TCP Over Optical Burst Switching: To Split or Not to Split?

Rajesh R. C. Bikram, *Student Member, IEEE*, and Vinod M. Vokkarane, *Member, IEEE*

***Abstract*—TCP-based applications account for a majority of data traffic in the Internet; thus, understanding and improving the performance of TCP over optical burst switching (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 propose a Split-TCP framework for the 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. We also develop an analytical model for end-to-end Split-TCP throughput and verify it with simulations.**

***Index Terms*—IP, OBS, TCP, WDM.**

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.

Optical burst switching (OBS) is one of the promising data transport paradigms for supporting the next-generation optical Internet. In 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]. In this paper, we adopt JET mechanism.

Manuscript received December 02, 2008; revised May 28, 2009. First published July 07, 2009; current version published October 07, 2009. This work was supported in part by the National Science Foundation (NSF) under grant CNS-0626798. This paper was presented in part in IEEE ICCCN 2008 St. Thomas, U.S. Virgin Islands 2008.

The authors are with the Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA 02747 USA (e-mail: rrc@umassd.edu; vvokkarane@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2009.2027031

The primary issue in the OBS core network is contention resolution, since core nodes do not have electronic 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 fiber delay lines [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]. In this paper, we consider a bufferless OBS data plane.

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), e-mail (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 queuing (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 packet loss as a congestion-control technique to determine the send rate into the network. TCP SACK implements congestion control using *time-out* (TO) and *fast-retransmission* (FR) mechanisms [15]. HS-TCP is a new congestion control algorithm for high bandwidth flows [17]. HS-TCP is more aggressive than traditional TCP when the congestion window size is high. HS-TCP uses a modified TCP response function, but it has the same slow start/TO behavior as standard TCP. We have implemented both the Baseline-TCP and Split-TCP framework using HS-TCP.

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) [18]. When the TCP sender detects this (false) congestion, it will trigger *slow start*, resulting in significantly reduced TCP throughput. Another example is when a random burst loss triggers, TCP FR 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 [19]–[21]. 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 framework was proposed [22] 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) [23], [24]. 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 [24] 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 [25], selective repeat protocol (SRP) [26], mobile-TCP [27], and mobile-end transport protocol [28]. The 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 proposed to improve Wireless-TCP performance, such as using parallel TCP flows [29], [30] and using explicit congestion notification [31]. In Split-TCP framework 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.

The primary contribution of this work is the introduction of Split-TCP over a hybrid IP-OBS networks. We isolate the impact of different physical media on the TCP congestion control algorithms by splitting an end-to-end TCP connection into three shorter TCP connections. The TCP/OBS connection handles FFRs and FTOs locally without affecting the adjacent TCP/IP connection at the sender. We also develop a new analytical model to calculate end-to-end TCP throughput using the proposed Split-TCP framework. We quantify the overall gain in performance due to the Split-TCP framework through extensive simulations and analytical modeling.

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 framework 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 framework 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). In Section IV, we model the average end-to-end throughput of Split-TCP. Section V discusses the simulation results, and Section VI concludes the paper and discusses potential areas of future work.

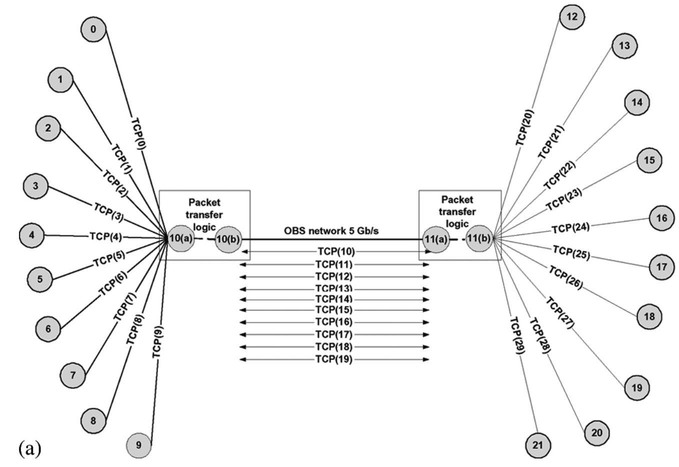
II. Split-TCP Framework

The next-generation optical Internet will be composed 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 (no buffering) 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 propose employing a Split-TCP framework. In the Split-TCP framework (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 to boost end-to-end throughput. In this paper, we propose two approaches for implementing the Split-TCP architecture, 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 an 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 nonpersistent TCP flows and $N:1:N$ Split-TCP approach uses a persistent TCP flow over the OBS-core. 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 flow setup (refer Table I, Columns 3, 4–6). Fig. 1(b) represents the $N:1:N$ Split-TCP flow setup (refer Table I, Columns 3, 5, 6)

In order to implement the Split-TCP framework, the OBS edge nodes should act as an interface between the split connec-



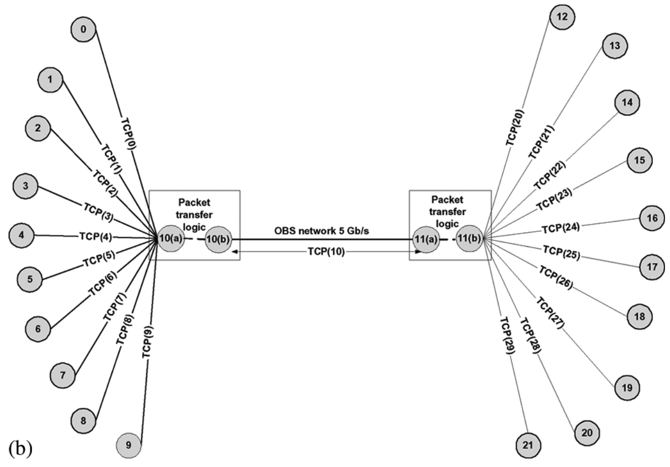


Fig. 1. Split-TCP architecture with TCP flows (a) 1:1:1 and (b) $N : 1 : N$.

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

tions. 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.

An important area of future work is to investigate different cache management policies at the end points of each Split-TCP connection and also investigate QoS-based packet scheduling algorithms to ensure low queueing delay at each packet cache.

In the proposed Split-TCP framework, we split a single long end-to-end TCP connection into multiple TCP connections with shorter round-trip times (RTTs). Due to splitting, each short connection becomes more responsive to both positive (ACKs) and negative (loss) events. In the case of a successful ACK being received by the sender of a Split-TCP connection, the sender can quickly slide its send window and continue transmission of new TCP segments. Also, the previously transmitted TCP segment can now hop on to the next Split-TCP connection (if any) moving toward the destination host. In the case of loss, the data has to be retransmitted only from the beginning of the current Split-TCP connection and not the original TCP sending host. Also, the Split-TCP architecture helps recover from losses on different transmission media within the network that incurs the loss. For example, if a burst is dropped in the OBS-core, the Split-TCP connection over OBS will be able to recover from that loss. This leads to improved TCP performance due to the isolation the loss behavior of OBS (FTO and FFR) from traditional IP networks. These enhancements result in significant improvement of end-to-end TCP throughput when compared to the Baseline-TCP approach.

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. $N : 1 : N$ Split-TCP approach is expected to outperform all other approaches as it uses a persistent TCP connection over the OBS core. A persistent TCP connection can reduce network congestion by minimizing the number of control packets and reduce the TCP connection setup latency for subsequent requests (three-way handshake). In the situation where in we have several long-lived TCP flows, $N : 1 : N$ should perform similar to 1:1:1 Split-TCP. In a real scenario, for $N : 1 : N$ Split-TCP approach, we can have $N : M : N$ Split-TCP configuration, where $1 \leq M \leq N$. M persistent flows will help to prevent Split-TCP buffer overflows and will pipeline more packets to the network without waiting one persistent connection. Moreover, in 1:1:1 Split-TCP approach there will be multiple TCP flows competing for wavelengths simultaneously. This could cause unnecessary contentions at the OBS-ingress node. By multiplexing several TCP/IP flows on to a single TCP/OBS flow, we can minimize unnecessary burst contentions and streamline data transfer.

