# TCP Over Optical Burst Switching (OBS): To Split or Not To Split?

Deepak Padmanabhan, Rajesh Bikram, and Vinod M. Vokkarane
Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA, USA
E-mail: deepakp1982@gmail.com and {rrc,vvokkarane}@umassd.edu

*Abstract*— **TCP-based applications account for a majority of data traffic in the Internet; thus understanding and improving the performance of TCP over OBS network is critical. In this paper, we identify the ill-effects of implementing TCP over a hybrid network (IP-access and OBS-core). We purpose a Split-TCP approach for a hybrid IP-OBS network to improve TCP performance. We propose two Split-TCP approaches, namely, *1:1:1* and *N:1:N*. We evaluate the performance of the proposed approaches over an IP-OBS hybrid network. Based on the simulation results, *N:1:N* Split-TCP approach outperforms all other approaches.** [1]

**Keywords: TCP, IP, WDM, and OBS.**

## I. Introduction

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

In optical burst switching (OBS), data to be transmitted is assembled into bursts and is switched through the network 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 burst's route. In OBS networks, apart from the data channels, each link has one or more control channels to transmit BHPs. BHPs carry information about the corresponding burst such as source, destination, burst duration, and offset time. Offset time is the time by which the burst and BHP are separated at the source and at subsequent intermediate nodes. The offset time allows BHP to be electronically processed ahead of time at each intermediate node before the corresponding burst arrives. Just-enough-time (JET) is one such one-way based OBS signaling technique [2].

The primary issue in the OBS core network is contention resolution, since core nodes do not have buffers. Contention occurs when two or more bursts contend for the same output port at the same time. There are several contention resolution (or loss minimization) techniques, such as optical buffering [3], wavelength conversion [4], deflection routing [5], and segmentation [6]. These loss minimization techniques are reactive in nature since they try to resolve the contention when it occurs. An alternative to loss minimization is to implement loss recovery techniques, such as cloning [7] and retransmission [8].

There is a tremendous need to support reliable connection-oriented end-to-end transport service for supporting new applications, such as Grid computing [9]. In the recent years, TCP-based applications, such as Web (HTTP), Email (SMTP), and peer-to-peer (P2P) file sharing [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. The popular TCP flavors are TCP Tahoe [11], TCP Reno [12], [13], TCP New-Reno [14], and TCP SACK [15]. The fundamental assumption in all these TCP flavors is that the underlying medium is electronic and that the packets experience queueing (buffering) delay due to congestion at the IP routers. Over the years, TCP has undergone significant changes in terms of congestion control mechanisms and handling issues concerning the need for high-bandwidth at the presence of high end-to-end delay between the sender and the receiver [16]. Primarily most of the TCP flavors differ in their implementation of congestion-control protocol. TCP SACK uses a packet loss as a congestion-control technique to estimate the available bandwidth in the network. TCP SACK implements congestion-control using *time-out* (TO) and *fast-retransmit* (FR) mechanisms [15]. In this paper, we use TCP SACK for all our simulations.

In OBS, due to the bufferless nature of the core network and the one-way JET signaling, the 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 sender's window is dropped due to contention at a low traffic load, then the TCP sender times out, leading to false congestion detection that is referred to as a *false time-out* (FTO) [17]. When the TCP sender detects this (false) congestion, it will trigger *slow start*, resulting in reduced TCP throughput. Another example is when a random burst loss triggers TCP fast retransmission for the case in which segments in a TCP sender's window are assembled into multiple bursts. The random burst loss in OBS will be interpreted as light congestion, leading to one or more TCP-layer *false fast retransmission* (FFR). Recently, few works have evaluated TCP throughput over an OBS network [18], [19], [20]. However, these works assume a constant random burst loss probability in the OBS network, and do not take into account TCP false congestion detection.

A similar problem is observed while using TCP connection over a wired-cum-wireless network. A Split-TCP approach was proposed [21] and significant improvement in the overall end-to-end throughput was observed. There have also been several research works for improving TCP throughput over wireless networks using forward error correction (FEC) and retransmissions (ARQ) [22], [23]. TCP performance degrades when end-to-end connections extend over wireless connections as the TCP sender assumes random wireless losses to be congestion losses resulting in unnecessary congestion control actions. The authors in [23] propose and evaluate the performance of wireless-aware TCP under different settings. There have been several works on split-connection based TCP versions, such as indirect TCP [24], selective repeat protocol (SRP) [25], mobile-TCP [26], and mobile-end transport protocol [27]. End-to-end TCP flow is split into multiple TCP flow-segments and each flow segment is managed in coordination to the adjoining segment. There are other novel mechanisms

[1]This work was supported in a part by the National Science Foundation (NSF) under Grant CNS-0626798.

proposed to improve Wireless-TCP performance, such as using parallel TCP flows [28], [29] and using explicit congestion notification [30]. In Split-TCP approach for wireless networks, the base station works as a proxy between sender and the mobile host. Retransmission of lost packets is done by the base station that contains the traces of every TCP flow passing through it.

