Wireless Personal Communications (2020) 111:1505–1524
https://doi.org/10.1007/s11277-019-06934-6

Optimal Load Balancing Linked Increased Algorithm for Multipath TCP

Rajnish Kumar Chaturvedi[1] · Satish Chand[1]

Published online: 16 November 2019

Abstract
The most commonly used mechanism for establishing a connection between two communicating entities is transmission control protocol (TCP) that provides a reliable connection. Due to the latest developments in communication technologies, one can use multiple connections in a network using Wi-Fi, LTE/HSPA, etc. Since the single path TCP (SPTCP) provides a single connection at a time, it cannot take the benefit of multiple connections. The multipath TCP (MPTCP) can this problem by dividing a flow into multiple subflows; each may use a different network connection. Thus, it has a great capability to achieve high throughput in comparison to the SPTCP. The existing MPTCP congestion control algorithms have different issues like TCP-friendliness, responsiveness, and load balancing. To address these issues, this paper proposes a new MPTCP congestion control algorithm, named as optimal load balancing linked increased algorithm (OLBLIA), that considers the congestion window corresponding to a path that is least congested. The window corresponding to the least congested path is increased with maximum size (i.e., nearly equal to the congestion window of SPTCP in the best path) that makes it more responsive and friendlier towards the SPTCP. Experimentally and analytically it is shown that the proposed OLBLIA resolves the above mentioned issues and outperforms the existing MPTCP's congestion control algorithms.

Keywords Congestion pricing · Congestion control · MPTCP · Throughput · Load balancing

1 Introduction

The end-user devices such as tablets, mobile phones, IoT (Internet-of-Things) devices, etc. have multiple interfaces (e.g. Wi-Fi, 4G/5G) that are supposed to provide various real-time next-generation services, like live data streaming, autonomous driving, etc. [1–3]. In order

✉ Rajnish Kumar Chaturvedi
chaturvedi040@gmail.com

Satish Chand
schand86@hotmail.com

[1] School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, Delhi 110067, India

to use real-time facilities, the underlying network should be more robust and have higher bandwidth and lower latency. The multipath TCP (MPTCP) [4] is a better mechanism to support these services by utilizing all available interfaces simultaneously and aggregating their bandwidth in a seamless manner [5–7]. It provides higher throughput and makes the network robust [8, 9]. Other major application of MPTCP can be in datacenter networks that have large number of switches and servers (nodes), which provide multiple paths between the nodes [10, 11]. The MPTCP is the protocol utilizes these paths simultaneously and aggregates their bandwidth in a seamless manner [12].

The multipath TCP may be considered as an extension of the normal TCP (i.e., single-path TCP (SPTCP) like TCP Reno) [13, 14]. The Internet Engineering Task Force (IETF) has an MPTCP working group that has standardized it [15]. The MPTCP breaks a TCP connection into multiple sub-connections according to available networks/interfaces to the user. It considers a sub connection corresponding to each interface as an individual path and each path has its round-trip-time (RTT) and congestion window (CW) [16, 17]. The CW of each path is decided by an MPTCP congestion control algorithm. An MPTCP congestion control algorithm follows the same phases as the SPTCP to update the CW of each path: in the slow-start phase, the CW increases exponentially and in the congestion avoidance phase linearly. According to Raiciu et al. [18], an MPTCP congestion control algorithm should fulfill three basic goals: (1) *Do not harm,* (2) *Improve throughput, and* (3) *Balance congestion*. The combined effect of these goals makes the MPTCP as TCP-friendly, less aggressive towards SPTCP, and more responsible [13, 19].

According to the working mechanisms of MPTCP congestion control algorithms, they may be categorized into three groups: *uncoupled Congestion Control, semicoupled congestion control, and coupled congestion control algorithms.*

- *Uncoupled congestion control (UCC) algorithms* The basic idea in these types of algorithm is that each subflow is treated as individual TCP connection and the CW of each subflow is changed without having any information about other paths. This solution is however not satisfactory because it has various types of problems like aggressiveness, fairness, etc.
- *Semi-coupled congestion control (SCC) algorithms* The basic idea of the semicoupled congestion control algorithms is to increase the CW of each subflow of the same source by a common factor at the same time. However, these types of algorithms have the problem of load balancing [20].
- *Coupled congestion control (CCC) algorithms* The basic idea in these types of algorithms is that each subflow has information about other subflows while deciding its CW that takes care of the congestion in other subflows. By doing this, load balancing is automatically taken care of and the MPTCP is less aggressive towards the normal TCP (SPTCP).