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 framework. 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

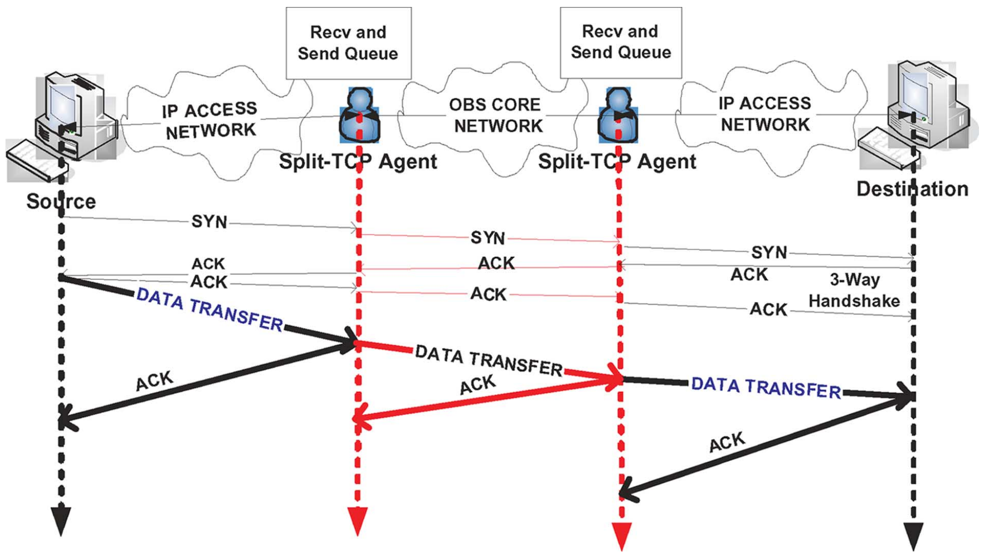


Fig. 2. Signaling framework for 1:1:1 Split-TCP approach.

destination. The proxy (edge nodes) sends the acknowledgment 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 (refer Fig. 2). 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 (ingress) queue and the OBS egress node maintains N sender queues and one receiver (egress) queue. These queues are responsible for isolating the problems in the IP access from the OBS-core network and vice versa.

The Split-TCP framework 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 framework over OBS networks. 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 [22] or Semi Split-TCP [32] 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 RFC 793.

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 following sections, we evaluate the performance of 1:1:1 and $N:1:N$ Split-TCP approaches using analytical modeling and simulations.

IV. Split-TCP: Throughput Modeling

In this section, we modeled the Split-TCP connection using the well-known analytical model of TCP [33] to obtain an approximation for the throughput of three TCP connections in the series as a function of packet loss probability and the average RTT. In this modeling, the receiver's advertised windows size is assumed to be constant. Our model contains the slow-start analysis in addition to the Padhye *et al.* [33] paper. A stochastic model of TCP congestion control is developed in several steps: first when loss is indicated by triple duplicate ACKs, then when loss is indicated by TOs, assuming the receiver's window remains constant. Most of the notation used in this paper is that same as in [33]. We model the end-to-end TCP throughput of the three Split-TCP connections to be the total number of packet received at the third connection.

Let $T_i^{\rm SS}$ denote the duration of the Split-TCP connection in the ith slow-start phase of TCP. Let $T_i^{\rm TD}$ denote the duration of the ith triple duplicate period (TDP; last ends with TO) and $T_i^{\rm TO}$ denotes the duration of the ith TO period. A TDP as defined in [33] is a period between two TD loss indications.

The total time required for a packet to be received at the destination is given by

$$T_i = T_i^{\rm SS} + T_i^{\rm TD} + T_i^{\rm TO}$$

and let M_i to be the total number of packets sent during T_i. If $\{(T_i, M_i)\}_{\rm i}$ is an independent identically distributed sequence of random variables, we have

$$B = \frac{E[M]}{E[T]}$$

where B is long-term steady-state send rate of TCP connection.

Let n_i be the number of rounds in $T_i^{\rm TD}$. Then for the jth round in the TD period of interval $T_i^{\rm TD}$, we define Y_{ij} to be the number of packets sent in the jth round of the ith period, X_{ij} to be the number of rounds in the ith period, and W_{ij} to be the window size at the end of the ith period. Also, R_i denotes the number of packets sent during the TO sequence T_i^{TO} and H_i denotes the number of packets sent during slow start T_i^{SS}

$$M_i = \sum_{j=1}^{n_i} Y_{ij} + R_i + H_i$$

and $\{S_i\}$ is defined as

$$S_i = \sum_{j=1}^{n_i} A_{ij} + T_i^{\mathrm{TO}} + T_i^{\mathrm{SS}}.$$

$\{M_i\}$ is the number of packet sent during $\{S_i\}$. Now, if we assume $\{n_i\}_i$ is an independent identically distributed sequence of random variables, independent of $\{Y_{ij}\}$ and $\{A_{ij}\}$, where $\{Y_{ij}\}$ to be the number of packet sent in the period and $\{A_{ij}\}$ to be the duration of the period, then we have

$$E[M] = E[n]E[Y] + E[R] + E[H]$$

and

$$E[T] = E[n]E[A] + E[T^{\mathrm{TO}}] + E[T^{\mathrm{SS}}].$$

To derive $E[n]$, there are n TDPs during T_i^{TD} and first $(n_i - 1)$ TDPs end in a TD while last TDP ends in TO [33]. Then, we can say that in T_i^{TD}, there is one TO loss indication out of n_i loss indications. Let Q represent the probability that the loss indication ending a TDP is a TO loss indication, so $Q = (1/E[n])$. Consequently

$$B = \frac{E[Y] + Q * (E[R] + E[H])}{E[A] + Q * (E[T^{\mathrm{TO}}] + E[T^{\mathrm{SS}}])}. \tag{1}$$

Loss Indication Due to Congestion Avoidance and FR: Congestion avoidance behavior is modeled in terms of rounds. A round starts with the back-to-back transmission of W packets, where W is the current size of TCP congestion window. Once all the packets in the congestion control window have been sent, no other packet is sent until the first ACK is received for one of these W packets. The receipt of the first ACK marks the end of the current round and the starting of next round. The duration of round is equal to the RTT and is assumed to be independent of the window size. It is also assumed that the time needed to send all the packets in a window is smaller than the RTT. The assumption is also made that a packet is lost in a round independently of any other packets lost in other rounds. Lets define p as probability that a packet is lost. To derive $E[Y]$ for (1), we must first discuss the TDP. Let α_i be the number of packets that have been transmitted in the TDP_i period and X_i denotes the round where this loss occurs. After packet α_i, $W_i - 1$ more packets are sent in an additional round before the TD period ends. So, the maximum number of packets sent during the TDP_i is $Y_i = \alpha_i + W_i - 1$, which are sent in the $X_i + 1$ round. It follows that:

$$E[Y] = E[\alpha] + E[W] - 1. \tag{2}$$

Let $\{\alpha_{ij}\}_i$ be a sequence of i.i.d random variables. We have three Split-TCP connections in series; so the probability that the packet is dropped in the first Split-TCP connection is p. The probability that a packet does not get dropped in the first connection is $(1-p)$. The probability that the packet is successful in the third Split-TCP connection between egress node and the electronic destination host is $(1-p)^3$, and the probability that the packet is dropped in the third TCP connection is $1-(1-p)^3$. The first Split-TCP connection is between the electronic source node A and the ingress node E_{ingress} of OBS network. The second Split-TCP connection is between the egress node E_{egress} of OBS and the electronic destination host *B*. The traditional single TCP connection between A and B is split between A to E_{ingress}, E_{ingress} to E_{egress}, and E_{egress} to B.