In this paper, we evaluate the concept of Split-TCP over a hybrid IP-OBS network. Though the issues in IP-OBS network are very different from wired-cum-wired networks, we believe that with certain modifications Split-TCP approach over IP-OBS network may yield significant performance benefits. The remainder of the paper is organized as follows. Section II describes the proposed Split-TCP approach in order to improve TCP performance over the hybrid (electro-optical) network. Section III discusses the issue of signaling of TCP over a hybrid network (IP-OBS). Section IV discusses the simulations results, and Section V concludes the paper and discusses potential areas of future work.

## II. Split-TCP Approach

The next-generation optical Internet will be comprised of hybrid networks, i.e., a combination of IP-access networks and OBS-core networks. In a typical hybrid network, two important phenomenon are observed:

1 IP-access network is the bottleneck link for the end-to-end TCP flow resulting in restricting the end-to-end throughput. Also, we know that as the round-trip delay between the sender-receiver pair increases, TCP throughput decreases.
2 When data is transmitted all-optically over the OBS core network, packet loss is primarily due to random burst contentions and not due to router buffer overflows (as is the case of IP-network). In the event of a random loss, the TCP sender at the end-host reduces its send rate and starts its congestion control mechanism, even though packet loss was due to a random burst contention.

In order to solve the previously mentioned issues of TCP running over a hybrid network, we purpose employing a Split-TCP approach. In the Split-TCP approach (refer Fig. 1), a single end-to-end TCP flow is divided into three independent TCP flows. One from the sending host to the optical ingress node (over the ingress IP-access), another TCP connection over the optical core, and the last connection from the optical egress node to the original destination host (over the egress IP-access). By doing so, we can isolate burst contention losses over the OBS network from the IP-access networks. Also, we can implement different TCP flavors specific to each network segment that can help boost end-to-end throughput. By splitting an end-to-end TCP flow, we can isolate losses of IP networks from OBS networks.

In this paper we propose two Split-TCP approaches, namely, *1:1:1* and *N:1:N*. In the *1:1:1* Split-TCP approach, each end-to-end TCP flow that spans over the IP-OBS-IP network is split into three Split-TCP connections, source host to OBS ingress, OBS ingress to OBS egress, and OBS egress to destination host. There is a one-to-one mapping between the ingress IP-access TCP flow, the OBS-core TCP flow, and the egress IP-access TCP flow. In the *N:1:N* Split-TCP approach, each end-to-end TCP flow that spans over IP-OBS-IP network is also split into three Split-TCP connections, and there is a *N:1* mapping between the ingress IP-access TCP flows and the OBS-core TCP flow and a *1:N* mapping from the OBS-core TCP flow and the egress IP-access TCP flows. In order to facilitate this, *1:1:1* Split-TCP approach, uses a *non-persistent* TCP flow and *N:1:N* Split-TCP approach, uses a *persistent* TCP flow. In the *1:1:1* Split-TCP approach, a non-persistent OBS-core TCP flow will be setup and terminated for every IP-access TCP. While in the *N:1:N* Split-TCP approach, a persistent OBS-core TCP flow will be setup once and will continue until all the IP-access TCP flows have terminated. In the *N:1:N* Split-TCP architecture, instead of maintaining $N$ TCP flows over the OBS network, the split-agent keeps track of only one TCP flow between the OBS edge nodes.

Fig. 1 depicts the Split-TCP flow setup for both *1:1:1* and *N:1:N* approaches over a simple hybrid (IP-OBS-IP) network. In a conventional TCP scenario, we would have had ten end-to-end TCP flows, one each from Node 0 - Node 12 (TCP 0 flow), Node 1 - Node 13 (TCP 1 flow), and so on (refer Table I, Columns 1-2). We refer to the conventional TCP approach as Baseline-TCP and compare the performance with different Split-TCP approaches. Fig. 1(a) represents the *1:1:1* Split-TCP setup, we can see that the Baseline-TCP 0 flow is now split into TCP 0 - TCP 10 - TCP 20 flows, the Baseline-TCP 1 flow is split in to TCP 1 - TCP 11 - TCP 21 flows, and so on (refer Table I, Columns 3-4,6). Fig. 1(b) represents the *N:1:N* Split-TCP setup, we can see that the Baseline-TCP 0 flow is now split into TCP 0 - TCP 10 - TCP 20 flows, the Baseline-TCP 1 flow is split into TCP 1 - TCP 10 - TCP 21 flows, and so on (refer Table I, Columns 3, 5-6).