The existing research shows that the coupled congestion control (CCC) algorithms outperform the uncoupled congestion control (UCC) algorithms and semi-coupled congestion control (SCC) algorithms. Thus, the CCC algorithms are best-suited for MPTCP. There have been discussed some algorithms in literature; the important ones include the balanced linked adaptation (BALIA) [21], opportunistic linked increase algorithm (OLIA) [22], and linked increase algorithm (LIA) [18]. The LIA does not provide efficient load balancing [20]. In the congestion avoidance phase, for each ACK of a subflow, the LIA increases its CW by either the maximum of the incremental term of the Kelly and Voice or the inverse of the CW [23]. For packet loss scenario, the CW is halved. The process of increasing the

CW for each ACK in the subflows of MPTCP results in performance degradation of the SPTCP, which shares the corresponding MPTCP subflows. This results in forced trade-off between the load balancing and responsiveness; further making it unfriendly for the SPTCP.

The OLIA [22] addresses the problem of load balancing, but like the LIA, it also suffers from the problem of unresponsiveness [19, 20]. The OLIA can be considered as an alternative to the coupled algorithm discussed in [24] for friendliness. For a subflow the CW in OLIA is increased by considering the incremental term of the Kelly and Voice [23]. The incremental term of the Kelly and Voice forming the responsiveness factor in the OLIA is the reason for unresponsiveness during the network changes. The timely reaction to the changes in network conditions can be characterized by responsiveness for an MPTCP algorithm.

The BALIA, a generalization of the LIA and OLIA algorithms, considers the TCP-friendliness, windows oscillation, and responsiveness. The fluctuations in the window size around the equilibrium point are characterized as the windows oscillation. In BALIA, a responsiveness factor (normalized factor) is multiplied with the incremental term of the Kelly and Voice. This is how the CW of a subflow in BALIA is increased for each ACK. In the packet loss scenario, the CW of a subflow is nearly halved [21]. The responsiveness factor increases the CW considerably for low data rate subflows. The low data rate in a subflow exists due to the bandwidth sharing with other TCP connections. This may result in high packet drop rate and the SPTCP can suffer from the aggressiveness and unfriendliness.

In next section, the proposed work is introduced that addresses the above mentioned problems of aggressiveness, responsiveness, TCP-friendliness, and load balancing.

2 Proposed Work: Optimal Load Balancing Linked Increased Algorithm

The MPTCP may be considered as an extension of the SPTCP that uses multiple networks for data transmission. As discussed in previous section, the existing MPTCP congestion control algorithms have many issues like load balancing, unresponsiveness, and friendliness. In order to mitigate these problems, a new MPTCP congestion control algorithm, named as Optimal Load Balancing Linked Increased Algorithm (OLBLIA), is proposed that addresses three basic goals, i.e., *Balance Congestion, Do not Harm, and Improve throughput*. Further, the OLBLIA provides better responsiveness, less aggressiveness towards SPTCP, high TCP-friendliness, and adequate load balancing. The proposed OLBLIA MPTCP algorithm adjusts its CW of each subflow during the congestion avoidance phase as follows:

- Increase window size $\mathrm{w_r}$ for each ACK of subflow *r* by

$$\mathrm{w_r} = \mathrm{w_r} + \left\{ \frac{\frac{\mathrm{w_r}}{\mathrm{rtt_r^2}}}{\left(\sum_{p \in R} \frac{\mathrm{w_p}}{\mathrm{rtt_p}}\right)^2} \alpha_{\mathrm{r}}^2 \right\} \tag{1}$$

- Decrease window size $\mathrm{w_r}$ for each packet loss of subflow *r* by

$$\mathrm{w_r} = \mathrm{w_r} - \frac{\mathrm{w_r}}{2} \tag{2}$$

where $\alpha_r = \frac{\sum_{p\in R} x_p}{max\{x_p\}}$, and $x_p = \frac{w_p}{rtt_p}$.

The increment term in (1) is the product of two terms. The first term is $\frac{w_r}{rtt_r^2} / \left(\sum_{p\in R} \frac{w_p}{rtt_p}\right)^2$, which is the Kelly and Voice's increment term, that is used to make the algorithm to fulfill "*Do not harm*" condition. The second term is α_r^2 that makes it TCP-friendly and more responsive towards the SPTCP.

The symbols used in this paper are listed in Table 1.

The first term of the increment term in (1) can be written as

$$\begin{aligned} \frac{\frac{w_r}{rtt_r^2}}{\left(\sum_{p\in R} \frac{w_p}{rtt_p}\right)^2} &= \frac{\frac{w_r}{rtt_r^2}}{\left(\frac{w_1}{rtt_1} + \frac{w_2}{rtt_2} + \cdots + \frac{w_r}{rtt_r} + \cdots + \frac{w_n}{rtt_n}\right)^2} \\ &= \frac{\frac{w_r}{rtt_r^2}}{\left(1 + \frac{\frac{w_1}{rtt_1}+\frac{w_2}{rtt_2}+\cdots+\frac{w_n}{rtt_n}}{\frac{w_r}{rtt_r}}\right)^2 * \frac{w_r^2}{rtt_r^2}} \\ &= \frac{1}{\left(1 + \frac{\frac{w_1}{rtt_1}+\frac{w_2}{rtt_2}+\cdots+\frac{w_n}{rtt_n}}{\frac{w_r}{rtt_r}}\right)^2 * w_r} \end{aligned} \tag{3}$$

