# TCP-aware Load-Balanced Routing in Optical Burst-Switched (OBS) Networks

Bharat Komatireddy, Deepak Chandran, and Vinod M. Vokkarane
* *Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA 02747, USA*
*E-mail: {g_bkomatired,g_dchandran,vvokkarane}@umassd.edu*

**Abstract**

TCP-over-OBS is a promising transport paradigm to support next-generation-Internet. Load-balanced OBS improves loss-performance. We identify the ill-effects of load-balancing on TCP-performance due to false-time-outs and false-triple-duplicates. We propose source-ordering to improve TCP-throughput by three-four times.

**OCIS Codes:** 060.4250 Networks and 060.4510 Optical communications.

## I. Introduction

In optical burst switching (OBS), data to be transmitted is assembled in to bursts and are switched through the network all optically. Each burst has an associated control packet called the burst header packet (BHP) and the BHP is sent ahead of time in order to configure the switches along the bursts' route. In OBS networks, apart from the data channels, each link has one or more control channels to transmit BHPs. BHPs carries information about the burst such as source, destination, burst duration, and offset time. Offset time is the time at which the burst and BHP are separated at the source and the subsequent intermediate nodes. The offset time allows for the BHP to be processed at each intermediate node before the data burst arrives. As the BHP travels from source to destination, it is processed at each intermediate node in order to configure the optical switches accordingly. Then the data burst cuts through the optical layer avoiding any further delays. Bandwidth is reserved only for the duration of the burst, this reservation technique is called just-enough-time (JET) [1].

The primary issue in the OBS core network is contention resolution, since the core does not have any buffers. Contention occurs when two or more bursts contend for the same output port at the same time. There are several contention resolution techniques, such as optical buffering, wavelength conversion, and deflection routing []. These contention resolution techniques are reactive techniques that try to resolve the contention when it occurs. These contention resolution techniques attempt to minimize the loss based on the local information at the node. An alternative to contention resolution is to avoid contention before it happens. In contention avoidance technique, the contentions can be avoided by source policing or by routing the traffic in such a way that the congestion in the network is minimized. In this paper, we adopt dynamic congestion-based load-balanced routing techniques to avoid contentions in the OBS network [2].

Many TCP-based applications, such as Web (HTTP), Email (SMTP), peer-to-peer file sharing, and grid computing, account for majority of data traffic in the Internet; thus understanding and improving the performance of TCP implementations over OBS networks is critical. TCP flavors primarily differ in their implementation of congestion-control mechanisms and their handling of high-bandwidth-delay flows. TCP Reno is one of the widely deployed TCP versions in the Internet. TCP Reno was primarily developed using the fundamental assumption that the underlying medium is electronic in nature, and the packets experience queueing delay during the congestion in the IP routers along the TCP flow. TCP Reno employs packet loss-based congestion-control using *timeout* (TO) and *triple duplicate* (TD) based mechanisms.

In this paper, we analyze the performance of TCP Reno over a load-balanced optical burst-switched network. The remainder of the paper is organized as follows. Section II provides background information on congestion-based load-balanced routing in OBS networks. Section III discusses the issue of supporting TCP over an independently load-balanced OBS network. Section IV discusses the simulations results and Section V concludes the paper.

## II. Load-Balanced Routing in OBS

Contention avoidance by balancing the load in the network involves two stages, *route calculation* and *route selection*. Route calculation can be implemented in a static or a dynamic manner. In *static route calculation* one or more routes are pre-calculated based on a static metric such as physical-distance or number of hops. In *dynamic route calculation* routes are computed periodically based on certain dynamic traffic information such as link congestion or number of contentions. In the route selection stage, one of the routes which are calculated (either statically or dynamically) is selected in a static or a dynamic manner for the burst transmission. In *static route selection*, a fixed fraction of traffic is sent on all the alternate paths. In *dynamic route selection*, routes are selected based on certain dynamic feedback information. The dynamic feedback information is the congestion on the link and the congestion information is communicated back to the source periodically for every interval $\tau$ using either interlink-state broadcasts or probe packets.

In this paper, we employed *static route-calculation with dynamic route-selection*. We first calculate two link-disjoint shortest-hop paths for every source-destination pair. After every interval $\tau$, we dynamically select the least-congested of the two paths and transmit the burst on that path. In order to implement this technique, each core node has to maintain a node information table, $L_{node}$, that stores the offered load on each of the node's outgoing link. Let $\tau_s$ and $\tau_d$ be the duration of successful burst arrivals and dropped burst arrivals during the interval $\tau$, respectively. The offered load on each of the node's outgoing link is expressed as the duration of all arriving bursts over the interval $\tau$, is given by, $L_{i,j} = \frac{\tau_s + \tau_d}{\tau}$. Each OBS edge node has to maintain a network load information table, $L_{net}$, that stores the information about the offered load of all links in the network. A link is congested, if $L_{i,j} \geq P_{max}$, where $P_{max}$ is the maximum load threshold on a link. At every $\tau$ seconds, all the ingress OBS node dynamically selects the least-congested path (among the two static paths) to all their destination nodes using the congestion-information of all the link along the two pre-calculated paths.

