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

Source-Ordering for Improved TCP Performance over Load-Balanced Optical Burst-Switched (OBS) Networks

Bharat Komatireddy · Neal Charbonneau · Vinod M. Vokkarane

Received: Friday, May 29, 2009/ Accepted:

Abstract Recent advances in optical switching technology allows for the creation of networks in which data bursts are switched optically at each node, offering a greater degree of flexibility suitable for handling bursty Internet traffic. TCP-based applications account for a majority of data traffic in the Internet; thus understanding and improving the performance of TCP implementations over OBS networks is critical. Previously, several articles show that load-balanced routing improves loss-performance in OBS. In this paper, we identify the ill-effects of load-balanced OBS on TCP performance caused by false time-outs and false fast-retransmits. We propose a *source-ordering* mechanism that significantly improves TCP throughput over a load-balanced OBS network.

Keywords Load-balancing · TCP · OBS

1 Introduction

Next-generation high-speed optical Internet will be required to support a broad range of emerging applications which may not only require significant bandwidth, but may also have strict requirements with respect to end-to-end delays and reliability of transmitted data.

In optical burst switching (OBS), data to be transmitted is assembled in to bursts and are switched through the network all optically [1]. 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) [2].

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 [3], wavelength conversion [4,5], and deflection routing [6]. These contention resolution techniques are reactive in nature, 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.

Load-balanced routing is an approach to implement contention avoidance in OBS [7]. Load-balanced routing involves two stages, *route calculation* and *route selection*. Both route calculation and route selection can be implemented in a static or a dynamic manner. In this paper, we adopt the a load-balanced routing tech-

---

Portions of this paper were published in IEEE/OSA OFC/NFOEC 2007 and IEEE BroadNets 2007. This work was supported in part by NSF Grant CNS-0626798.

Department of Computer and Information
University of Massachusetts, Dartmouth, MA 02747
E-mail: vvokkarane@ieee.org

nique with static route-calculation and dynamic route-selection as proposed in [7]. At every τ seconds, all the ingress OBS node dynamically selects the least-congested path (among the two static link-disjoint minimum hop paths) to all their destination nodes using the cumulative congestion information of all the links along the two pre-calculated paths. A link is said to be congested, if offered load on Link (i, j), $L_{i,j} \geq P_{max}$, where P_{max} is the maximum load threshold on a link. Let τ_s and τ_d be the duration of successful burst arrivals and dropped burst arrivals during the interval τ, respectively. The offered load on each of the node's outgoing link is expressed as the duration of all arriving bursts over the interval τ, is given by, $L_{i,j} = \frac{\tau_s+\tau_d}{\tau}$.

Load-balanced routing is possible since we implement source-routing. With source-routing, the ingress node specifies the route that a burst will take. The intermediate core nodes simply read the route information from the BHP and forward the burst to the next hop specified. Source-routing requires knowledge of the network topology. OBS networks may be implemented in long haul or metro area networks so knowledge of the entire network is possible. The dissemination of network load on all the links can be done through a routing protocol using link state updates like OSPF. Again, OBS networks will be limited in size so link state updates will not cause congestion on the control channels. Based on these load updates, the source can change the path that a burst will take by specifying the path in the BHP.

There is a tremendous need to support reliable connection oriented end-to-end transport service for supporting new applications, such as the Grid systems. In the recent years, transmission control protocol (TCP)-based applications, such as Web (HTTP), Email (SMTP), peer-to-peer file sharing [8,9], and grid computing [10], account for a majority of data traffic in the Internet; thus understanding and improving the performance of TCP implementations over OBS networks is critical. One problem that arises when TCP traffic traverses over OBS networks is that the random burst loss due to contention may be falsely interpreted as network congestion by the TCP layer. We will discuss this in detail in Section 2.

While load-balanced routing can reduce the number of random contentions, it can also lead to reordering at the TCP layer, which can degrade TCP performance. In this paper, we propose a *source ordering* mechanism, that aims to neutralize the negative impact of the delay-differential between multiple transmission paths in the OBS network on higher-layer TCP performance so the benefits of load-balancing can still be obtained. The remainder of the paper is organized as follows. Section 2 will provide background for TCP and TCP over OBS. Section 3 discusses the issue of supporting TCP over an independently load-balanced OBS network. Section 4 describes the proposed *source ordering* mechanism in order to improve TCP performance over a load-balanced OBS network. Section 5 discusses the simulations results and Section 6 concludes the paper.

2 Background

In this section we will provide some background on TCP and the issues with TCP over OBS networks. We will discuss further issues with TCP over load-balanced OBS networks in Section 3.