since $\left(1 + \frac{\frac{w_1}{rtt_1}+\frac{w_2}{rtt_2}+\cdots+\frac{w_n}{rtt_n}}{\frac{w_r}{rtt_r}}\right)^2 \geq 1$, it can be written as follows:

$$\frac{1}{\left(1 + \frac{\frac{w_1}{rtt_1}+\frac{w_2}{rtt_2}+\cdots+\frac{w_n}{rtt_n}}{\frac{w_r}{rtt_r}}\right)^2 * w_r} \leq \frac{1}{w_r} \tag{4}$$

Equation (4) shows that the first term in the increment term in (1) never takes more value than the inverse of CW of that subflow, i.e. $\frac{1}{w_r}$. This $\frac{1}{w_r}$ is the incremental term of the SPTCP (TCP-Reno) for each ACK. This proves that the first term in the increment term in (1) makes the proposed algorithm to less aggressive towards the SPTCP.

The second term in the increment term in (1) is α_r^2 and the value of α is always greater than one. For the best path, increment term in (1) is very close to the SPTCP for each ACK. The increment term in (1) can be written as follows:

Table 1 Symbols used in paper

Symbols	Description
s	Source
R	Set of routes
r	Subflow
rtt_r	Round-trip-time of subflow r
w_r	Congestion window of subflow r
x_r	Data rate of subflow r
α_r	Responsiveness factor

$$\begin{aligned}\frac{\frac{w_r}{rtt_r^2}}{\left(\sum_{p\in R}\frac{w_p}{rtt_p}\right)^2}\alpha_r^2 &= \frac{\frac{w_r}{rtt_r^2}}{\left(\sum_{p\in R}x_p\right)^2}\alpha_r^2 \\ &= \frac{\frac{w_r}{rtt_r^2}}{\left(\sum_{p\in R}x_p\right)^2} * \left(\frac{\sum_{p\in R}x_p}{max\{x_p\}}\right)^2 \\ &= \frac{w_r}{rtt_r^2 * \left(max\{x_p\}\right)^2} \\ &= \frac{w_r}{rtt_r^2 * \left(max\left\{\frac{w_p}{rtt_p}\right\}\right)^2}\end{aligned} \tag{5}$$

Assuming the RTT for each subflow nearly the same, Eq. (5) can be written as follows:

$$\frac{\frac{w_r}{rtt_r^2}}{\left(\sum_{p\in R}\frac{w_p}{rtt_p}\right)^2}\alpha_r^2 = \frac{w_r}{\left(max\{w_p\}\right)^2} \tag{6}$$

We consider two cases to find the upper and lower bounds of CW in the proposed mechanism:

(a) *Case 1* Among all subflows, the CW for rth subflow w_r is maximum. Eq. (6) can be written as follows:

$$\frac{\frac{w_r}{rtt_r^2}}{\left(\sum_{p\in R}\frac{w_p}{rtt_p}\right)^2}\alpha_r^2 = \frac{1}{w_r} \tag{7}$$

As evident from (7), for each ACK, the proposed algorithm ensures that the increment in CW of a subflow is equal to the inverse of the CW of the subflow that has maximum data rate. Thus, it never takes more bandwidth than the SPTCP in case of the best path.

(b) *Case 2* Among all sublows, CW of rth subflow w_r is minimum. Eq. (6) can be written as follows:

$$\frac{\frac{w_r}{rtt_r^2}}{\left(\sum_{p\in R}\frac{w_p}{rtt_p}\right)^2}\alpha_r^2 = \frac{w_r}{\left(max\{w_p\}\right)^2} \le \frac{1}{w_r} \tag{8}$$

From (8), it can be seen that the increment term in (1) is upper-bounded by the inverse of the CW of that subflow. Thus, the proposed algorithm satisfies "Do not harm" condition, as shown in (7) and (8).

In (1), α helps managing the change in the current CW for load balancing during the congestion avoidance phase. It helps in distributing more load via the best path and minimum load via the worst path. So, it tries to provide the data load in each subflow up to its optimum capacity, without affecting the performance of other SPTCP flows shared with MPTCP's subflows.

The fundamental concept of the proposed algorithm is that the CW of each subflow of a multipath TCP is increased according to the available bandwidth of the link shared by that subflow. It increases in the least amount the CW of a subflow corresponding to the lowest data flow rate (most congested path) and increases in the maximum amount the CW corresponding to the subflow that has maximum data flow rate (least congested path). Thus, it reduces the flappiness (*during the availability of multiple good paths, randomly flipping the traffic among the paths is termed as flappiness*). When it detects that the congested path becomes free by getting the more available bandwidth, it increases the data rate by increasing the window size with a higher rate of that flow. Thus, it overcomes the congested flow very quickly and becomes more responsive. In next subsection, a model is discussed to understand the proposed mechanism in better way.