$$A ------- E_{\mathrm{ingress}} ---- E_{\mathrm{egress}} -------- B$$

The probability that $\alpha_i = k$ is equal to the probability that exactly $k-1$ packets are successfully acknowledged before a loss occurs, hence

$$P[\alpha = k] = (1-p)^{3(k-1)} * \left(1-(1-p)^3\right), \quad k = 1,2,3,\ldots. \tag{3}$$

The mean value of α is thus

$$\begin{aligned} E[\alpha] &= \sum_{k=1}^{\infty} (1-p)^{3(k-1)} \left(1-(1-p)^3\right) * k \\ &= \left(1-(1-p)^3\right) \sum_{k=1}^{\infty} (1-p)^{3(k-1)} * k \\ &= \frac{1}{(1-(1-p)^3)}. \end{aligned} \tag{4}$$

From (2) and (4), it follows that

$$\begin{aligned} E[Y] &= \frac{1}{(1-(1-p)^3)} - 1 + E[W] \\ &= \frac{(1-p)^3}{(1-(1-p)^3)} + E[W]. \end{aligned} \tag{5}$$

To derive the equation for $E[X]$, we consider the evolution of W_i as function of rounds. At the beginning of last round, the window size would be

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b}, \quad i = 1,2,3\ldots$$

where $1/b$ is slope for each and every new ACKs for the window size. As we know, Y_i is total number of packet transmitted during TDP_i. We can express Y_i by

$$Y_i = \frac{X_i}{2} * \left[W_i + \frac{W_{i-1}}{2} - 1\right] + \beta_i \tag{6}$$

where β_i is the number of packet sent in the last round. As $\{X_i\}$ and $\{W_i\}$ are mutually independent sequence of i.i.d random variables, we have

$$E[W] = \frac{2}{b} * E[X] \tag{7}$$

and

$$E[Y] = \left(\frac{E[X]}{2} * \frac{E[W]}{2} + E[W] - 1\right) + E[\beta] \tag{8}$$

as $\beta = E[W]/2$. Substituting it in the previous equation gives

$$E[Y] = \frac{b}{4}E[W]\left(\frac{E[W]}{2} + E[W] - 1\right) + \frac{E[W]}{2}. \quad (9)$$

Substituting (9) in (5) gives

$$E[W] = \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^3}{1-(1-p)^3}\right)}. \quad (10)$$

Substituting (10) in (5), we have

$$E[Y] = \frac{(1-p)^3}{1-(1-p)^3} + \frac{(2+b)}{3b} + \sqrt{\left(\frac{b+2}{3b}\right)^2 + \frac{8}{3b}\left(\frac{(1-p)^3}{1-(1-p)^3}\right)}. \quad (11)$$

To derive $E[A]$ for TDP_i, let a_{ij} be the duration of the jth round of TDP_i. Hence, the duration of TDP_i is given by

$$A_i = \sum_{j=1}^{X_{i+1}} a_{ij}. \quad (12)$$

Therefore

$$E[A] = (E[X]+1)\,E[a]. \quad (13)$$

$E[a]$ is denoted by RTT, which is the average RTT of three Split-TCP individual RTT. RTT is the total RTT value of third TCP connection in the series.

$$E[A]=\mathrm{RTT}\left(\frac{(2+b)}{6}+\sqrt{\left(\frac{b+2}{6}\right)^3+\frac{2b}{3}\frac{(1-p)^3}{(1-(1-p)^3)}}+\frac{2b}{3}\right). \quad (14)$$

TO and Retransmission: We now obtain an expression for Q, which is the probability that the packet is dropped and eventually times out in the penultimate round, $E[R]$, and $E[T^{\mathrm{TO}}]$. $A(w,k)$ denotes probability that the first k packets are ACKed in a round of w packets, given there is a sequence of one or more losses in the round

$$A(w,k) = \frac{(1-p)^{3k}(1-(1-p)^{3)}}{1-(1-p)^{3w}}.$$

Also, we have to find the $C(n,m)$ to be the probability that m packets are ACKed in sequence in the last round (when n packets were sent).

$$C(n,m) = \begin{cases} (1-p)^{3m}(1-(1-p)^{3)} & m \le n-1 \\ (1-p)^{3n} & m = n. \end{cases}$$

TO occurs in the penultimate round if the number of packets in the window, k, is less than or equal to 3. So, $Q^{\sim}$ is the probability that a loss in a window of size w is a TO given by

$$Q^{\sim}(w) = \begin{cases} 1 & w \le 3 \\ \sum_{k=0}^{2} A(w,k) & \\ \quad + \sum_{k=3}^{w} A(w,k)\sum_{m=0}^{2} C(k,m). & \end{cases} \quad (15)$$

After binomial expansion, we get (16), shown in the equation at the bottom of the page.

Q, the probability that a loss indication is a TO, is given by

$$Q = \sum_{w=1}^{\infty} Q^{\sim}(w)P[W=w] = E[Q^{\sim}].$$

Therefore, we can approximate

$$Q \approx Q^{\sim}(E[W]). \quad (17)$$

To derive $E[R]$, we need to find the probability distribution of the number of TOs in a TO sequence, given that there is a TO. A sequence of k TOs occurs when there are $k-1$ consecutive losses (first loss is given) followed by a successfully transmitted packet. Consequently, the number of TOs in a TO sequence has a geometric distribution, and thus

$$\begin{aligned} P[R=k] &= (1-p)^3\left(1-(1-p)^3\right)^{(k-1)} \\ E[R] &= \sum_{k=1}^{\infty} kP[R=k] \\ &= (1-p)^3\sum_{k=1}^{\infty} k*(1-(1-p^3)^{k-1} \\ &= \frac{(1-p)^3}{(1-(1-(1-p)^3)^2}. \end{aligned}$$

After reducing the aforementioned equation, we can get

$$E[R] = \frac{1}{(1-p)^3}. \quad (18)$$

Next, we derive $E[T^{\mathrm{TO}}]$, which is the average duration of a TO sequence excluding retransmission. We know that the first six TOs in one sequence have length $2^{i-1}\mathrm{TO}$, $i = 1\ldots 6$ with all immediately following TOs having length $64 * \mathrm{TO}$. Then, the duration of sequence with k TOs is given by

$$L_{k=}\begin{cases} (2^k-1)\mathrm{TO} & k \le 6 \\ (63+64(k-6)\mathrm{TO} & k \ge 7 \end{cases}$$

$$Q^{\sim}(w) = \frac{\min\left(1, \left(1-(1-p)^9\right) * \left[\left(1+(1-p)^9\right)\left(1-(1-p)^{3w-9}\right)\right]\right)}{(1-(1-p)^{3w})}. \quad (16)$$