From Fig. 1, we can see that there are 30 *1:1:1* Split-TCP flows and 21 *N:1:N* Split-TCP flows representing the original 10 Baseline-TCP flows. The split-agents are responsible for ensuring the end-to-end reliable data transfer and managing the split connections. Split-agents (at Node 10 and Node 11), split one Baseline-TCP connection into three parts. The split-agents communicate with each other and manage the Split-TCP connections of a flow. Once the entire data transfer is completed by all the three *1:1:1* Split-TCP flows that represent an end-to-end Baseline-TCP flow, the agents terminate the Split-TCP flows. In the case of *N:1:N* Split-TCP flows, the split-agent will terminate the OBS-core TCP flow only if all the IP-access flows have terminated.

In order to implement the Split-TCP approach, the OBS edge nodes should act as an interface between the split connections. Each of the OBS edge nodes also need to track all the TCP connections and should have the ability to transfer data packets from one flow to another. This involves implementing an agent that handles packet transfer logic between the adjacent TCP flow segments. One of the fundamental advantageous of splitting a single long end-to-end TCP connection into multiple short TCP connections is to improve TCP throughput over the split connections due to lower roundtrip times (RTT). This results in significant improvement of overall end-to-end TCP throughput.

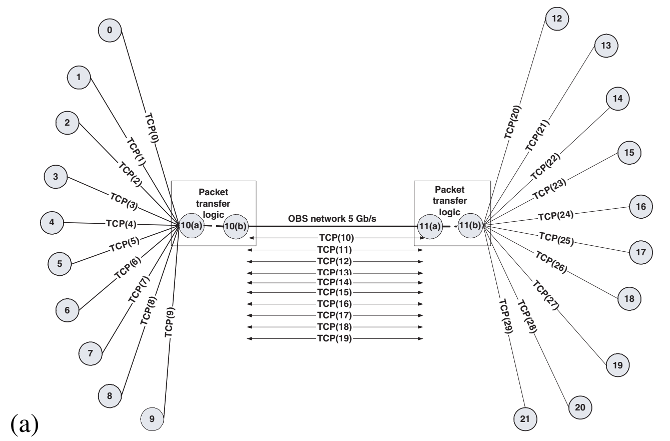
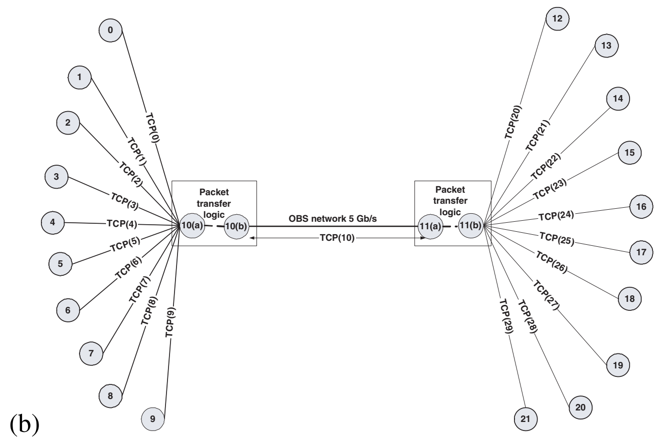



Fig. 1. Split-TCP Architecture with TCP flows (a) *1:1:1* and (b) *N:1:N*.

TABLE I
TCP FLOW SETUP: BASELINE, *1:1:1* SPLIT, AND *N:1:N* SPLIT

| TCP SRC-DST | Baseline | IN | 1:1:1 OBS | N:1:N OBS | OUT |
|---|---|---|---|---|---|
| N0-N12 | TCP 0 | TCP 0 | TCP 10 | TCP 10 | TCP 20 |
| N1-N13 | TCP 1 | TCP 1 | TCP 11 | TCP 10 | TCP 21 |
| N2-N14 | TCP 2 | TCP 2 | TCP 12 | TCP 10 | TCP 22 |
| N3-N15 | TCP 3 | TCP 3 | TCP 13 | TCP 10 | TCP 23 |
| N4-N16 | TCP 4 | TCP 4 | TCP 14 | TCP 10 | TCP 24 |
| N5-N17 | TCP 5 | TCP 5 | TCP 15 | TCP 10 | TCP 25 |
| N6-N18 | TCP 6 | TCP 6 | TCP 16 | TCP 10 | TCP 26 |
| N7-N19 | TCP 7 | TCP 7 | TCP 17 | TCP 10 | TCP 27 |
| N8-N20 | TCP 8 | TCP 8 | TCP 18 | TCP 10 | TCP 28 |
| N9-N21 | TCP 9 | TCP 9 | TCP 19 | TCP 10 | TCP 29 |