The TCP flavors we will evaluate are High Speed TCP with SACK option [11,12] (we will refer to this as HS-TCP-SACK), TCP FAST [13], and TCP CUBIC [14]. The fundamental assumption of all these TCP flavors is that the underlying medium is electronic in nature, and that the packets experience queueing (buffering) delays during congestion in the electronic IP routers along the path of the TCP flow.

TCP flavors primarily differ in their implementation of congestion control mechanisms. TCP and its various flavors can be classified into three categories based on congestion control mechanisms, they are loss-based, delay-based, and rate-based. HS-TCP-SACK and CUBIC are loss-based congestion-control techniques that use packet losses to estimate the available bandwidth in networks. TCP SACK is a widely deployed TCP version in the Internet. HS-TCP-SACK and CUBIC employ loss-based congestion-control using *time-out* (TO) and *fast-retransmit* (FR) based mechanisms [15].

On the other hand, delay-based TCP flavors, such as TCP FAST, use delay measurements to estimate available bandwidth in the network. The queueing delay measured in TCP can provide information about the degree of network congestion, which will make TCP implementation easier to stabilize a network with a target fairness and high utilization.

HS-TCP-SACK, CUBIC, and FAST were designed for high speed networks so they can take advantage of large amounts of bandwidth. HS-TCP-SACK modifies TCP SACK's window increase and decrease algorithm. In traditional TCP, after a loss detection by triple duplicates the congestion window is halved. While in congestion avoidance, the congestion window is increased by one per RTT (assuming a window can be sent and ACKed in one RTT). This behavior causes TCP to perform poorly in networks with large bandwidth because a single loss cuts the window drastically and it takes many RTTs to recover. With HS-TCP-SACK, both the window increment in congestion avoidance and the window decrement after triple duplicates is a function of the current window size. When the window gets larger, the increases are larger and the decreases are smaller. This allows HS-TCP to utilize large amounts of bandwidth.

CUBIC is a more drastic change to TCP Reno. Instead of Reno's additive increase and decrease, CUBIC uses a cubic function to control the congestion window. The concave and convex portions of the cubic function allow CUBIC to quickly reach a steady state in the network and then to probe for available bandwidth. CUBIC also provides better fairness than HS-TCP since the growth of the congestion window is based on the time since the last congestion event instead of RTT as in traditional TCP flavors.

FAST is based on the same concept as TCP Vegas [16]. FAST uses queueing delays as a indication that the network is congested instead of loss events like traditional TCP. FAST measures the RTT and uses this to estimate the number of packets queued in the network. FAST tries to keep a constant number of packets in the network, so if the estimated number is smaller than this amount, FAST increases its send rate. It also decreases its send rate when the number of estimated packets in the network is too high. FAST uses larger increments and decrements depending on the estimated number of packets in the network than TCP Vegas. All three of these TCP flavors require only sender side modifications.

Using TCP over OBS networks results in poor performance. Due to the bufferless nature of OBS core network and the one-way based signaling scheme, the 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. 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, leading to false congestion detection. This false congestion detection is referred to as a *false time-out* (FTO) [17]. 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. A burst loss will be interpreted as light network congestion and will trigger one or more TCP-layer fast retransmissions.

3 TCP over Load-Balanced OBS

Static load-balanced routing techniques uses two fixed paths to transmit data between each source-destination pair, a primary path and an alternate path. The alternate typically being longer than and link-disjoint from the primary. 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 (δ) encountered may cause out-of-order reception of TCP segments (IP packets) at the destination, resulting in FTOs and FFRs.

Consider the following illustration scenario to better understand the issue of FTOs and FFRs due to load-balanced routing in OBS networks. In Fig. 5(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. 5(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 the 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] to the TCP sender. This results in FFRs at the TCP sender. Note that if the path delay-differential is significant, TCP sender may experience FTOs.

4 Source Ordering

In order to neutralize the negative impact of the path delay-differential caused by load-balanced routing in OBS, we propose *source ordering.* In OBS, all the ingress nodes implement source routing to transmit bursts to their corresponding destinations. In source ordering, the ingress node pre-calculates the path delay-differential between the primary minimum-hop path and the alternate second minimum-hop path, $\delta = \mid P_1 - P_2 \mid$, where P_1 is the end-to-end delay on the primary path and P_2 is the end-to-end delay on the alternate path.