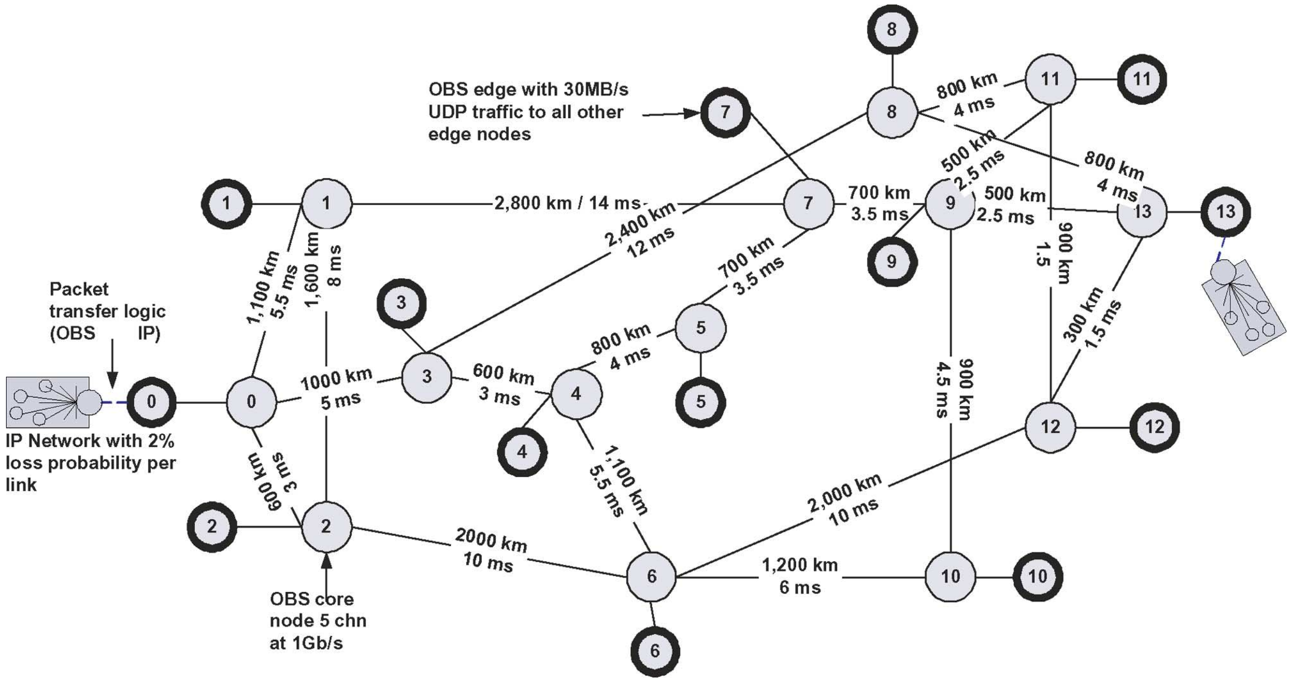


Fig. 3. Split-TCP flows over 14-node NSF network.

and the mean of T^{TO} is

$$E[T^{\mathrm{TO}}] = \sum_{k=1}^{\infty} L_k P[R = k] \tag{19}$$

where

$$\sum_{k=1}^{6} L_k P[R{=}K]{=}(1{-}p)^3\mathrm{TO}\sum_{k=1}^{6}(2^k{-}1){*}\left(1{-}(1-p)^3\right)^{k-1}$$

and

$$\sum_{k=7}^{\infty} L_k P[R = K] = (1-p)^3$$
$$\times \sum_{k=7}^{\infty}\left[63 + 64(k-6) * \left(1-(1-p)^3\right)^{k-1}\right]. \tag{20}$$

Slow-Start Behavior of TCP: The derivation of slow-start TCP is similar to the derivation of congestion avoidance derivation $E[Y]$ as in the Section I.

$E[H]$ is the total number of packet sent during the slow start, so $E[H]$ is given by

$$E[H] = \frac{1}{(1-(1-p)^3)} + \frac{1}{2b}\left[\frac{(1-p)^3}{(1-(1-p)^3)}\right] - \frac{3}{2}. \tag{21}$$

Derivation of $E[T^{\mathrm{SS}}]$ is the same as the derivation of $E[T^{\mathrm{TD}}]$ and given by

$$E[T^{\mathrm{SS}}] = (E[X]+1)\,E[r] \tag{22}$$

where

$$E[X] = \sum_{k=1}^{\infty} k * (1-p)^3)^{2^{k-1}-1}$$

and $E[r] = RTT$, which is the average RTT of the three split connections. In order for the $X_i\mathrm{th}$ round to start, all the packets in the $X_i - 1$ round should be transmitted successfully, so in other words $2^{k-1} - 1$ must be transmitted successfully.

For our model, we are considering limitation of the receiver having a constant receive window size. Let $E[u1]$ be the window size of first Split-TCP connection, $E[u2]$ be the window size of second Split-TCP connection, and $E[u3]$ window size of third Split-TCP connection. Let $W_{\max}$ is the maximum window size. Therefore, the average end-to-end throughput for Split-TCP when $E[u1] < W_{\max}$, $E[u2] < W_{\max}$, and $E[u3] < W_{\max}$ is given by (1) by substituting the values from (11), (14), (17), (18), (19), (20), and (21) to it.

V. Numerical Results

In this section, we evaluate the performance of the proposed Split-TCP approaches using both proposed analytical model and extensive simulations results. All the simulations are performed using Network Simulator-2.33 (NS-2.33) with the OBS module (OBS-0.9a) . The simulations were performed over the NSF network consisting of 14 nodes (refer Fig. 3). The small box connected with dotted lines at Node 0 represents the ingress IP-access Split-TCP flows (referred as IN) and at Node 13 represents egress IP-access Split-TCP flows (referred as OUT). The IP-access networks have a delay of 100 ms between each source node and the OBS ingress and a delay of 100 ms between OBS egress and the destination node. We use a Linux (Fedora) machine with 2 quad core 3 GHz CPUs and 8 GB RAM configuration to run NS-2 simulations.

For the $N:1:N$ Split-TCP approach, we have ten Split-TCP flows over the source IP-access network to the OBS-ingress, one persistent Split-TCP flow from OBS-ingress Node 0 to OBS-egress Node 13 and ten Split-TCP flows over the destination IP-access network. For 1:1:1, we have

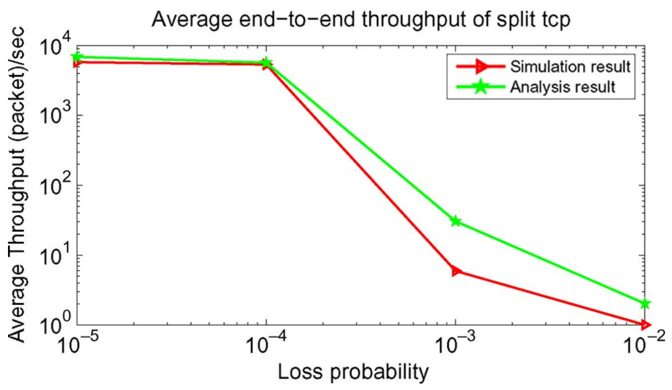


Fig. 4. Average end-to-end Split-TCP throughput using the analytical model.

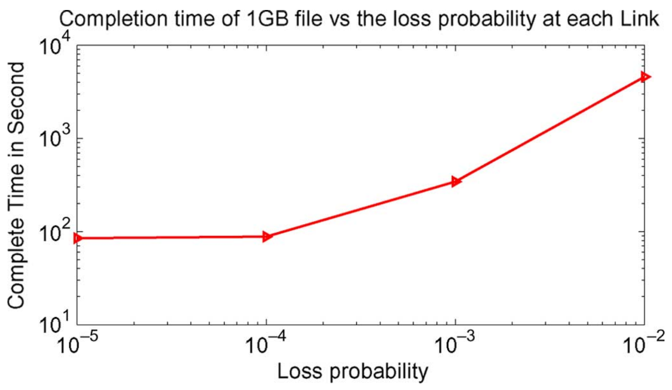


Fig. 5. Completion time of 1 GB file transfer for different contention probabilities using the analytical model.

ten Split-TCP flows from each domain (source IP-access, OBS-core, and destination IP-access), from Node 0 to Node 13. We run a variable bit-rate UDP background traffic at 30 Mb/s between each NSF source–destination pair for all the simulation plots. An FTP traffic generator is used to send a 1 GB file over the different types of TCP connections. We adopt a mixed-timer-and-threshold burst assembly mechanism with a maximum burst size of 1 MB with a timer of 2 ms. The OBS core implements the LAUC-VF scheduling algorithm [34], which selects the latest available unscheduled data channel for each arriving data burst.