The advantage of implementing a *1:1:1* Split-TCP approach is that the split-agent's packet transfer function will be easier to implement, while in a *N:1:N* Split-TCP approach, the BHP may have to store additional information about the flow-mapping so as to facilitate the reverse mapping at the OBS egress node. On the other hand, *N:1:N* Split-TCP approach has the advantage of maintaining a single TCP flow for every pair of OBS-edge nodes, while *1:1:1* Split-TCP approach has to maintain several TCP flows (possibly thousands) leading to increased overhead at the OBS edge nodes. Also, *N:1:N* Split-TCP approach is expected to outperform all other approaches as it does not delay the data transfer due to the implementation of TCP three-way handshake over the OBS-core network.

## III. SPLIT-TCP SIGNALING

Coordination between TCP and OBS layers, if implemented properly, may optimize the network performance. Careful investigation of the tradeoffs of cross-layer optimization is a critical factor for the practical adoption of Split-TCP approach. One key design parameter to be considered is the additional signaling overhead necessary, and it has to be weighed with respect to the gain obtained due to coordination of the two layers.

The OBS ingress node acts as the proxy for the destination while communicating with the source and at the same time acts as a proxy for the sender while communicating to the destination. The proxy (edge nodes) sends the acknowledgement to the sender on behalf of destination host, this can be achieved through *acknowledgment spoofing*. Once acknowledged, the custody of the packet is transferred from the source host to the OBS ingress node. The OBS ingress node is now responsible for sending the packets to the destination. Each OBS edge node has to save all the traces of the TCP flows passing through it, as the node has to handle retransmissions and out-of-order packets. Isolation of network losses and recovery of packets for the *N:1:N* Split-TCP approach or the *1:1:1* Split-TCP approach increases the network throughput. OBS ingress and egress nodes maintain one receiver and one sender packet queue in the case of *1:1:1* Split-TCP approach. In the *N:1:N* Split-TCP approach, the OBS ingress node maintains $N$ receiver queues and one sender queue and the OBS egress node maintains $N$ sender queues and one receiver queue. These queues are responsible for isolating the problems in the IP-access from the OBS-core network and vice-versa.

The Split-TCP approach violates the end-to-end semantics of conventional TCP. Most applications are not sensitive to end-to-end semantics but some applications based on interactive communication do rely on the end-to-end flows. In this paper, we are evaluating the performance benefit of the Split-TCP approach. Implementation of end-to-end TCP semantics over the Split-TCP architecture is outside the scope of this paper. An interesting area of future work is to implement SNOOP [21] or Semi Split-TCP [31] approaches to address the issue of providing end-to-end semantics over a Split-TCP architecture. Termination of the Split-TCP flows are basically same as the conventional TCP termination described in RFC793.

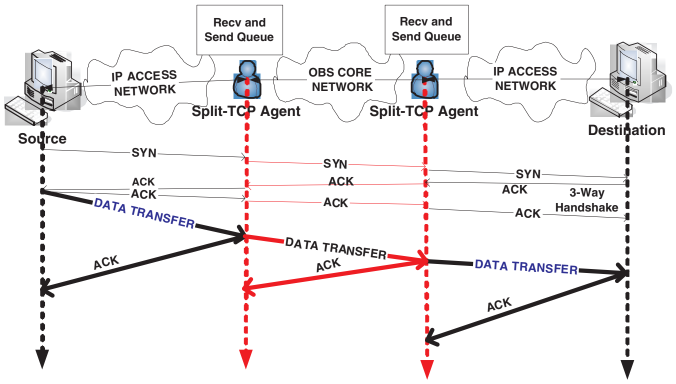


Fig. 2. Signaling framework for *1:1:1* Split-TCP approach.

In the *N:1:N* Split-TCP approach, the split-agent combines data from all ingress connections on to a single flow over OBS network. The egress split-agent routes the data packets to its intended destination. In the next section, we evaluate the performance of *1:1:1* and *N:1:N* Split-TCP approaches.

## IV. Simulation Results

All the simulations are performed on Network Simulator-2.28 (NS-2.28) [32] with OBS module (OBS-0.9a) [33]. The simulations were performed over the NSF network consists of 14 nodes (refer Fig. 3). The small box connected with dotted lines at Node 0 represents the ingress IP-access Split-TCP flows (IN) and at Node 13 represents egress IP-access Split-TCP flows (OUT). We have 10 TCP flows from Node 0 to Node 13 and a constant bit-rate traffic source (UDP) at 30 MB/s between each NSF source-destination pair. We have set a 2% packet loss probability at both the IN and OUT IP-access networks. The only loss in the OBS-core network is due to random data burst contentions. Each core link has 5 data channels at 1 Gb/s. Note that the Split-TCP approach can be applied to any TCP flavor. In this paper, we have used TCP SACK for all simulation-based comparisons.