## III. TCP Reno over Load-Balanced OBS

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.

In TCP Reno, after a TD loss is detected, the source retransmits one lost segment, reduces the size of congestion window by half, and enters into a fast recovery phase. During the fast recovery phase, the source increases the congestion window by one segment for each duplicate ACK that it receives. After receiving half a window of duplicate ACKs, the congestion window size will be the same as the window size prior to the TD detection. Thus, the source can send a new packet for each additional duplicate ACK that it receives. The source exits fast recovery upon the receipt of the ACK that acknowledges the retransmitted lost segment, and enters into a *congestion avoidance* phase. TCP Reno is suitable for single segment loss in the sending window, but does not handle multiple losses well.

TCP Reno [3] is a loss-based TCP versions that use packet-loss events to estimate the available bandwidth in network. TCP Reno sender react to a burst drop by duplicate ACKs. TCP implementations performs slow start on TO and TO indicates congestion in an IP network. This loss-based approach may cause *False Time Out* (FTO) [4] and *False Triple Duplicate* (FTD) in an OBS network when there is no serious congestion in the network. The FTO is caused because of random burst contention instead of IP router buffer overflow. While FTD is caused because of the out-of-order arrival of TCP segments at the destination due to the delay-differential between the primary and the alternate paths.

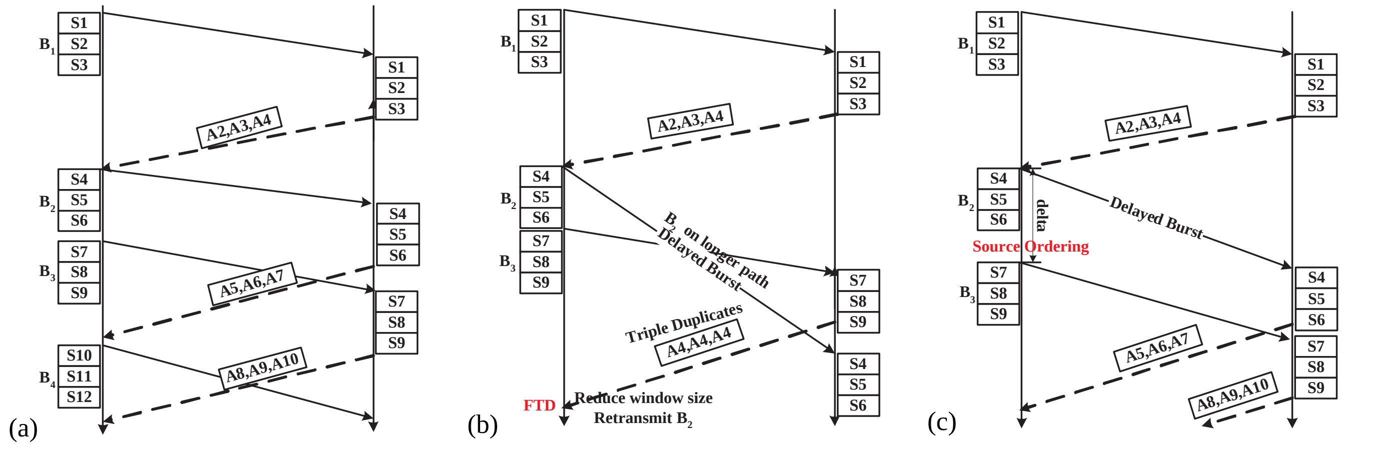

Fig. 1. (c) TCP over OBS with fixed routing (b) FTD in TCP-over-OBS with load-balanced routing. (c) Source ordering to minimize FTD (and FTO) in TCP-over-OBS with load-balanced routing.

In load-balanced routing has a primary path and an alternate path. The alternate being link-disjoint from the primary, usually has a longer delay. In such a scenario, the bursts transmitted on the alternate path incurs longer delay compared to the bursts transmitted on the primary path. The path delay-differential ($\delta$) encountered may cause out-of-order reception of TCP segments (IP packets) at the destination, resulting in FTOs and FTDs.

