long run-times of our simulations at high UDP send rates. For example, the simulation with a maximum of two retransmissions with a UDP send rate of 700Mbps was past 87,000 seconds of simulation time, much higher than the point we have for a maximum of seven retransmissions at that rate. Also, for 675Mbps both one and two maximum retransmissions was over 80,000 seconds. Similarly, for the No ARQ simulations, at a 200Mbps send rate the simulation time was past 80,000 seconds.

6.3 HS-TCP (with SACK) over OBS with Different Burst Sizes

HS-TCP is better suited for high bandwidth-delay product flows over OBS networks. In this section we evaluate the performance HS-TCP with SACK over OBS networks with different burst sizes. In Fig. 7 (a) and (b) we compare the effect of varying the maximum burst size. Each TCP sender transmits a 500MB file using HS-TCP [29]. The UDP send rate is set to 50Mbps. Everything else is the same as in the previous set of simulations. We can see that for smaller burst sizes, there are more fast retransmits because the TCP sender's window is spread out across multiple bursts (medium flows). Losing one burst may result in a fast retransmit. Similarly, if many TCP flows are put into a single larger burst, the number of fast retransmits will also increase. With burst retransmission, the performance increases as burst size increases. With a large maximum burst size, all of the TCP sender's window will fit in a single burst. When this burst is dropped, the burst retransmission can successfully retransmit the burst and no *dupacks* will be sent from the TCP receiver because the entire send window will be recovered. Without retransmission, however, when the maximum burst size increases, a contention can lead to the entire TCP sender window being dropped, resulting in timeouts instead of fast retransmissions as in the case of smaller burst sizes.

We can also see in Fig. 7(a) that at 50Mbps send rate, burst retransmission does not prevent false fast retransmits. This limitation can be resolved by reordering the bursts at the egress using a technique proposed for wireless networks [14].

7 Conclusion

If implemented as a high-speed core network, an OBS network must be able to present an accurate indication of the network congestion to the higher layers; otherwise the higher layers may falsely assume that the core network is congested and may take unnecessary actions that will lead to significant degradation in performance. In this paper, we proposed a controlled burst retransmission scheme for an OBS network. The retransmission scheme not only improves the burst loss probability, but also presents more accurate OBS network congestion information to the higher layers, such as TCP. By presenting a better picture of network congestion, the OBS retransmission scheme can reduce the number of FTOs and can also reduce the TCP fast recovery period, thereby significantly improving the TCP throughput. OBS retransmission requires electronic buffers at the ingress, and

the buffer capacity can be determined by the delay constraint. We analyzed the burst loss probability for an OBS network with burst retransmission by taking into account of the delay constraint, and we analyzed the TCP throughput for the case in which TCP is implemented over an OBS network with burst retransmission. Through extensive simulations we have found that controlled burst retransmissions significantly improve the performance of TCP over OBS. We show that our simulation results closely match the analytical results. We also determined the maximum number of retransmissions for TCP over an OBS network with burst retransmission.

In our current retransmission scheme, the retransmitted bursts have the same priority as the original bursts. A possible area of future work is implementing priorities so that during a contention a retransmitted burst will be dropped in favor of an original burst so the retransmitted bursts do not affect original traffic. Another area of future work is to evaluate the effect of controlled burst retransmission scheme on recently proposed TCP flavors that can achieve better performance in a high-bandwidth and high-delay network environment, such as CUBIC [37] and Fast TCP [30]. A third area of future work is to compare the controlled retransmission scheme with other existing loss recovery schemes in OBS.