2.1 Proposed Model

Consider a network with a set of links $L = \{l_1, l_2, \dots, l_n\}$ and c_i as the finite capacity of link l_i. The links in L are shared by a set of sources $S = \{s_1, s_2, \dots, s_m\}$. Each source $s \in S$ is a fixed collection of subflows/paths $R_s \subseteq R$, where $R = \{r | r \in R_s, s \in S\}$ denotes the collection of all subflows. Each subflow $r \in R_s$ uses a set of links $L_r \subseteq L$. let A_{lr} be a $N \times R$ routing matrix defined as follows:

$$A_{lr} = \begin{cases} 1, & \textit{if } l \in r \\ 0, & \textit{otherwise} \end{cases}$$

Let source $s \in S$ maintain a congestion window w_r and data transmission rate $x_r = w_r/rtt_r$, for subflow $r \in R_s$ at some particular time, where rtt_r denotes the RTT of subflow r. Let p_r be the congestion price at that time associated with link l. Denote the aggregate price on subflow r as $q_r = \sum_{l \in L} A_{lr} p_l$ and the aggregate traffic rate on link l as $y_r = \sum_{r \in R} A_{lr} x_r$. For subflow $r \in R_S$, x_r, w_r, q_r represent the corresponding state variables and for source $s \in S$, x_s, w_s, q_s represent the corresponding state variables such that $x_s = x_r$, $w_s = w_r$, $q_s = q_r$, for $r \in R_s$, $s \in S$.

In the MPTCP model, let each source transmit the data by different subflows at transmission rate x_r using subflow r. For each successful ACK, the CW is increased by $I_r(w_s)$ and, for each loss, the CW is decreased by $D_r(w_s)$ on subflow r, where w_s represents the vector of window sizes on different subflows of the source s. For each increment/decrement, the maximum loss based the MPTCP algorithm is given by following:

- For each ACK on subflow r, $w_r = w_r + I_r(w_s)$.
- For each packet loss on subflow r, $w_r = w_r + D_r(w_s)$.

Source s transmits x_r packets per unit time on subflow r and receives positive/negative ACK at almost same rate, assuming every packet is acknowledged. The source s receives $x_r(1 - q_r)$ as the number of positive ACKs per unit time on subflow r and each positive ACK increases the CW by $I_r(w_s)$. It receives $x_r q_r$ as the number of negative ACKs per unit time on subflow r and each negative ACK decreases its CW by $D_r(w_s)$. According to *Low* [25], for each RTT, the net change in CW, δw_r, on subflow r is roughly given by

$$\delta w_r = \left(I_r(w_s)(1 - q_r) - D_r(w_s) q_r\right) w_r$$

The above equation estimates the change in CW that helps the proposed algorithm to tradeoff among load balancing, TCP-friendliness, and responsiveness. Next section introduces maximization of the utility function to show that the proposed algorithm has optimal solution.

2.2 Utility Maximization

An utility function is associated to each subflow of the MPTCP source, which needs be maximized, in order to provide the high throughput. It can be represented as a constrained maximization problem, as given below.

$$P: \quad \max_{x_s} \sum_{s \in S} U_s(x_s) \tag{9}$$

$$\text{Subject to } y_l \le c_l \quad l \in L \tag{10}$$

where x_s is data rate vector of source flow $s \in S$ and $U_s : \mathbb{R}_+^{|s|} \to \mathbb{R}$ is a concave function.

The solution of problem P will provide the optimum source rate $x_s = \left(x_r, r \in R_s, s \in S\right)$. According to the duality model of TCP [22], if a utility function U_s associated with source flow s of an SPTCP ($s = 1$) and it is strictly concave, then the problem given in (9)–(10) will give a unique and optimal value of its data rate x_s. We now prove that our proposed OLBLIA algorithm has a unique and optimal data rate for each subflow $r \in R_s$ and $s \in S$.

Define the Lagrangian of P given in (9)–(10) as follows:

$$\begin{aligned} L(x,p) &= \sum_{s \in S} U_s(x_s) - \sum_{l \in L} p_l(y_l - c_l) \\ &= \sum_{s \in S} U_s(x_s) - \sum_{l \in L} p_l y_l + \sum_{l \in L} p_l c_l \\ &= \sum_{s \in S} U_s(x_s) - \sum_{l \in L} p_l \sum_{r \in R_s} A_{lr} x_r + \sum_{l \in L} p_l c_l \\ &= \sum_{s \in S} \left(U_s\left(x_s\right) - \sum_{r \in R_s} q_r x_r \right) + \sum_{l \in L} p_l c_l \end{aligned}$$