Consider the following illustration scenario to better understand the issue of FTOs and FTDs due to load-balanced routing in OBS networks. In Fig. 1(a), Burst B1 consisting of three segments [S1,S2,S3] is transmitted and the corresponding acknowledgements [A2,A3,A4] are received. Assuming that the flow is in slow-start phase, congestion window doubles and the sender can possibly send at least six packets. Burst B2 consisting of segments [S4,S5,S6] is sent followed by Burst B3 consisting of segments [S7,S8,S9] and so on. In Fig. 1(b), load-balanced routing in the OBS-layer may result in Burst B2 and Burst B3 being transmitted on two different paths, say B2 on secondary path and B3 on the primary shortest path. The Burst B2 [S4,S5,S6] gets delayed due to longer alternate path, Burst B3 [S7,S8,S9] reaches destination before Burst B2 since Burst B3 contains three out-of-order segments [S7,S8,S9], the receiver will send three duplicate ACKs [A4,A4,A4]. This results in FTDs at the TCP sender. Note that if the path delay-differential ($\delta$) between the primary and the secondary paths is significant, the TCP sender may experience FTOs. In Fig. 1(c), we resolve this issue by implementing *source ordering* at the OBS ingress edge node. We delay the burst for the amount of time equivalent to the path delay-differential of the two routes, using electronic buffering at the ingress OBS edge node. Note that the ingress node is aware of the path delay-differential since OBS predominantly implements source-routing.

## IV. Simulation Results

In order to evaluate the proposed mechanism, we have obtained performance results using *ns-2*, using the 6-node OBS network depicted in Fig. 2(a). The following are the network assumptions: Every fiber link has four data channels operating at the transmission rate of 1 Gb/s. The time-based burst assembly algorithm is adopted, with a timer value of 20 ms. The OBS network implement JET with LAUC-VF. The offset time is 50 $\mu$s and the per-node burst header processing time is 5 $\mu$s. We have set up 100 TCP Reno flows from Node 1 to Node 6 using 100 FTP sources (with infinite data) generating packets of average size being 2 KB. Load-balanced routing is implemented in the OBS-layer with interval, $\tau = 1$ ms and congestion threshold, $p_{max} = 0.1$. We can observe for the network topology that the primary shortest-hop path is $P_1(N_1 - N_2 - N_4 - N_6)$ and the alternate link-disjoint path is $P_2(N_1 - N_3 - N_5 - N_6)$. In order to simulate the effect of path delay-differential between the primary and the alternate paths on TCP performance, we vary the delay of Link $L_{3,5}$ from 10 ms to 210 ms to generate $\delta$ values ranging from 0 ms to 200 ms. We plot the different performance metrics for three different values of $\delta$ (0 ms, 100 ms, and 200 ms). In order to understand TCP performance with and without load-balanced OBS routing, we also plot for the default scenario (referred as *baseline*), wherein the OBS network does not implement load-balanced routing.