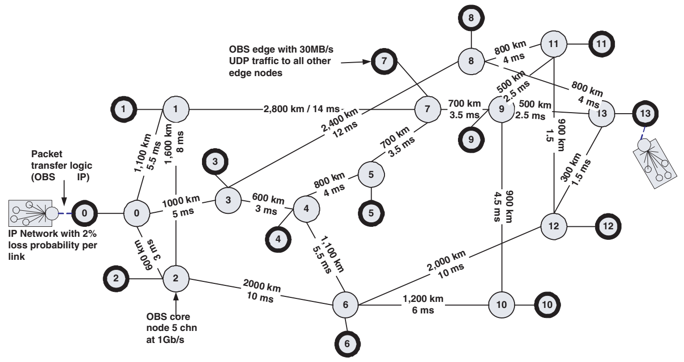


Fig. 3. Split-TCP flows over 14-node NSF network.

We have compared TCP performance for all the proposed approaches using the following metrics:

- **Average TCP Throughput (B/s):** average number of bytes received by the TCP receiver over a time period.
- **Average TCP Congestion Window (# packets):** average send rate of a TCP flow over a time period.
- **Cumulative Number of TCP Fast-Retransmits (FRs):** total number of fast retransmits that have occurred until the current time instant.
- **Cumulative Number of TCP Timeouts (TOs):** total number of time-outs that have occurred until the current time instant.
- **Average TCP Throughput versus different Burst Assembly Timeout (BAT) values:** Average TCP throughput over OBS with different BAT values. BAT values range from 0.1 ms to 10 ms.
- **Flow Completion Time (for 1 GB data transfer):** The total transmission time from source host to destination host.

### A. Average TCP Throughput

Figure 4(a) represents the average throughput for the Baseline-TCP and the *1:1:1* Split-TCP approach and Fig. 4(b) represents the average throughput for the *N:1:N* Split-TCP. We send 1 GB of data over the course of the simulation and we observe that the Baseline-TCP takes $10,000$ seconds to complete the data transfer, while *1:1:1* Split-TCP takes about 400 seconds and *N:1:N* Split-TCP takes just 75 seconds. This is a significant performance improvement over Baseline-TCP.

### B. Average TCP Congestion Window (AWND)

Figure 5(a) shows the average congestion window size of Baseline-TCP and the *1:1:1* Split-TCP (IN, OBS, and OUT) flows and Fig. 5(b) represents the average congestion window size for the *N:1:N* Split-TCP. We observe that the congestion window plots are similar to the TCP throughput graphs, wherein the Split-TCP approaches outperform the Baseline-TCP. Also, *N:1:N* Split-TCP approach significantly outperforms *1:1:1* Split-TCP approach.

### C. Cumulative Number of TCP Fast-Retransmits (FRs)

Figure 6(a) shows the cumulative number of TCP fast-retransmits for Baseline-TCP and the *1:1:1* Split-TCP (IN, OBS, and OUT) flows and Fig. 6(b) represents the same for the *N:1:N* Split-TCP. Fig. 6(a) shows very few fast retransmissions for all the *1:1:1* Split-TCP flows (IN, OBS, and OUT) compared to Baseline-TCP. From Fig. 6(b) we observe only 22 fast-retransmits in the case of *N:1:N* Split-TCP, which indicate random burst contention loss of 22 bursts that were immediately recovered before encountering any time-outs.

### D. Cumulative Number of TCP Time-outs (TOs)

Figure 7(a) shows the cumulative number of TCP time-outs for Baseline-TCP and *1:1:1* Split-TCP (IN, OBS, and OUT) flows. The *1:1:1* Split-TCP flows in IN network encounter few timeouts due to the pre-set 2% loss probability in the IP-access network. TCP over OBS network encounters numerous time-outs primarily due to the fact that OBS-core TCP flows are *fast flows* [18], when we implement a *1:1:1* Split-TCP approach. The OUT network encounters few time-outs due to the pre-set 2% loss probability in the IP-access network. In *N:1:N* Split-TCP approach, we do not observe any time-outs, as the majority of the TCP flows behave as *medium flows* [18].