By the duality theory [26], (x^*, p^*) is primal–dual optimal if and only if x* is primal feasible, p* is dual feasible, and the complementary slackness condition holds as given below:

$$x^* = arg \max_{x_s \ge 0} L\left(x_s, p^*\right)$$

the primal problem is

$$\max_{x_s \ge 0} L\left(x_s, p^*\right) = \sum_{s \in S} \underset{x_s \ge 0}{max} \left(U_s\left(x_s\right) - x_s \sum_{r \in R_s} q_r^* \right) + \sum_{l \in L} p_l^* c_l$$

where

$$x_s = \sum_{r \in R_s} x_r, \ and \ q_r^* = \sum_{l \in L} p_l^* A_{lr}$$

and its Lagrangian dual is

$$\min_{p_l \geq 0} \sum_{s \in S} \max_{x_s \geq 0} \left(U_s(x_s) - x_s \sum_{r \in R_s} q_r \right) + \sum_{l \in L} p_l c_l$$

the KKT condition at optimality (x^*, p^*), $x_s^* > 0$, gives

$$U_s'(x_s^*) = q_r^* \tag{11}$$

For $x_s^* = 0$, it gives

$$U_s'(x_s^*) \leq q_r^* \tag{12}$$

Equations (11) and (12) give the following:

$$\frac{\partial L}{\partial x_s}(x^*, p^*) \leq 0$$

The above equality will hold for $x_s^* > 0$. Since $L(x, p^*)$ is concave in x, this is the necessary and sufficient KKT condition [25] for x^* to maximize $L(x, p^*)$ over $x \geq 0$. Thus, we have shown that x^* is the optimal point of $L(x, p^*)$. In other words, $L(x, p^*)$ provides the maximum value for $x = x^*$.

2.3 TCP-Friendliness

The scenario where the bottleneck link of network capacity c is shared by both SPTCP flow and MPTCP subflow, and no MPTCP subflow takes bandwidth more than an SPTCP subflow (c.f. Fig. 1). Such MPTCP flow is said to be TCP-friendly [14, 27]. Since the capacity of the bottleneck link is c, it means that all other links of the network have capacity strictly more than c. The links have fixed capacities but possibly different RTTs. Let a test network have two flows with window sizes as w_1 and w_2, respectively; initially with condition $w_1 \geq w_2$. Let δw_1 and δw_2 be the changes in w_1 and w_2, respectively. For fair allocation of bandwidth at equilibrium, the following condition should hold:

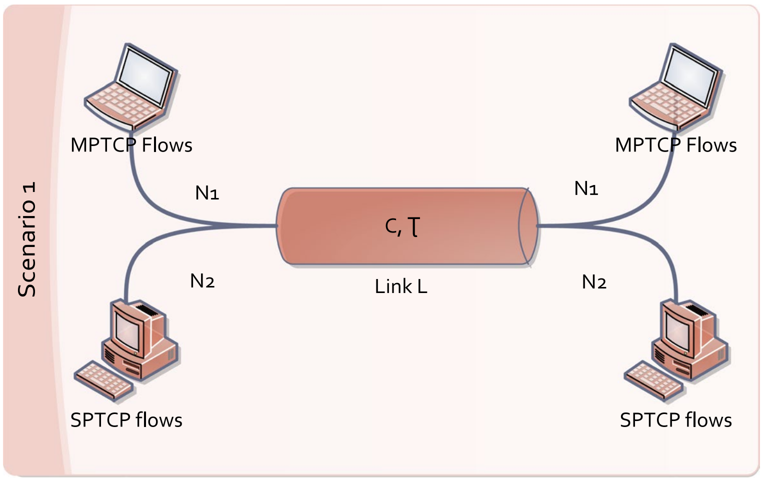

Fig. 1 SPTCP flows and MPTCP flows share the bottleneck link L

$$\mathrm{w}_1 + \delta \mathrm{w}_1 = \mathrm{w}_2 + \delta \mathrm{w}_2 \tag{13}$$

since $w_1 \geq w_2$, Eq. (13) gives $\delta w_1 \leq \delta w_2$.

Thus, we have

$$\frac{\delta \mathrm{w}_1}{\mathrm{w}_1} \leq \frac{\delta \mathrm{w}_2}{\mathrm{w}_2} \tag{14}$$

Equation (14) tells that if two flows are friendlier to each other, then the changes in their respective windows should be such that both get nearly the same throughput in stable state. More details can be seen in [14, 28]. Here, it has been discussed for two flows by considering a single algorithm, which can be SPTCP or MPTCP. Thus, if both the flows pass through a single link, then, for friendliness, both the flows sould have nearly the same throughput at equilibrium.