We have used the NS-2.33 error model to produce 2% packet loss probability at both the IN and OUT IP-access networks. The loss in the OBS-core network is only due to random data burst contentions. Each core link has 16 data channels at 1 Gb/s. Note that the Split-TCP framework can be applied to any TCP flavor. In this paper, we have used TCP HS-TCP for all simulation-based comparisons. We have compared the TCP performance for all the proposed approaches with different metrics, such as average flow completion time, average congestion window size, average TCP throughput, total number of TOs, and total number of FRs.

A. Analytical Results

In this section, we verify the analytical model developed in Section IV. In Fig. 4, we compare the throughput determined by our expression for a 1:1:1 Split-TCP against the throughput generated by NS-2.33 for a single TCP connection between the same end hosts. To make the analysis simple, we assumed following assumptions:

1) The processing time is negligible and there is no packets loss at the intermediate nodes.
2) The Split-TCP proxy buffers are large enough so as to avoid packet loss due to buffer overflows.
3) The TCP connection is split in such a way that the RTT for each of the three Split-TCP connections is one-third of RTT of longer end-to-end Baseline TCP connection.
4) $\mathrm{RTT} = 100\ \mathrm{ms}$ and $RT0 = 2^{*}\mathrm{RTT}$ for the modeling. RTT refers to the RTT of the third split connection.

We compute the end-to-end throughput using 1:1:1 Split-TCP framework for different loss probabilities. When we say loss probability, we mean the loss probability in each Split-TCP connection. We set the same loss probability value for the IP and OBS networks. Hence, the total end-to-end loss probability is three times the per-split loss probability (as there is loss in each of the three Split-TCP connections). From the Fig. 4, we can observe that the simulation results closely match with the analytical modeling results. Effective TCP throughput decreases as we increase the loss probability in each Split-TCP connection, due to increased TOs.

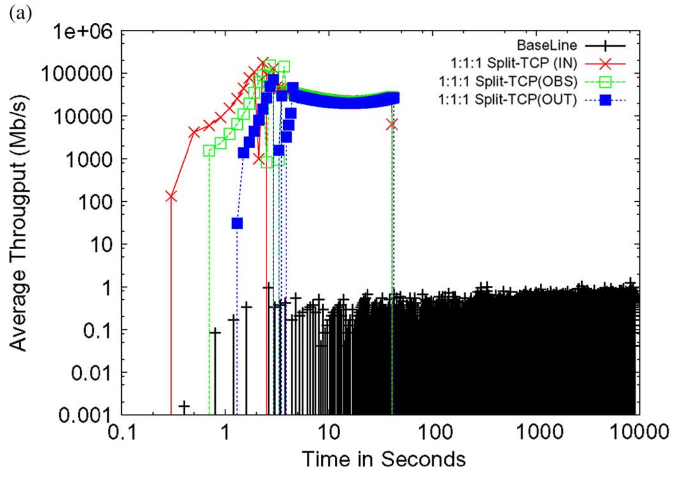
In Fig. 5, we compare the total transmission time required for a 1 GB file transfer with different contention probabilities using the model. As the effective loss probability decreases, the file is able to be transferred in less time. From the figure, we can observe that when the contention probabilities of each Split-TCP connection is set to 10^{-5}, it takes 84 s to transmit the entire file. While at 10^{-2} loss probability, the completion time using Split-TCP is about 5000 s.

B. Average Throughput

In Fig. 6, we compare the average TCP 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. We also observe that for $N : 1 : N$ Split-TCP with burst assembly timer (BAT) values above 2 ms results in high end-to-end TCP throughput as compared to Baseline-TCP [Fig. 10(b)]. The improvement of throughput in each split-connection compared to baseline is due to the fact that each TCP connection handles the congestion problem locally without forwarding to the adjacent IP or OBS network. We observe that the average TCP throughput over OBS with Split-TCP approaches increase significantly because of FTO and FFR isolation from the original sender resulting in the sender injecting more packets in to the network.

C. Average TCP Congestion Window (AWND)

In Fig. 7(a), we compare the average congestion window size for Baseline-TCP and the 1:1:1 Split-TCP (IN, OBS, and OUT) flows, and Fig. 7(b) represents the average congestion window size for the $N : 1 : N$ Split-TCP and Baseline-TCP. We observed that $N : 1 : N$ for OBS split TCP connection uses the congestion window aggressively than 1:1:1 with increasing bat values. It is almost ten times improvement over 1:1:1. The improvement of OBS congestion window is because $N : 1 : N$ uses persistent TCP connection. Both Split-TCP approaches



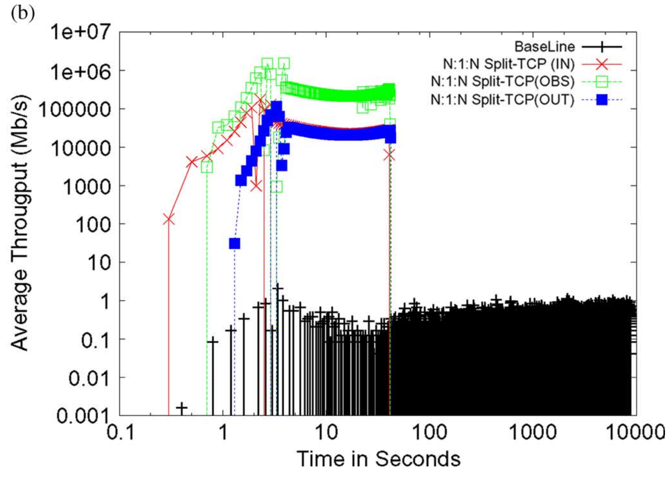
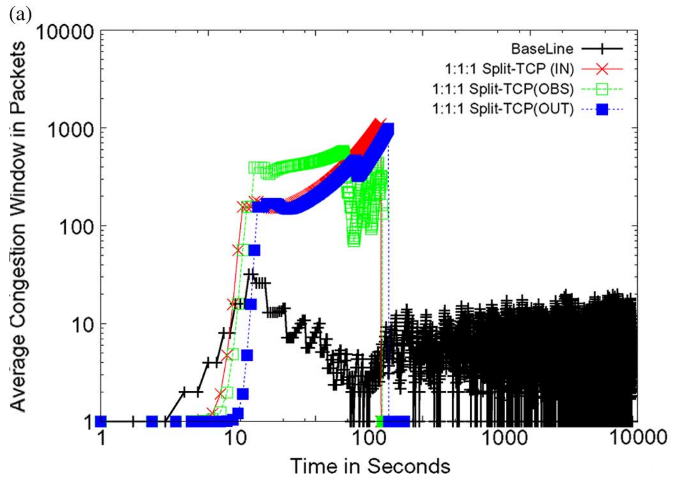


Fig. 6. Average TCP throughput for (a) 10:10:10 Split-TCP (IN, OUT, and OBS) and (b) 10:1:10 Split-TCP (IN, OUT, and OBS).