References

1. J.P. Jue and V.M. Vokkarane, Optical Burst Switched Networks, Springer, 2005.
2. C. Qiao and M. Yoo, "Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet," *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69-84, Jan. 1999.
3. J.Y. Wei and R.I. McFarland Jr., "Just-in-time Signaling for WDM Optical Burst Switching Networks," *Journal of Lightwave Technology*, vol. 18, no. 12, pp. 2019-2037, Dec. 2000.
4. I. Baldine, G.N. Rouskas, H.G. Perros, and D. Stevenson, "JumpStart: A Just-in-Time Signaling Architecture for WDM Burst-Switched Network," *IEEE Communications*, vol. 40, no. 2, pp. 82-89, Feb. 2002.
5. X. Yu, C. Qiao, and Y. Liu, "TCP Implementation and False Time Out Detection in OBS Networks," *Proceedings, IEEE INFOCOM*, 2004.
6. B. Komattireddy and V.M. Vokkarane, "Source-Ordering for Improved TCP Performance over Load-Balanced Optical Burst-Switched (OBS) Networks," *Proceedings, IEEE/CreateNet BROADNETS 2007,* Optical Networking Symposium, Raleigh, NC, Sep. 2007.
7. I. Stoica, R. Morris, and et. al., "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications," *Proceedings, ACM SIGCOMM*, 2001.
8. K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," *Proceeding, ACM SIGMETRICS*, 2003.
9. I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling Scalable Virtual Organizations," Intenational Journal of High Performance Computing Applications," vol. 15, no. 3, pp. 200-222, 2001.
10. A. Detti and M. Listanti, "Impact of Segments Aggregation on TCP Reno Flows in Optical Burst Switching Networks," *Proceedings, IEEE INFOCOM*, 2002.
11. X. Yu, C. Qiao, Y. Liu, and D. Towsley, "Performance Evaluation of TCP Implementations in OBS Networks," *Technique Report 2003-13*, The State University of New York at Buffalo, 2003.
12. S. Gowda, R. K Shenai, K. Sivalingam, and H. C. Cankaya, "Performance Evaluation of TCP over Optical Burst-Switched (OBS) WDM Networks," *Proceedings, IEEE ICC*, 2003.
13. H. Balakrishnan and et. al., "Improving TCP/IP performance over wireless networks," *Proceedings, 1st ACM Conf. on Mobile Computing and Networking,* Nov. 1995.
14. C. Barakat and A.A. Fawal, "Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic," *Performance Evaluation Journal,* vol. 57, pp. 423500, Aug. 2004.
15. D. Barman, I. Matta, E. Altman, and R.E. Azouzi, "TCP optimization through FEC, ARQ and transmission power tradeoffs," *Proceedings, Intl. Conf. on Wired/Wireless Internet Communications (WWIC),* Feb. 2004.
16. A. V. Bakre and B. R. Badrinath, "Implementation and performance evaluation of indirect TCP," *IEEE Transaction on Computers,* vol. 46, Mar. 1997.
17. I. Chlamtac, A. Fumagalli, et. al., "CORD: Contention Resolution by Delay Lines," *IEEE Journal on Selected Areas Communication*, vol. 14, pp.1014-1029, June 1996.
18. 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.
19. S. Danielsen, P. Hansen, and K. Stubkjear, "Wavelength Conversion in Optical Packet Switching," *Journal of Lightwave Technology*, vol. 16, no. 12, pp. 2095-2108, Dec. 1998.
20. V. M. Vokkarane, J. P. Jue, and S. Sitaraman, "Burst Segmentation: an Approach for Reducing Packet Loss in Optical Burst Switched Networks," *Proceedings, IEEE ICC*, 2002.
21. F. Forghierri, A. Bononi, and P. Prucnal, "Analysis and Comparison of Hot-potato and Single-buffer Deflection Routing in Very High Bit Rate Optical Mesh Networks," *IEEE Transaction on Communications*, vol. 43, no.1, pp. 88-98, Jan. 1995.
22. C. Hsu, T. Liu, and N. Huang, "Performance Analysis of Deflection Routing in Optical Burst-switched Networks," *Proceedings, IEEE INFOCOM*, 2002.
23. W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *RFC 2001*, 1997.
24. S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", *RFC 2582*, 1999.
25. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," *RFC 2018*, 1996.
26. K. Fall and S. Floyd, "Simulation-based comparisons of Taho, Reno, and SACK TCP," *Proceeding of ACM SIGCOMM*,1996.
27. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, April 2000.
28. Q. Zhang, V. Vokkarane, J.P. Jue, and B. Chen, "Absolute QoS Differentiation in Optical Burst-Switched Networks," *IEEE Journal on Selected Areas in Communications,* Optical Communications and Networking Series, vol. 22, no. 9, pp. 1781-1795, Nov. 2004.
29. S. Floyd, "HighSpeed TCP for Large Congestion Windows," *RFC 3649*, 2003.
30. C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *Proceedings, IEEE INFOCOM*, 2004.
31. C. Cameron, J. Choiy, S. Bilgrami, et al, "Fixed-Point performance analysis of TCP over optical burst switched networks," *proceedings, ATNAC,* 2004.
32. M. Zukerman, E. Wong, Z. Rosberg, G. M. Lee, and H. L. Vu, "On teletraffic applications to OBS," *IEEE Communication Letters,* vol. 8, no. 1, 2004.
33. Network Simulator 2, "http://www.isi.edu/nsnam/ns/"