We now discuss the friendliness for two MPTCP algorithms. Let M_1 and M_2 be two MPTCP algorithms, which are executed on the same test network at different times. When M_1 shares the test network with SPTCP, assuming its aggregate throughput as X_1 over the available paths in equilibrium, the SPTCP attains its maximum possible throughput as $\left(T - X_1\right)$. When M_2 shares the test network with the same SPTCP, assuming its aggregate throughput as X_2, over the available paths in equilibrium. Here, T is the total throughput of the test network (i.e., $T = X_1 + X_2$). For $X_1 \geq X_2$, we say that M_1 is friendlier than M_2.

2.4 Responsiveness

In a system, the term responsiveness is defined as how fast the system converges to the equilibrium locally [14]. Here, we use the test networks, as shown in Figs. 1, 2, and 3, in which the MPTCP users share the link with SPTCP users. To demonstrate the dynamic performance (responsiveness) of each algorithm, assume that the SPTCP flows are alive for a limited period (say, 60–80 s), while the MPTCP subflows are alive for longer period (say, 0–200 s). Let M_1 and M_2 be two MPTCP algorithms that are executed on the same test network separately. We

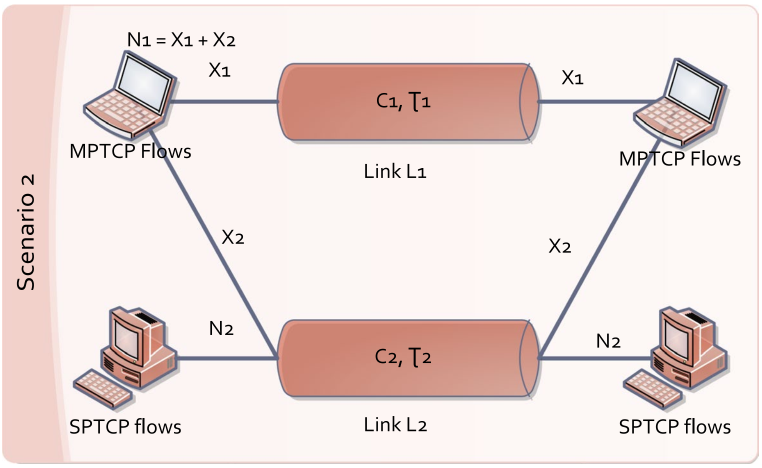

Fig. 2 MPTCP flows pass through the links L_1 and L_2 both and SPTCP flows pass through link L_2 only

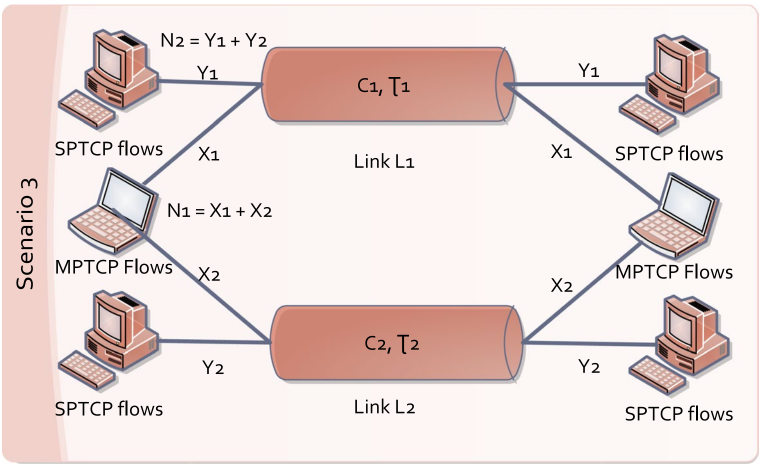

Fig. 3 Both SPTCP flows and MPTCP flows pass through both links L_1 and L_2

say M_1 is more responsive than M_2 if M_1 gets a stable stage more frequently than M_2, or how often M_1 recovers its flows, which are shared with the SPTCP (when SPTCP releases the link). The algorithm M_1 is more responsible than M_2 if the following condition holds:

$$\frac{\delta I_1(\mathrm{w_s})}{\delta D_1(\mathrm{w_s})} \geq \frac{\delta I_2(\mathrm{w_s})}{\delta D_2(\mathrm{w_s})} \tag{15}$$

where $\delta I_1(\mathrm{w_s})$ and $\delta I_2(\mathrm{w_s})$ are aggregate increments in CW of source s by M_1 and M_2 algorithms, respectively. $\delta D_1(\mathrm{w_s})$ and $\delta D_2(\mathrm{w_s})$ are the aggregate decrements in CW of source s by M_1 and M_2 algorithms, respectively.

The aggregate increment and aggregate decrement are calculated by the increment factor and decrement factor of each subflow of source s only. The algorithm M_1 is more responsible than M_2 when at least one of the three following conditions holds.

(a) The aggregate increment value of M_1 is more than that of M_2 and the aggregate decrement value of M_1 is less than that of M_2.
(b) The aggregate decrement value of M_1 is less than that of M_2, while the aggregate increment values of M_1 and M_2 are nearly the same.
(c) The aggregate increment value of M_1 is more than that of M_2, while the aggregate decrement values of M_1 and M_2 are nearly the same.