We observe that every time the ingress node performs a path-switch from the longer alternate path to the shorter primary path, some of the bursts transmitted on the primary path may overtake the previously transmitted bursts on the longer alternate path before reaching the destination. Every time we perform a long-to-short path-switch, this scenario is quite common especially when the δ value is large. This differential in

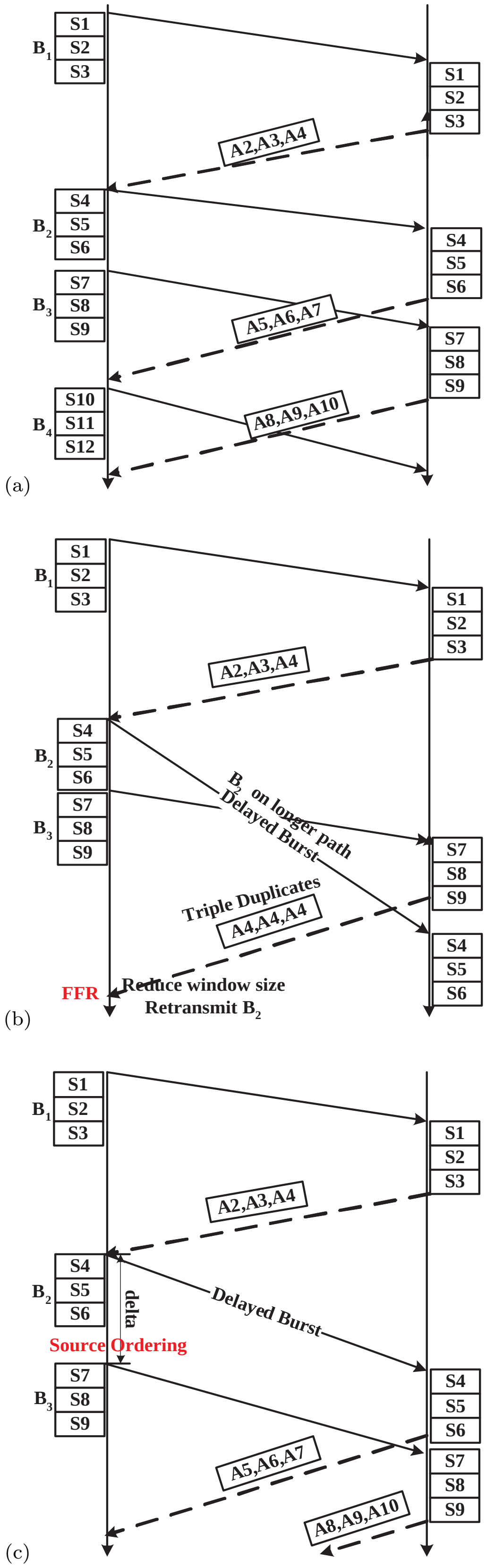


Fig. 1 (a) TCP-over-OBS with fixed-routing, (b) FFR in TCP over load-balanced OBS, and (c) source-ordering to minimize FFR (and FTO) in TCP over load-balanced OBS.

path-delay can result in FFRs and possibly FTOs (refer Fig. 1(b)). In source ordering, every time a long-to-short path-switch occurs, we electronically buffer the bursts for δ seconds before we start transmitting on the shorter path.

In Fig. 1(c), every time a long-to-short path-switch occurs we delay the burst for the amount of time equivalent to the path delay-differential of the two paths, using electronic buffering at the ingress OBS node. Note that the ingress node is aware of the path delay-differential since OBS implements source-routing.

Implementation of the source-ordering mechanism can be done entirely at the ingress node. We assume static route calculation, so each ingress node knows the primary and alternate paths for a given egress node to use for load-balancing. Only burst scheduling needs to be modified at the ingress. When a path-switch occurs from the longer path to the shorter path, the scheduler will have to delay the next burst to be sent, long enough so that it would not reach the egress before the last burst sent on the longer path. Implementation of source ordering does not need to take into account individual flows, it only needs to ensure that bursts (destined to the same egress) are delivered in order.

Another possibility to overcome the issues with reordering is to implement destination ordering, where the egress reorders bursts instead of the ingress. The issue with this approach is that the destination does not know when a path switch has occurred so it will not know if a burst has arrived out-of-order because of a path switch or because of a contention. This will result in longer recovery times in a case of contention since the egress will have to wait to see if the expected burst will arrive before deburstification the out-of-order burst. The egress would also need some mechanism to determine if an arriving burst is in order or not, like sequence numbers for bursts. Destination ordering as highlighted above leads to several complications; we restrict the evaluation of source ordering in the paper.