### E. Average TCP throughput versus different BAT values

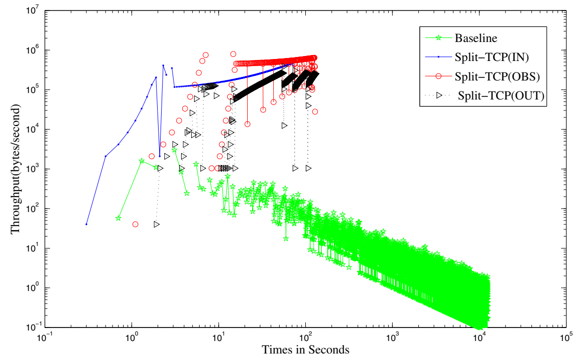
Figure 7(b) shows the throughput between Baseline-TCP and Split-TCP approaches for a hybrid network. From the graph, we observe that the Split-TCP approaches outperform the Baseline-TCP approach for every BAT value. If the BAT value is chosen appropriately we can increase the link utilization efficiently. We also observe that a BAT value of 10 ms (optimal BAT) generates the highest end-to-end TCP throughput.

### F. Flow Completion Time (for 1 GB data transfer)

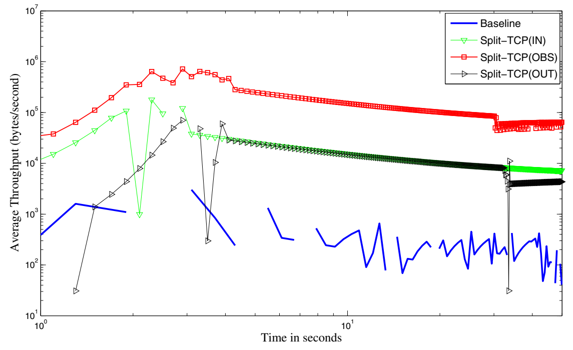
Figure 8 shows the total time to transmit a file from the sender to the destination. The *N:1:N* Split-TCP approach transfers the total file to destination in less time than *1:1:1* Split-TCP approach for different BAT values. When the BAT value is 10ms, the transmission of total packet takes about 77 seconds for *N:1:N* Split-TCP approach compared to the about 667 seconds for *1:1:1* Split-TCP approach. While Baseline-TCP approach took about 10,000 seconds to transfer the same amount of data (refer Fig. 4(a)).

## V. Conclusion: To Split Or Not To Split?

From the above discussions, we can clearly see that the Split-TCP approach significantly improves overall TCP throughput over a hybrid network. By isolating network issues and data contention losses over OBS network from regular

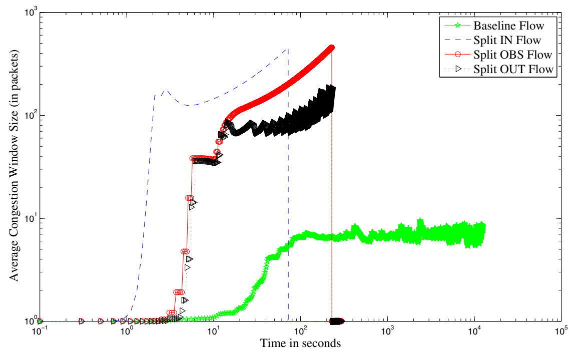


(a)

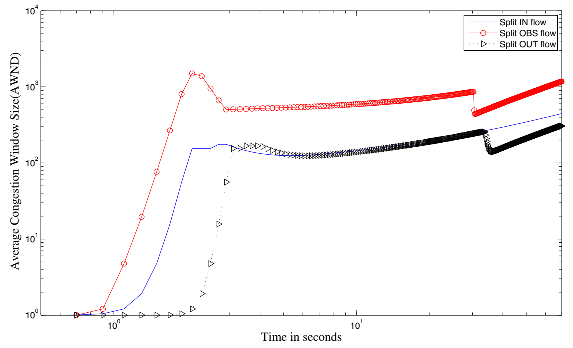


(b)

Fig. 4. Average TCP throughput for (a) 10:10:10 Split-TCP (IN, OUT, OBS) and Baseline-TCP and (b) 10:1:10 Split-TCP (IN, OUT, OBS).

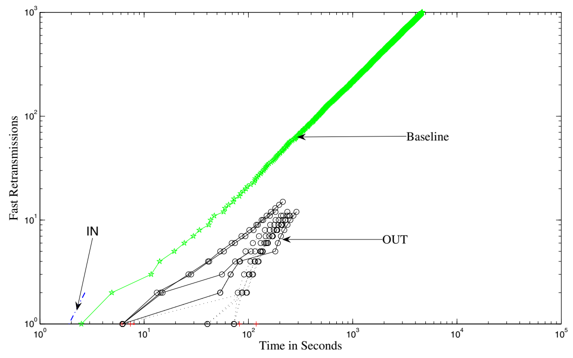


(a)