After discussing the utility maximization, friendliness, and responsiveness of the proposed OLBLIA algorithm, we now discuss its experimental results.

3 Experimental Results

This section presents the simulation setup, followed by a detailed discussion of the experimental results. In order to assess how well the proposed algorithm performs, we compare it with BALIA, LIA, and OLIA MPTCP algorithms. In literature, it has been shown that

BALIA, OLIA, and LIA algorithms outperform other existing algorithms such as uncoupled and semicoupled algorithms. Therefore, we will compare the performance of our algorithm with that of these algorithms.

3.1 Simulation Setup

Here, three different scenarios are created based on sharing of the links to evaluate the performance of the proposed OLBLIA algorithm. For simulation setup, we used ns3 [29]. In first scenario, there is a single link shared between the MPTCP users and SPTCP users, as shown in Fig. 1. In second scenario, there are two links out of which one link is allocated to MPTCP users only and other link is shared among the MPTCP and SPTCP users, as shown in Fig. 2. In third scenario, both links are shared among the MPTCP and SPTCP users, as shown in Fig. 3.

Figure 1 depicts scenario one, where the SPTCP users and MPTCP users transfer the data via the same link (L) that has maximum bandwidth C, with round-trip-time τ. Here, the total number of subflows the MPTCP users and SPTCP users are N_1 and N_2 respectively. The MPTCP users can send the data having one or multiple subflows through link L, which may or may not be shared with the SPTCP.

Figure 2 depicts scenario two, where the link L_1 (maximum bandwidth C_1 and round-trip-time τ_1) is dedicated to only MPTCP users. The second link L_2 (maximum bandwidth C_2 with round-trip-time τ_2) is used to transfer the data of the SPTCP users and MPTCP users. For MPTCP users, X_1 number of subflows out of N_1 subflows coresponds to the link L_1 and the remaining X_2 number of subflows to the link L_2. For SPTCP users, the total N_2 flows (one flow per TCP user) correspond to the link L_2.

Figure 3 depicts third scenario, where both links L_1 and L_2 are shared among the MPTCP and SPTCP users to transfer the data. For MPTCP users, X_1 number of subflows out of N_1 corresponds to the link L_1 and the remaining X_2 number of subflows corresponds to the link L_2. For SPTCP users, Y_1 number of subflows out of N_2 flows corresponds to the link L_1 and the rest Y_2 number of subflows corresponds to the link L_2.

The simulations are performed under above discussed scenarios by varying the number of nodes (for SPTCP flows and MPTCP subflows ranging from $1, 2, \ldots, 25$), link capacity, and RTT. The link capacity in simulation is considered as 10 Mbps with a RTT of 5 ms for each subflow/flow. The queue size is 100 packets at each node monitored by the RED active queue management algorithm. The simulations are performed for 0–200 s for each case of each scenario. We discuss the performance of the proposed OLBLIA algorithm and compare it with that of the existing MPTCP algorithms: BALIA, OLIA, and LIA, in terms of the throughput, responsiveness, and friendliness.

3.2 Throughput

The rate of successful messages delivery in a network over a communication channel is defined as the throughput. It can be measured in bits per second (bit/s or bps) or in data packets per time slot or data packets per second (p/s or pps). The proposed OLBLIA algorithm is used to compute the throughput for measuring the network performance by varying the number of subflows. We also compute the throughput of BALIA, OLIA, and LIA algorithms for comparison purpose.

In first and second scenarios each, three cases are considered and in third scenario two cases as shown in Table 2.

Case 1 of first scenario has one MPTCP subflow and one SPTCP flow. In this case, the OLBLIA algorithm provides the same throughput as the BALIA, OLIA, and LIA algorithms, which is same as that of the SPTCP (i.e., 4.87 Mbps). Here, it can be seen that none of MPTCP algorithm i.e., proposed OLBLIA, LIA. OLIA and BALIA algorithms is not aggressive towards the SPTCP because their throughputs are bounded by the throughput of the SPTCP, i.e., 4.87 Mbps. Thus, they are not unfriendly towards SPTCP.

Case 2 of first scenario has 10 MPTCP subflows and 5 SPTCP flows. Here, none of the MPTCP algorithms should have throughput more than 10/3, i.e., 6.67 Mbps. If any MPTCP algorithm has throughput more than 6.67 Mbps, then that algorithm will become aggressive towards the SPTCP. The OLBLIA, BALIA, OLIA, and LIA algorithms have throughputs as 6.56, 7.33, 7.29 and 7.32 Mbps, as shown in Table 2. Thus, the existing algorithms i.e., BALIA, OLIA, and LIA are aggressive towards the SPTCP. These algorithms in turn degrade the performance of the SPTCP, which is against the TCP-friendliness.