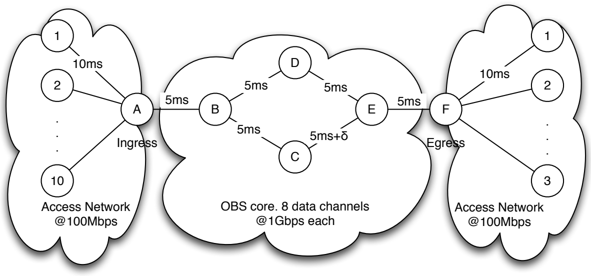


Fig. 2 Simulation Topology.

5 Simulation Results

In this section we will discuss simulations results obtained from ns2 with the OWns module [18] for simulating OBS networks. We evaluate source ordering over a load-balanced OBS network under a number of different scenarios and then compare source ordering to regular TCP over load-balanced OBS. The load-balanced routing uses two fixed paths and the least-congested path is dynamically chosen. First, we vary the delay differential between the primary and alternate path. Next we evaluate source ordering with different burst sizes. After that we examine the impact of loss on source ordering. The impact of the load balancing parameter ρ is also investigated.

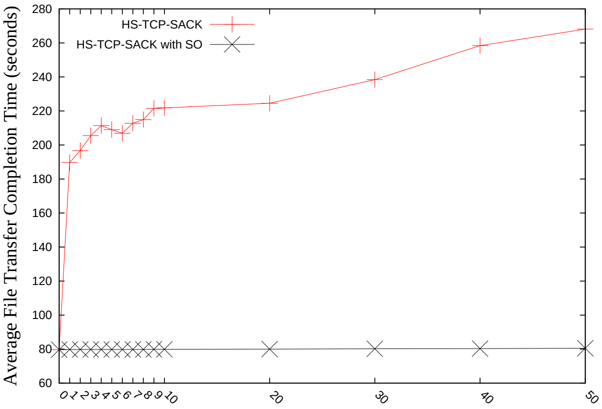
The topology used in the simulations is shown in Fig. 2. We have an access network consisting of 10 nodes connected to the OBS ingress node. Each access node has a TCP flow to the corresponding nodes on the right side. The electronic nodes are numbered while the OBS nodes use letters. The primary path in the network is A-B-D-E-F while the alternate path is A-B-C-E-F.

The TCP flows use the High Speed TCP window increase and decrease functionality [12] with SACK. Each flow sends a 1GB file using FTP. The network uses load balancing between the two paths in the core. The τ parameter is set to 500ms and ρ_{max} is set to 5%. This means that every 500ms the path will change if the congestion on the current path exceeds 5%. The other parameters are as follows: the max burst size is 100KB, the burst assembly timer (BAT) is 10ms, the delay differential, δ, is 5ms, and there is no loss. Each of these parameters (except BAT) will be varied in the following subsections while analyzing the performance of source ordering. In our graphs, we use "with SO" labels to mean with source ordering.

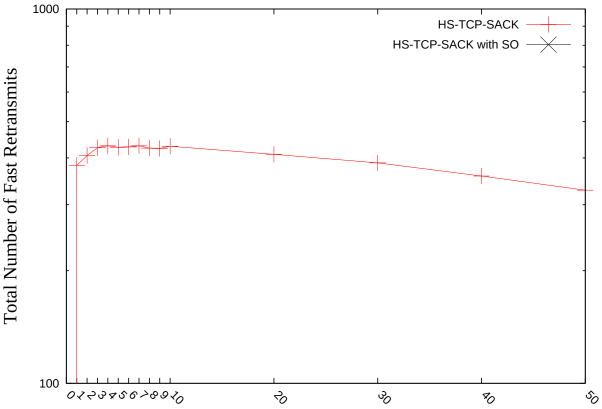
5.1 Performance with Varying Delay Differential

In the first set of simulations we vary the delay differential between the primary and alternate path. There is no intentional loss for these simulations. The results are shown in Fig. 3.

From Fig. 3(a) we observe that even a difference of 1ms in the alternate paths results in reordering. Source ordering is able to adjust since it reorder the burst at the ingress but HS-TCP-SACK experiences false fast retransmissions whenever reordering occurs, resulting in much higher completion times. Fig. 3(b) shows that each of the 10 flows with source ordering does not experience even a single false fast retransmit, while HS-TCP-SACK flows without source ordering experience false fast retransmits repeatedly. In this case, we observe up to an 300% improvement in average completion time for a 1GB file.



(a) Average flow completion time.



(b) Total number of fast retransmits across all flows.