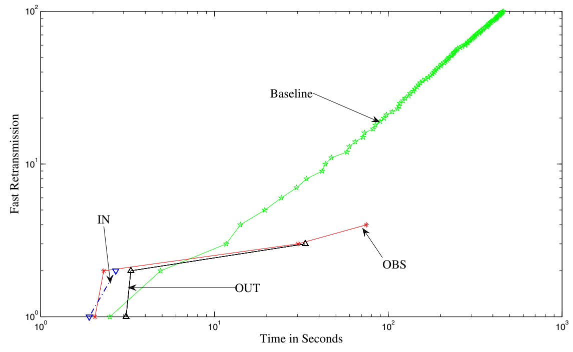


(b)

Fig. 5. Average congestion window size for (a) 10:10:10 Split-TCP (IN, OUT, OBS) and Baseline-TCP and (b) 10:1:10 Split-TCP (IN, OUT, OBS).

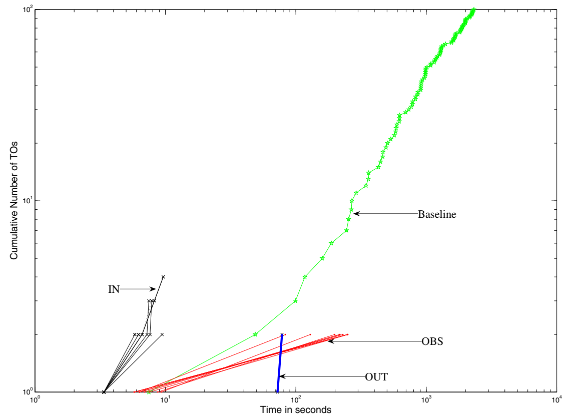


(a)

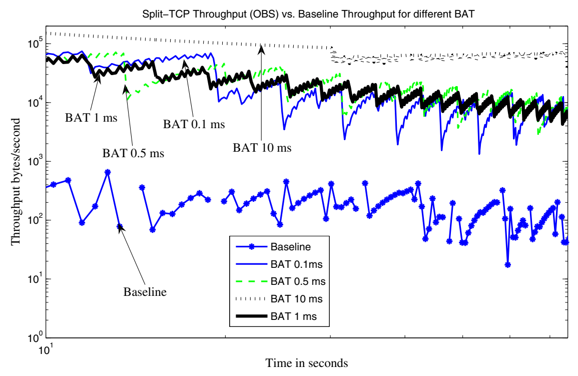


(b)

Fig. 6. Cumulative number of fast-retransmits for (a) 10:10:10 Split-TCP (IN, OUT, OBS) and Baseline-TCP and (b) 10:1:10 Split-TCP (IN, OUT, OBS).



(a)



(b)

Fig. 7. (a) Cumulative number of TCP time-outs for 10:10:10 Split-TCP. (b) Average TCP throughput of Baseline-TCP and *10:1:10* Split-TCP over OBS network with different BAT values.

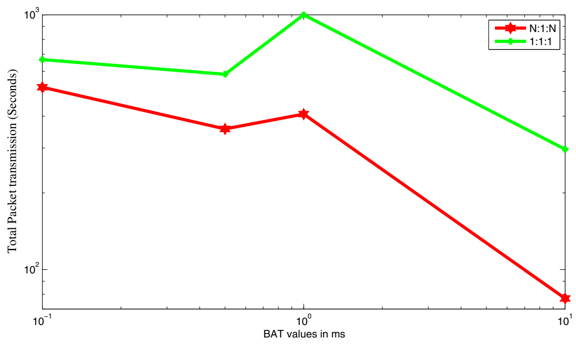


Fig. 8. Flow Completion Time (for 1 GB data transfer) using *1:1:1* Split-TCP and *N:1:N* Split-TCP.

IP network, the Split-TCP approach achieves improved TCP performance. Different BAT values play a major role in the overall throughput of the system. The *N:1:N* Split-TCP approach significantly improves TCP throughput as compared to the *1:1:1* Split-TCP approach and conventional TCP approach.

Comparing the total delay required to transfer 1 GB data, we observe that *N:1:N* Split-TCP is **129** times faster than Baseline-TCP where as *1:1:1* Split-TCP is **14** times faster than Baseline-TCP. Also, if we compare the average TCP throughput, *1:1:1* Split-TCP is up to **18** times better than Baseline-TCP and *N:1:N* Split-TCP is up to **156** times faster than Baseline-TCP. Therefore, we can conclude that adopting Split-TCP approach is significantly beneficial compared to implementing conventional end-to-end TCP connections over a hybrid IP-OBS network.