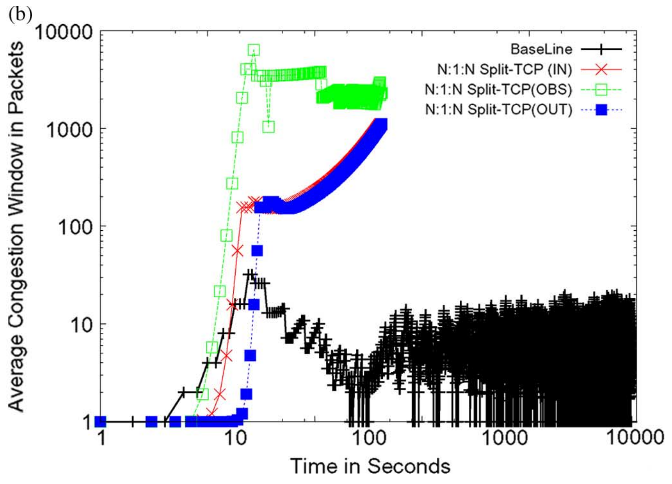


Fig. 7. Average congestion window size for (a) 10:10:10 Split-TCP (IN, OUT, and OBS) and (b) 10:1:10 Split-TCP (IN, OUT, and OBS).

outperforms Baseline-TCP as it handles FFRs and FTOs locally (short Split-TCP connection between OBS edge nodes).

D. Average Number of FRs

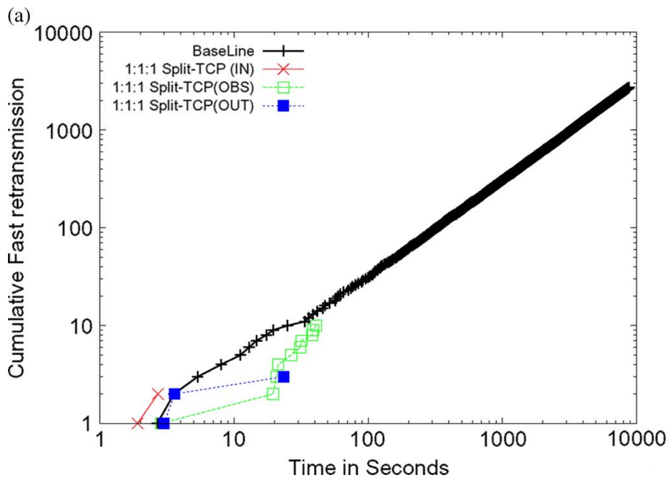
In Fig. 8(a), we compare the average fast retransmits for Baseline-TCP and the 1:1:1 Split-TCP (IN, OBS, and OUT) flows, and Fig. 8(b) represents the same for the $N : 1 : N$ Split-TCP (IN, OBS, and OUT) flows and Baseline-TCP. Fig. 8(a) shows very few FRs for all the 1:1:1 Split-TCP flows (IN, OBS, and OUT) compared to Baseline-TCP. From Fig. 8(b), we observe ten fast retransmits in the case of $N : 1 : N$ Split-TCP for OBS, which indicate random burst contention loss of ten bursts that were immediately recovered locally before encountering any TOs at the original sender. The fewer numbers of FRs for Split-TCP approaches compared to Baseline-TCP is due to the fact that short Split-TCP connection handles congestion problem locally without forwarding it to adjacent split connections.

E. Average Number of TCP TOs

In Fig. 9(a), we compare cumulative number of TCP TOs 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 TOs due to the preset 2% loss probability in the IP-access network. TCP over OBS network encounters numerous TOs primarily due to the fact that OBS-core TCP flows are *fast flows* [19], when we implement a 1:1:1 Split-TCP approach. The OUT network encounters few TOs due to the preset 2% loss probability in the IP-access network. Fig. 9(b) plots the total number of TCP TOs for Baseline-TCP and $N : 1 : N$ Split-TCP approach. We observe few TOs at OBS, as there is only one TCP connection between the OBS edge nodes.

F. Average TCP Throughput Over OBS Versus Different BAT Values

In Fig. 10(b), we compare the average TCP throughput for 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 observe from the average flow completion time versus different BAT values plot for Baseline, $N : 1 : N$, and 1:1:1 approaches that after BAT value of 2 ms, the $N : 1 : N$ and 1:1:1 Split-TCP performance are flat as the BAT increases.



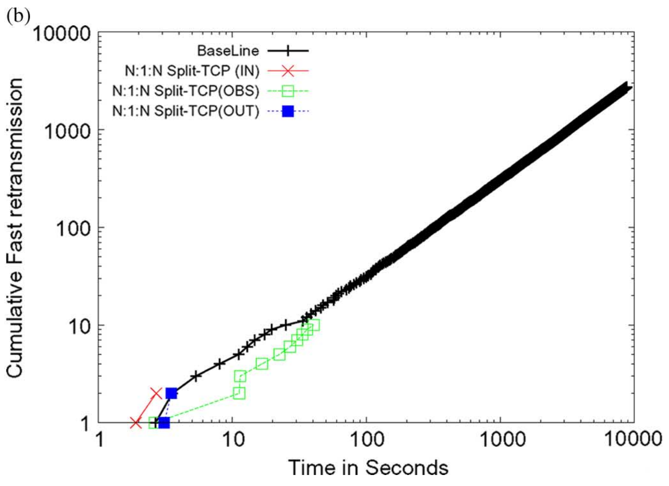
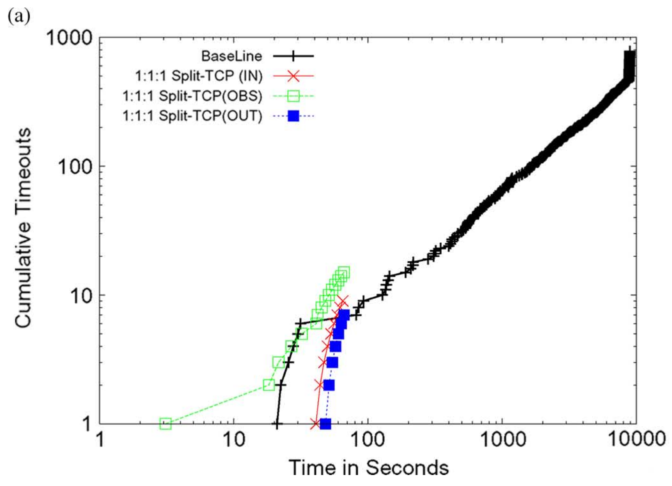


Fig. 8. Cumulative number of fast retransmits for (a) 10:10:10 Split-TCP (IN, OUT, and OBS) and (b) 10:1:10 Split-TCP (IN, OUT, and OBS).



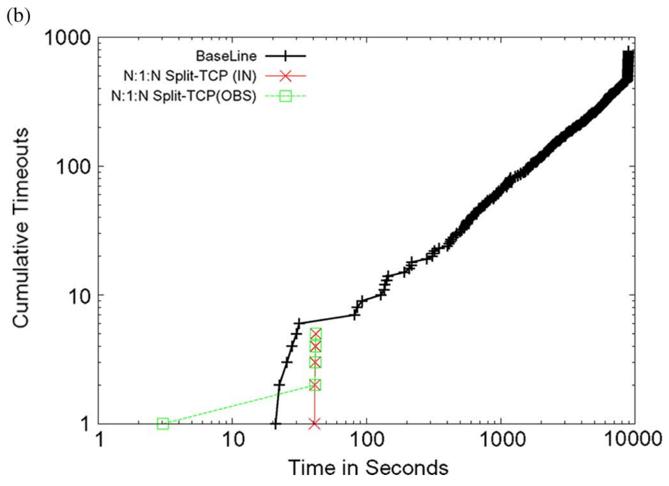


Fig. 9. Cumulative number of TCP TOs for (a) 10:10:10 Split-TCP (IN, OUT, and OBS) and (b) 10:1:10 Split-TCP (IN, OUT, and OBS).