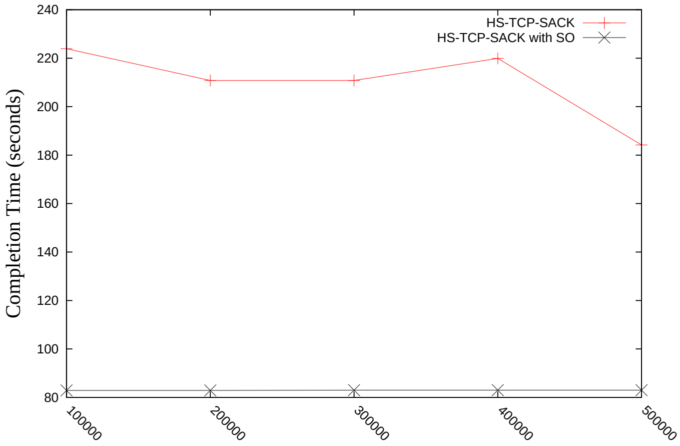
Fig. 3 Comparison of performance of FTP file transfers, each of the 10 flows sending a 1GB file, with varying delay differential.

5.2 Performance with Varying Burst Size

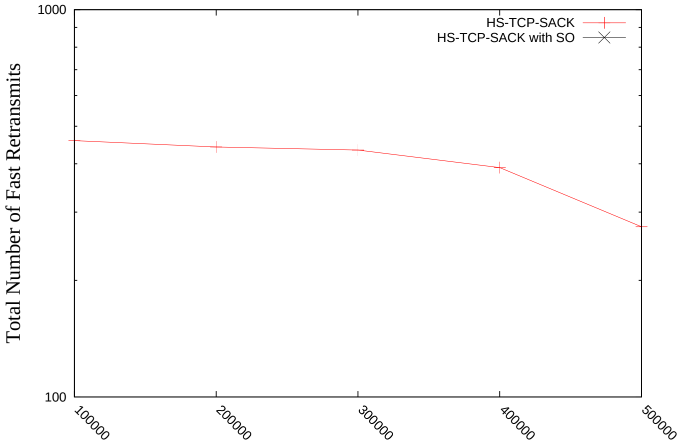
In this section, we briefly analyze the affects of burst size on reordering. In Fig. 4(a) we plot the completion time while varying the maximum burst size. The burst size has little affect with source ordering approach but does have an affect on HS-TCP-SACK without source reordering. From Fig. 4(b), there is a decrease in the number of false fast retransmits experienced by regular HS-TCP-SACK as the burst size increases. This is simply because as the bursts get bigger, more data is able to be sent on the same path instead of getting split onto different paths.

5.3 Performance with Varying Loss Levels

We analyze the affects of random contentions in the OBS core on TCP's performance. Fig. 5(a) shows the average completion time for HS-TCP-SACK with and without source ordering. The delay differential, δ, is set

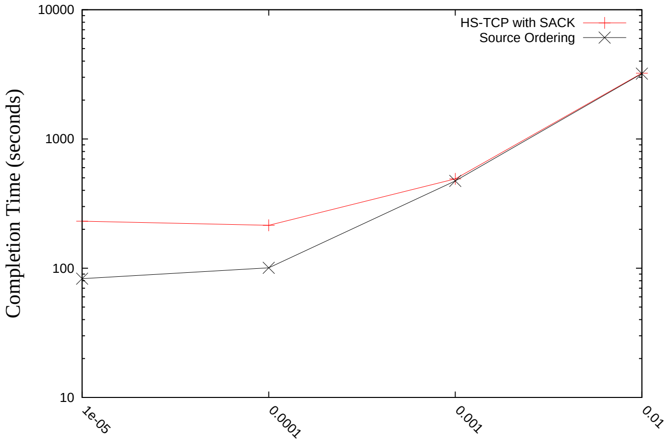


(a) Average flow completion time.

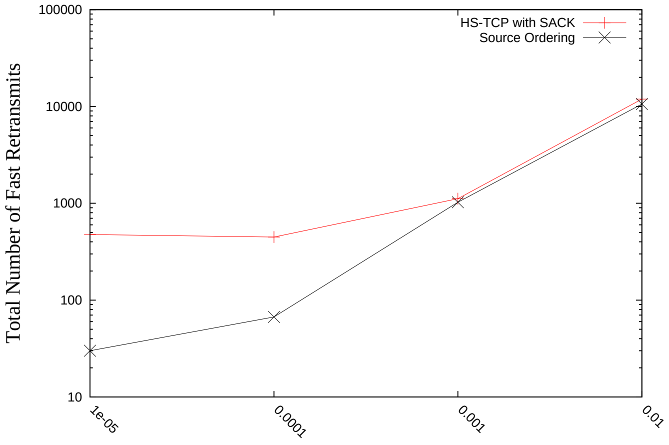


(b) Total number of fast retransmits across all flows.