The two fundamental issues with the Split-TCP approach are the violation of the end-to-end semantics and the overhead incurred in implementing the packet transfer logic at the each split-agent. In order to overcome violation of end-to-end TCP semantics, the split-agent can be enhanced to implement a Snoop-like agent discussed in [21]. The Semi Split-TCP approach can also preserves the end-to-end semantics of TCP, which is discussed in [31]. In the Split-TCP approach, OBS edge nodes have to store large traces of TCP flows. There are several interesting areas of future work, such as cache management, queue management, queue scheduling, fairness, and scalability. Another area of future work is to further improve Split-TCP throughput by implementing loss minimization and loss recovery mechanism at the OBS layer.

## 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] 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, Jun. 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, Sep. 1998.

[5] S. Yao, B. Mukherjee, S. J. B. Yoo, and S. Dixit, "A unified study of contention-resolution schemes in optical packet-switched networks," in *IEEE/OSA Journal of Lightwave Technology*, Mar. 2003.

[6] 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, Nov.-Dec. 2003.

[7] X. Huang, V.M. Vokkarane, and J.P. Jue, "Burst cloning: A proactive scheme to reduce data loss in optical burst-switched networks," in *Proceedings, IEEE International Conference on Communications (ICC)*, May 2005.

[8] Q. Zhang, V. M. Vokkarane, Y. Wang, and J. P. Jue, "Analysis of TCP over optical burst-switched networks with burst retransmission," *IEEE Globecom 2005, Photonic Technologies for Communications Symposium*, Nov. 2005.

[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, pp. 200–222, 2004.

[10] I. Stoica and et. al., "Chord: A scalable peer-to-peer lookup protocol for Internet applications," in *Proceedings of ACM SIGCOMM*, 2001.

[11] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, pp. 5–21, Jul. 1996.

[12] V. Jacobson, "Congestion avoidance and control," in *Proceedings, ACM SIGCOMM*, 1988.

[13] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *RFC 2001*, 1997.

[14] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," *RFC 2582*, 1999.

[15] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," *RFC 2018*, 1996.

[16] W. Feng and P. Tinnakornsrisuphap, "The failure of TCP in high-performance computational grids," in *Proceedings, Supercomputing Conference*, 2000.

[17] X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false time out detection in OBS networks," in *Proceedings, IEEE INFOCOM*, Mar. 2004.

[18] A. Detti and M. Listanti, "Impact of segments aggregation on TCP Reno flows in optical burst switching networks," in *Proceedings, IEEE INFOCOM*, 2002.

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

[20] S. Gowda, R. Shenai, K. Sivalingam, and H. C. Cankaya, "Performance evaluation of TCP over optical burst-switched (OBS) WDM networks," in *Proceedings, IEEE International Conference on Communications (ICC)*, May 2003, vol. 2, pp. 1433–1437.

[21] H. Balakrishnan and et. al., "Improving TCP/IP performance over wireless networks," in *Proceedings, 1st ACM Conf. on Mobile Computing and Networking*, Nov. 1995.

[22] 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. 423–500, Aug. 2004.

[23] D. Barman, I. Matta, E. Altman, and R.E. Azouzi, "TCP optimization through FEC, ARQ and transmission power tradeoffs," in *2nd Intl. Conf. on Wired/Wireless Internet Communications (WWIC)*, Feb. 2004.

[24] A. V. Bakre and B. R. Badrinath, "Implementation and performance evaluation of indirect TCP," *IEEE Trans. Computers*, vol. 46, Mar. 1997.

[25] R. Yavatkar and N. Bhagwat, "Improving endtoend performance of TCP over mobile internetworks," in *Workshop on Mobile Computing Systems and Applications*, Dec. 1994.

[26] Z. Haas and P. Agrawal, "MobileTCP: An asymmetric transport protocol design for mobile systems," in *Proceedings, IEEE International Conference on Communications (ICC)*, Jun. 1997.

[27] K. Wang and S. K. Tripathi, "Mobileend transport protocol: An alternative to TCP/IP over wireless links," in *Proceedings, IEEE INFOCOM*, Mar. 1998, pp. 1046–1053.

[28] T.J. Hacker, B.D. Athey, and B. Noble, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," in *Proceedings, International Parallel and Distributed Processing Symposium (IPDPS)*, 2002, pp. 434–443.

[29] H. Yun and R. Sivakumar, "pTCP: an end-to-end transport layer protocol for striped connections," in *Proceedings, 10th IEEE International Conference on Network Protocols (ICNP)*, Nov. 2002, pp. 24–33.

[30] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10–24, Oct. 1994.

[31] F. Xie, N. Jiang, Y. Hua Ho, and K. A. Hua, "Semi-split TCP: Maintaining end-to-end semantics for split TCP," in *32nd IEEE Conference on Local Computer Networks*, 2007.

[32] "Network simulator: http://www.isi.edu/nsnam/ns/," .

[33] "OBS-NS simulator: http://wine.icu.ac.kr/õbsns/index.php," .