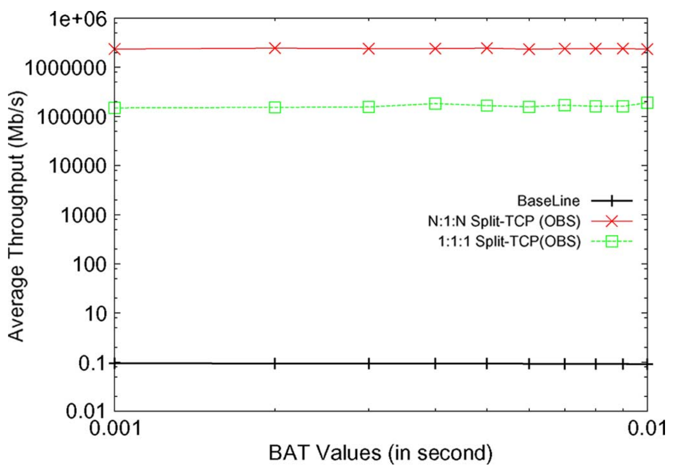


Fig. 10. Average TCP throughput of Baseline-TCP, $N : 1 : N$, and 1:1:1 Split-TCP over OBS network with different BAT values.

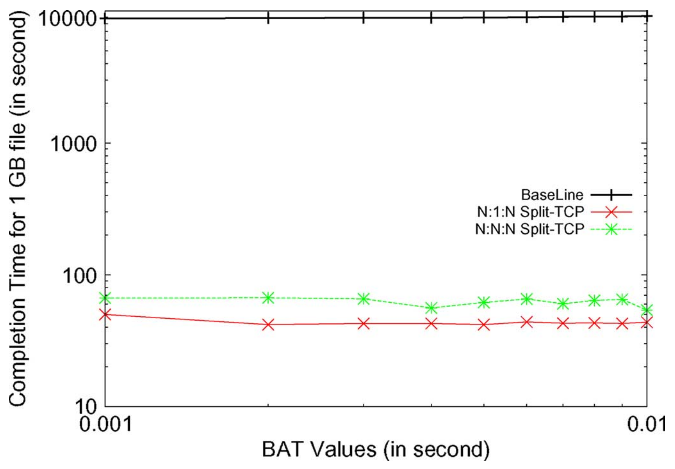


Fig. 11. Flow completion time for 1 GB file transfer using 1:1:1 Split-TCP and $N : 1 : N$ Split-TCP.

G. Flow Completion Time (for 1 GB Data Transfer)

In Fig. 11, we compare the total time to transmit a file from the sender to the destination for the different TCP approaches. 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. With a BAT value of 2 ms, the completion time is about 43 s for $N : 1 : N$ Split-TCP approach compared to that

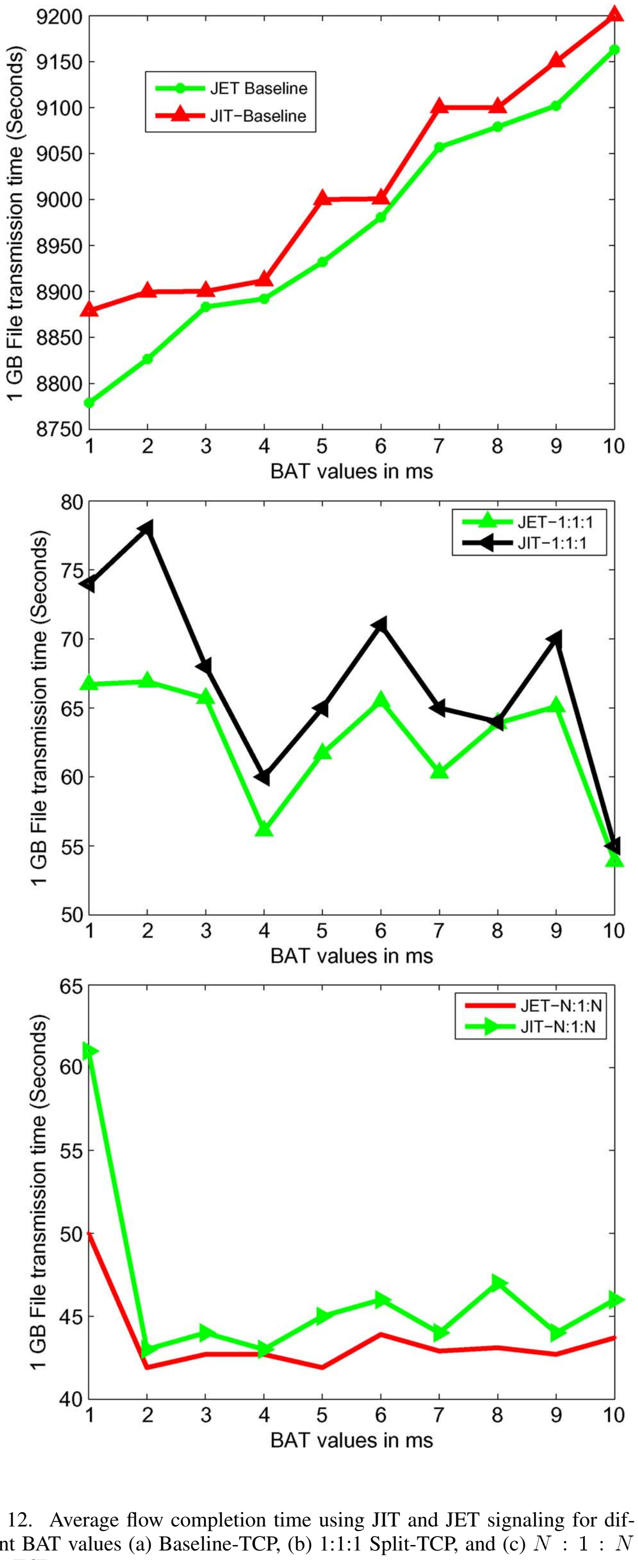


Fig. 12. Average flow completion time using JIT and JET signaling for different BAT values (a) Baseline-TCP, (b) 1:1:1 Split-TCP, and (c) $N:1:N$ Split-TCP.

about 65 s for 1:1:1 Split-TCP approach, while Baseline-TCP approach took about 10 000 s to transfer the same amount of data [refer Fig. 6(a)]. We observe that after a $\mathrm{BAT} = 2$ ms, the $N:1:N$ and 1:1:1 Split-TCP approaches perform flat with increase of BAT.

H. Split-TCP Framework Over Different OBS Signaling Techniques

All the previous simulation plots have been obtained using JET signaling technique. In this section, we evaluate the performance just-in-time (JIT) signaling technique [35]. JIT implements regular reservation and explicit release-based one-way signaling. We implemented JIT in ns 2. JIT favors simplicity over efficiency. From Fig. 12, JIT perform slightly worse than JET for all the TCP versions. We observed that for every BAT value, JET outperforms JIT for all three TCP approaches, Baseline-TCP, 1:1:1 Split-TCP, and $N:1:N$ Split-TCP approaches.

VI. Conclusion

From the aforementioned discussions, we can clearly see that the Split-TCP framework significantly improves overall TCP performance over a hybrid IP-OBS network. By isolating random data contention losses over OBS network from regular IP network, the Split-TCP framework achieves improved TCP performance. The $N:1:N$ Split-TCP approach significantly improves TCP throughput as compared to the 1:1:1 Split-TCP approach and the conventional TCP approach.

Comparing the total delay required to transfer 1 GB data, we observe that $N:1:N$ Split-TCP is 136 times faster than conventional TCP approach, where as 1:1:1 Split-TCP is 14 times faster than conventional TCP approach for almost all BAT values greater than 2 ms. Also, if we compare the average TCP throughput, 1:1:1 Split-TCP is up to ten times better than conventional TCP and $N:1:N$ Split-TCP is up to 190 times faster than conventional TCP. Therefore, we can conclude that adopting the Split-TCP framework is significantly beneficial compared to implementing conventional end-to-end TCP connections over a hybrid IP-OBS network. We show that our simulation results closely match the proposed analytical model results for end-to-end Split-TCP throughput.