Fig. 4 Comparison of performance of FTP file transfers, each of the 10 flows sending a 1GB file, with varying max burst sizes.



(a) Average flow completion time.



(b) Total number of fast retransmits across all flows.

Fig. 5 Comparison of performance of FTP file transfers, each of the 10 flows sending a 1GB file, with random contentions.

to 50ms. We can observe that for low loss probability there is a significant increase in performance, up to 300%, but as loss probability increases, there is little gain. This is due to the fact that real loss results in lower TCP send rate, this leads to lower load in the core. Lower TCP arrival rate may not trigger load-balanced routing since we have only TCP-based traffic in the network.

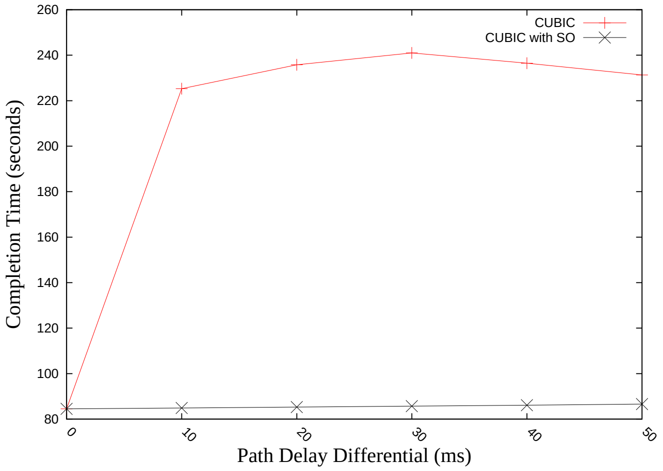
The interesting point on this graph is the performance at 0.001 loss probability. This is the last point where there is reordering in the network, at higher loss probabilities there is not enough TCP traffic to cause reordering. There is a 60s difference in completion time, or about a 6% improvement.

5.4 Performance of CUBIC and FAST

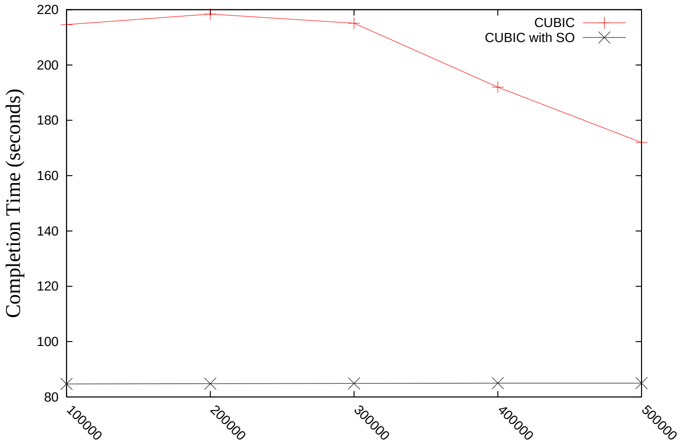
In this section we run simulations using CUBIC [14] and FAST [13]. Both are designed for high-speed networks. CUBIC uses a cubic function to determine the sender's congestion window growth, which allows it to ramp up quickly to reach a stable state and then again to probe for available bandwidth. FAST is a delay-based TCP similar to TCP Vegas but modified for high-speed networks. For FAST, we use the default ns2 configuration parameters with $\alpha = 100$, $\beta = 100$, and $\gamma = 0.5$.

We first present the results for CUBIC. Fig 6 plots the completion time for varying δ values, burst sizes, and loss levels as was done for HS-TCP-SACK. Comparing Fig. 6(a) with Fig. 3(a) shows that CUBIC has very similar performance compared to HS-TCP-SACK with the exception of the continued increase up to $\delta = 50$ that HS-TCP-SACK experiences without source ordering. Similarly, comparing Fig. 4(a) to Fig. 6(b) and Fig. 5(a) to Fig. 6(c) shows that CUBIC and HS-TCP-SACK perform similarly in situations for varying burst sizes and loss.