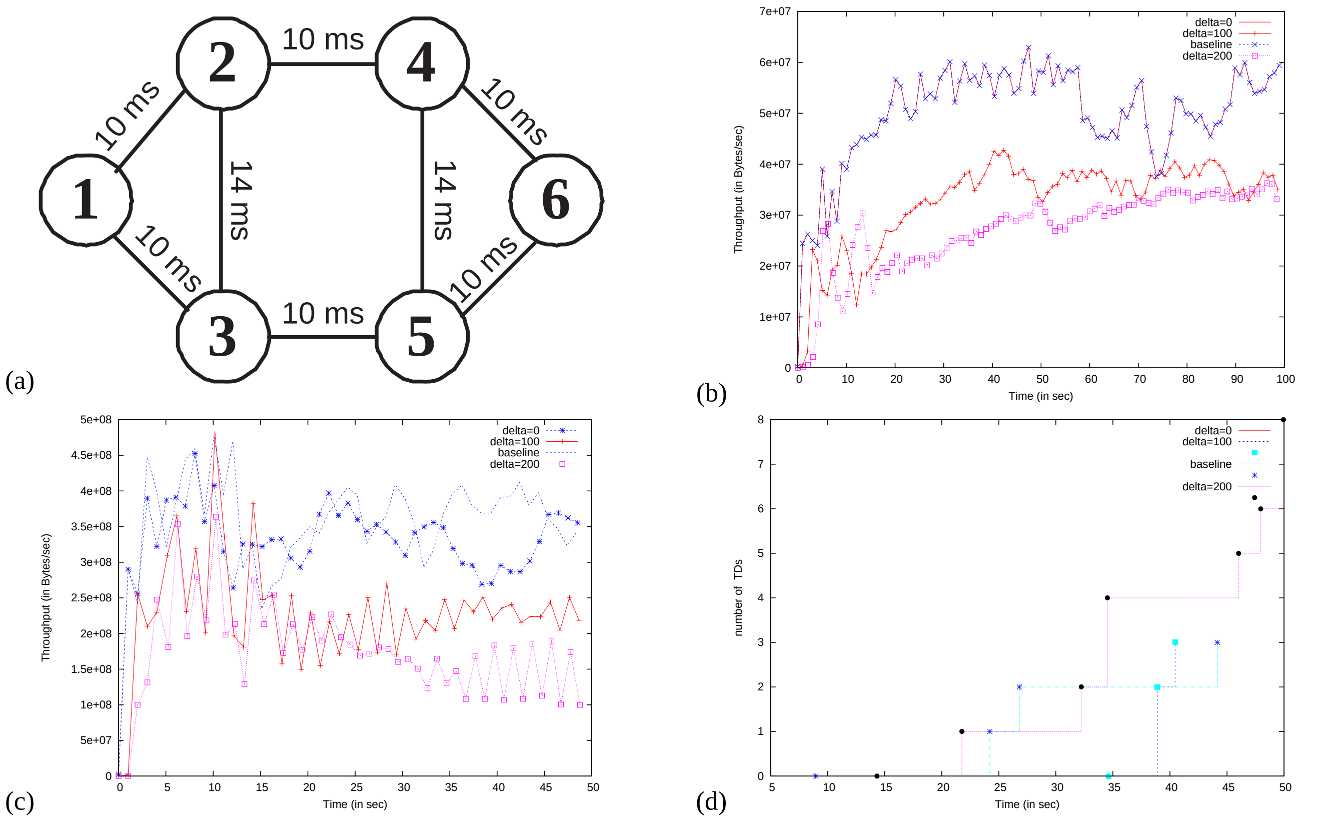

Fig. 2. (a) Simulation network. (b) On-line TCP throughput vs. simulation time for a pure TCP traffic network. (c) On-line TCP throughput vs. simulation time for a network with 100 TCP flows from $N_1$ to $N_6$ and all other flows running UDP traffic at 2 Mb/s. (d) Number of cumulative triple duplicates (TDs) vs. simulation time for the same network set up as Fig 2(c).

Figure 2(b) plots the on-line TCP throughput for a single flow (Flow 1 out of 100 flows) from $N_1$ to $N_6$ over a simulation period of 100 s for a purely TCP traffic network. We observe that the TCP throughput with source ordering can achieve up to **three times** the TCP throughput without source ordering (refer Fig. 2(b) $\delta = 0$ and $\delta = 200$ ms at time = 22 sec) for the scenarios considered in our simulation. Fig. 2(c) plots the on-line TCP throughput versus simulation time for a network with 100 TCP flows from $N_1$ to $N_6$ and all other flows running UDP traffic at 2 Mb/s. We again observe that the TCP throughput with source ordering can achieve up to **four times** the TCP throughput without source ordering (refer Fig. 2(c) $\delta = 0$ and $\delta = 200$ ms at time = 47 sec) for the scenarios considered in our simulation. Also the TCP throughput of without load-balancing *baseline* and with load-balancing and source-ordering ($\delta = 0$) is similar. We strongly believe that as the path delay-differential increases, the TCP throughput will continue to drop. Fig 2(d) supports our claim by plotting number the cumulative TDs versus time for the different scenarios. We observe that there are no TDs for the case of $\delta$ = 0 ms.

## V. Conclusion

In this paper, we have identified the ill-effects of OBS-layer load-balanced routing on higher-layer TCP performance. Through extensive simulations it is clear that the value of the path delay-differential has a significant impact on the higher-layer TCP performance. We propose a simple source-ordering approach that maintains the order of the bursts using electronic buffers at the ingress OBS edge node, so as to minimize the number of false timeouts and false triple duplicates. We observe that source-ordering can improve the TCP throughput by up to three-four times.

## References

[1] C. Qiao and M. Yoo, "Optical Burst Switching (OBS) - A New Paradigm for an Optical Internet," *JHSN*, Jan. 1999.
[2] G. Thodime et al., "Dynamic Congestion-Based Load Balanced Routing in Optical Burst-Switched Networks," *IEEE GLOBECOM 2003.*
[3] V. Jacobson, "Congestion Avoidance and Control," *ACM SIGCOMM*, 1988.
[4] X. Yu, C. Qiao, and Y. Liu, "TCP Implementation and False Time Out Detection in OBS Networks," *IEEE INFOCOM 2004.*