The two fundamental issues with the Split-TCP framework 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 [22]. The Semi Split-TCP approach can also preserve the end-to-end semantics of TCP, which is discussed in [32]. In the Split-TCP framework, 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.

In this paper, the proposed Split-TCP framework has been extensively discussed over optical burst-switched networks. The Split-TCP framework will work with minor modifications on other WDM-based optical transport networks, such as optical circuit switching, optical packet switching, and optical flow switching.

Another area of future work is the in-depth research and analysis of Split-TCP architecture over OBS ring metro networks. OBS metro ring networks is gaining increasing interest by operators. In an OBS metro ring with N nodes, each node owns a home wavelength on which it transmits data bursts

[36]. This ring network can also utilize the benefits of the proposed Split-TCP framework in order to obtain TCP performance speed up. Split-TCP proxies can be implemented on the gateway nodes that connect metro ring networks.

ACKNOWLEDGMENT

The authors sincerely thank all the reviewers for their valuable comments.

REFERENCES

[1] J. P. Jue and V. M. Vokkarane, *Optical Burst Switched Networks*. New York: Springer-Verlag, 2005.

[2] C. Qiao and M. Yoo, "Optical burst switching (OBS)—A new paradigm for an optical Internet," *J. High Speed Netw.*, vol. 8, no. 1, pp. 69–84, 1999.

[3] I. Chlamtac, A. Fumagalli, L. G. Kazovsky, P. Melman, W. H. Nelson, P. Poggiolini, M. Cerisola, A. N. M. M. Choudhury, T. K. Fong, R. T. Hofmeister, C.-L. Lu, A. Mekkittikul, D. J. M. Sabido, C.-J. Suh, and E. W. M. Wong, "CORD: Contention resolution by delay lines," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 1014–1029, Jun. 1996.

[4] B. Ramamurthy and B. Mukherjee, "Wavelength conversion in WDM networking," *IEEE J. Sel. Areas Commun.*, 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," *J. Lightw. Technol.*, vol. 21, no. 9, pp. 672–683, Mar. 2003.

[6] V. M. Vokkarane and J. P. Jue, "Burst segmentation: An approach for reducing packet loss in optical burst switched networks," *SPIE Opt. Netw. Mag.*, vol. 4, no. 6, pp. 81–89, 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 *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2005, pp. 1673–1677.

[8] Q. Zhang, V. M. Vokkarane, Y. Wang, and J. P. Jue, "Analysis of TCP over optical burst-switched networks with burst retransmission," in *Proc. IEEE Globecom 2005 Photon. Technol. Commun. Symp.*, Nov. , p. 6.

[9] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, pp. 200–222, 2004.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications," in *Proc. ACM SIGCOMM*, 2001, pp. 149–160.

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

[12] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, 1988, pp. 314–329.

[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 *Proc. Supercomput. Conf.*, 2000, p. 37.

[17] S. Floyd, Highspeed TCP for large congestion windows *RFC 3649*, Dec. 2003.

[18] X. Yu, C. Qiao, and Y. Liu, "TCP implementations and false time out detection in OBS networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 774–784.

[19] A. Detti and M. Listanti, "Impact of segments aggregation on TCP Reno flows in optical burst switching networks," in *Proc. IEEE INFOCOM*, 2002, pp. 1803–1812.

[20] X. Yu, C. Qiao, Y. Liu, and D. Towsley, Performance evaluation of TCP implementations in OBS networks The State Univ. New York at Buffalo, Tech. Rep. 2003-13, 2003.

[21] S. Gowda, R. Shenai, K. Sivalingam, and H. C. Cankaya, "Performance evaluation of TCP over optical burst-switched (OBS) WDM networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2003, vol. 2, pp. 1433–1437.

[22] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz , "Improving TCP/IP performance over wireless networks," in *Proc. 1st ACM Conf. Mobile Comput. Netw.*, Nov. 1995, pp. 2–11.

[23] C. Barakat and A. A. Fawal, "Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic," *Perform. Eval. J.*, vol. 57, pp. 423–500, 2004.

[24] D. Barman, I. Matta, E. Altman, and R. E. Azouzi, "TCP optimization through FEC, ARQ and transmission power tradeoffs," in *Proc. 2nd Int. Conf. Wired/Wireless Internet Commun. (WWIC)*, Feb. 2004, pp. 87–98.

[25] A. V. Bakre and B. R. Badrinath, "Implementation and performance evaluation of indirect TCP," *IEEE Trans. Comput.*, vol. 46, no. 3, pp. 260–278, Mar. 1997.

[26] R. Yavatkar and N. Bhagwat, "Improving endtoend performance of TCP over mobile internetworks," in *Proc. Workshop Mobile Comput. Syst. Appl.*, Dec. 1994, pp. 146–152.

[27] Z. Haas and P. Agrawal, "MobileTCP: An asymmetric transport protocol design for mobile systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 1997, pp. 1054–1058.

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

[29] 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 *Proc. Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2002, pp. 434–443.

[30] H. Yun and R. Sivakumar, "pTCP: An end-to-end transport layer protocol for striped connections," in *Proc. 10th IEEE Int. Conf. Netw. Protocols (ICNP)*, Nov. 2002, pp. 24–33.

[31] S. Floyd, "TCP and explicit congestion notification," *ACM Comput. Commun. Rev.*, vol. 24, pp. 10–24, 1994.

[32] F. Xie, N. Jiang, Y. H. Ho, and K. A. Hua, "Semi-split TCP: Maintaining end-to-end semantics for split TCP," in *Proc. 32nd IEEE Conf. Local Comput. Netw.*, 2007, pp. 303–314.

[33] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, Apr. 2000.

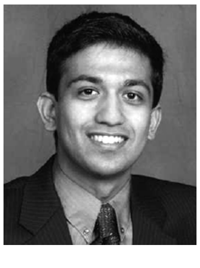
[34] Y. Xiong, M. Vanderhoute, and H. C. Cankaya, "Control architecture in optical burst-switched WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1838–1854, Oct. 2000.

[35] J. Y. Wei and R. I. McFarland Jr., "Just-in-time signaling for WDM optical burst switching networks," *J. Lightw. Technol.*, vol. 18, no. 12, pp. 2019–2037, Dec. 2000.

[36] [Online]. Available: Available: http://www.matissenetworks.com/ 2008

Rajesh R. C. Bikram (S'07) received the B.E. degree in information technology from Pokhara University, Pokhara, Nepal, in 2005. He is currently pursuing the M.S. degree in computer science at the University of Massachusetts, Dartmouth.

His recent research interests include computer networks and operation system design.



Vinod M. Vokkarane (S'02–M'04) received the B.E. degree (with honors) in computer science and engineering from the University of Mysore, Mysore, India, in 1999, the M.S. and Ph.D. degrees in computer science from the University of Texas at Dallas, Richardson, in 2001 and 2004, respectively.

Currently, he is an Assistant Professor of computer and information science at the University of Massachusetts, Dartmouth. He also serves as an Associate Editor for IEEE COMMUNICATION LETTERS. His recent research interests include design and analysis of architectures and protocols for optical and wireless networks. He is the coauthor of the book *Optical Burst Switched Networks*, Springer, 2005.

Dr. Vokkarane is a recipient of the Texas Telecommunication Engineering Consortium Fellowship 2002–2003 and the University of Texas at Dallas Computer Science Dissertation of the Year Award 2003–2004.