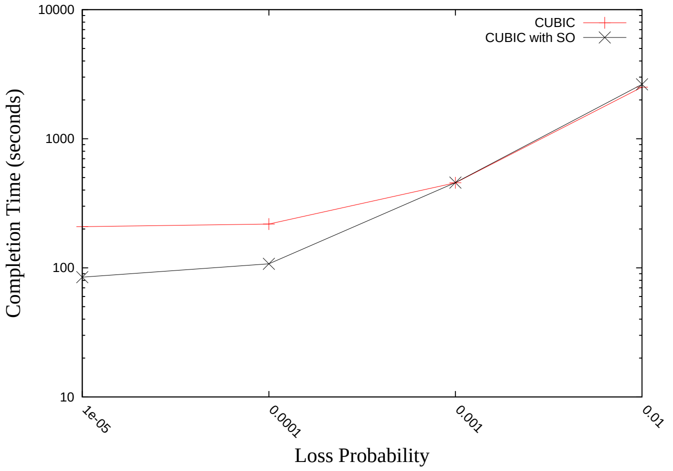
Fig. 7 shows our results for FAST. Fig. 7(a) shows FAST with varying δ values. FAST with source ordering is about 20s slower than HS-TCP-SACK and CUBIC,



(a) Average flow completion time vs. δ for CUBIC.

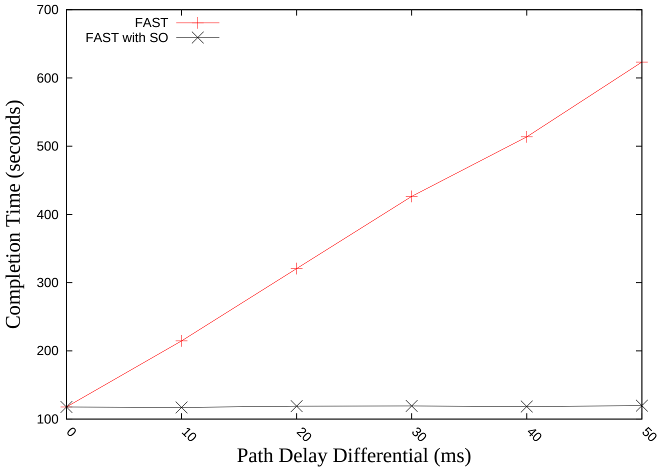


(b) Average flow completion time vs. maximum burst size for CUBIC

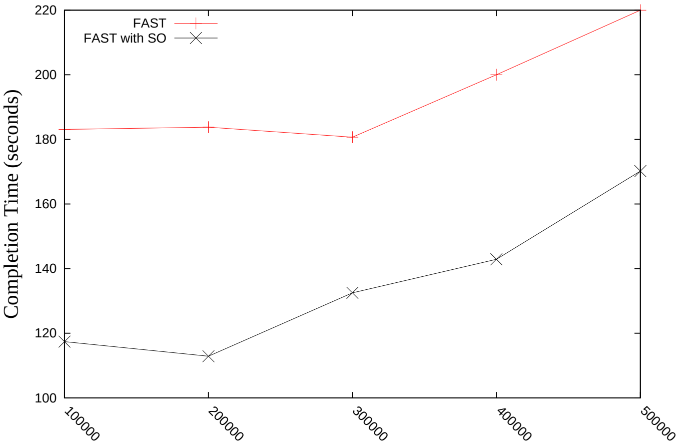


(c) Average flow completion time vs. loss for CUBIC.

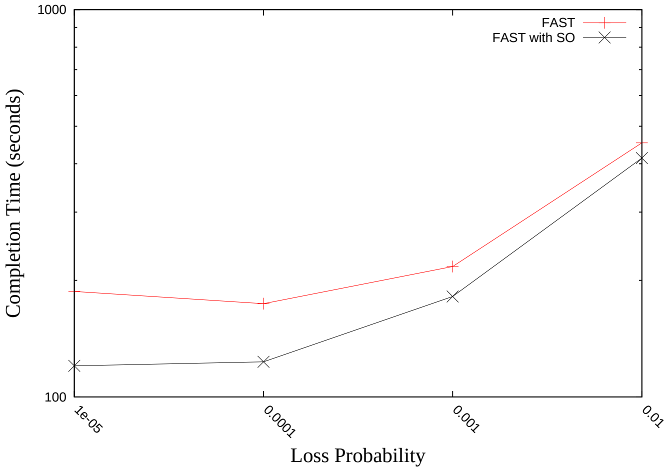
Fig. 6 Comparison of performance of FTP file transfers using TCP CUBIC.



(a) Average flow completion time vs. δ for FAST.



(b) Average flow completion time vs. maximum burst size for FAST.



(c) Average flow completion time vs. loss for FAST.

Fig. 7 Comparison of performance of FTP file transfers using TCP FAST.

while FAST without source ordering performs worse than HS-TCP-SACK, especially at higher δ values. At $\delta = 50ms$ FAST takes three times longer to finish than HS-TCP-SACK. Fig. 7(b) compares FAST with varying burst sizes. For HS-TCP-SACK and CUBIC, larger burst sizes led to better performance when source ordering was not used. This is not the case for FAST. Both with and without source ordering have worse performance as burst size increases. Lastly, Fig. 7(c) compares FAST's performance with varying loss. FAST has better completion time at higher loss than HS-TCP-SACK or CUBIC and even at high loss rates FAST with source ordering still outperforms FAST without source ordering. This behavior can be explained by the fact that FAST was not designed to work on the bufferless OBS networks. FAST uses estimated RTT to determine its send rate. When router buffers begin to overflow, the RTT increases and FAST lowers its send rate. Since there are no buffers in OBS, FAST does not work as it was designed to. As the maximum burst size and path delay differential increase the RTT also increases, which FAST interprets as congestion, leading to poor performance.

6 Conclusion

In this paper, we have evaluated the performance of different TCP flavors, such as FAST, HS-TCP-SACK, and CUBIC over a load-balanced OBS. In load-balanced routing, two routes are first calculated statically and the least-congested route is selected dynamically for data transmission. We identify 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 OBS ingress node, so as to minimize the number of false time-outs and false fast-retransmit. We observe that source-ordering can improve the TCP throughput by up to 400%.

An important area of future work is to implement load-balanced routing with Reordering Robust (RR-TCP) [19] in order to avoid false fast retransmits and false time-outs. Another area of future work is to implement TCP over OBS with burst segmentation [20]. Burst segmentation will increase the probability of a burst reaching the destination, leading to reduction of false fast-retransmits (and false time-outs). This can also have a significant positive impact on the TCP-over-OBS performance.

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, January 1999.
3. I. Chlamtac, A. Fumagalli, L. G. Kazovsky, and et al., "CORD: Contention resolution by delay lines," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 5, pp. 1014–1029, June 1996.
4. B. Ramamurthy and B. Mukherjee, "Wavelength conversion in WDM networking," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 1061–1073, September 1998.
5. A. Bononi, G. A. Castanon, and O. K. Tonguz, "Analysis of hot-potato optical networks with wavelength conversion," *IEEE/OSA Journal of Lightwave Technology*, vol. 17, no. 4, pp. 525–534, April 1999.
6. S. Yao, B. Mukherjee, S. J. B. Yoo, and S. Dixit, "A unified study of contention-resolution schemes in optical packet-switched networks," *IEEE/OSA Journal of Lightwave Technology*, March 2003.
7. G.P.V. Thodime, V. M. Vokkarane, and J. P. Jue, "Dynamic congestion-based load balanced routing in optical burst-switched networks," in *Proceedings, IEEE Globecom*, December 2003, vol. 5, pp. 2694–2698.
8. I. Stoica and et. al., "Chord: A scalable peer-to-peer lookup protocol for internet applications," in *Proceedings, ACM SIGCOMM*, 2001.
9. 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," in *Proceedings, ACM SIGMETRICS*, 2003.
10. 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, pp. 200–222, 2004.
11. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," *RFC 2018*, 1996.
12. S. Floyd, "Highspeed TCP for large congestion windows," *RFC 3649*, December 2003.
13. D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246–1259, Dec. 2006.
14. S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *SIGOPS Operating Systems Review*, vol. 42, no. 5, 2008.
15. V. Jacobson, "Congestion avoidance and control," in *Proceedings, ACM SIGCOMM*, 1989.
16. L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP vegas: new techniques for congestion detection and avoidance," *SIGCOMM Computer Communication. Review*, vol. 24, no. 4, pp. 24–35, 1994.
17. X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false time out detection in OBS networks," in *Proceedings, IEEE Infocom*, March 2004.
18. "OBS-NS simulator: http://wine.icu.ac.kr/õbsns/index.php," .
19. M. Zhang, B. Karp, S. Floyd, and L. Peterson, "RR-TCP: a reordering-robust TCP with DSACK," in Proceedings, IEEE ICNP, Nov. 2003, pp. 95–106.
20. V. M. Vokkarane and J. P. Jue, "Burst segmentation: An approach for reducing packet loss in optical burst switched networks," *SPIE Optical Networks Magazine*, vol. 4, no. 6, pp. 81–89, November-December 2003.