Case 3 of *first scenario* has 20 MPTCP subflows and 5 SPTCP flows. The OLBLIA, BALIA, OLIA, and LIA algorithms have throughputs as 7.98, 7.52, 7.53, and 7.52 Mbps; however, none of the MPTCP algorithms should have throughput more than 10 * 20/25, i.e., 8 Mbps, otherwise that algorithm will become aggressive towards SPTCP. Here,

Table 2 Throughput (in Mbps) of OLBLIA, BALIA, OLIA, and LIA algorithms [RTT (τ) as 5 ms, capacity of each link as 10 Mbps]

Case		LIA	OLIA	BALIA	Proposed OLBLIA
Scenario 1					
1	MPTCP (1)	4.87	4.87	4.87	4.87
	SPTCP (1)	4.87	4.87	4.87	4.87
2	MPTCP(10)	7.32	7.29	7.33	6.56
	SPTCP (5)	2.62	2.64	2.61	3.42
3	MPTCP (20)	7.52	7.53	7.52	7.98
	SPTCP (5)	2.26	2.25	2.26	1.97
Scenario 2					
1	MPTCP(10) $\{X_1=5, X_2=5\}$	15.12 (9.78, 5.34)	14.55 (9.27, 5.28)	15.18 (9.79, 5.39)	14.69 (9.74, 4.95)
	SPTCP (5)	4.59	4.65	4.54	4.94
2	MPTCP (20) $\{X_1=15, X_2=5\}$	14.96 (9.80, 5.16)	14.40 (9.32, 5.08)	15.03 (9.85, 5.18)	14.89 (9.96, 4.93)
	SPTCP (5)	4.34	4.42	4.36	4.94
3	MPTCP (25) $\{X_1=20, X_2=5\}$	12.00 (9.95, 2.05)	11.96 (9.91, 2.05)	11.99 (9.95, 2.04)	13.19 (9.95, 3.24)
	SPTCP (10)	7.93	7.92	7.94	6.64
Scenario 3					
1	MPTCP(10) $\{X_1=5, X_2=5\}$	10.66 (5.32, 5.34)	10.39 (5.18, 5.21)	10.42 (5.22, 5.20)	9.86 (4.92, 4.94)
	SPTCP (10) $\{Y_1=5, Y_2=5\}$	9.30 (4.69, 4.61)	9.46 (4.72, 4.74)	9.06 (4.32, 4.74)	9.85 (4.93, 4.92)
2	MPTCP (25) $\{X_1=15, X_2=10\}$	14.00 (7.02, 6.98)	13.86 (6.94, 6.92)	14.08 (7.07, 7.01)	14.06 (7.47, 6.59)
	SPTCP (10) $\{Y_1=5, Y_2=5\}$	5.51 (2.63, 2.88)	5.48 (2.62, 2.86)	5.45 (2.61, 2.84)	5.70 (2.41, 3.29)

all algorithms have throughput less than 8 Mbps, i.e., none of the MPTCP algorithms is aggressive towards the SPTCP. But, in this case, the OLBLIA provides better throughput than the BALIA, OLIA, and LIA algorithms and hence it utilizes the MPTCP subflows better than the existing algorithms, i.e., BALIA, OLIA, and LIA.

We now consider second scenario that also has three cases.

In *case 1 of second scenario*, there are 10 MPTCP subflows and 5 SPTCP flows in which $X_1 = 5$ subflows of the MPTCP user pass through the link L_1 and the rest $X_2 = 5$ subflows pass through the link L_2. Here, the throughput of any MPTCP algorithm for link L_2 should not be more than 5 Mbps, which is same as that of SPTCP. The throughputs of MPTCP subflows through link L_2 using the proposed OLBLIA, LIA. OLIA and BALIA algorithms are 4.95, 5.34, 5.28, and 5.39 Mbps, respectively, as shown in Table 2. Here, it can be seen that the existing algorithms i.e., LIA. OLIA and BALIA algorithms are aggressive towards the SPTCP because their throughputs are more than that of the SPTCP, i.e., 5 Mbps. So, these algorithms degrade the performance of SPTCP, which is against the TCP-friendliness. For our OLBLIA algorithm, the throughput of MPTCP is 4.95 Mbps, which is very close to 5 Mbps. Further, the throughput of the SPTCP should be as close as possible to 5 Mbps. However, the throughput of the SPTCP for the LIA, OLIA, and BALIA algorithms are 4.59, 4.65, and 4.54 Mbps, respectively; whereas, the throughput of the SPTCP for our proposed OLBLIA is 4.94 Mbps. Thus, our algorithm performs better than the existing algorithms in this case also.

In *case 2 of second scenario*, there are 20 MPTCP subflows and 5 SPTCP flows in which $X_1 = 15$ subflows of the MPTCP user pass through the link L_1 and the rest $X_2 = 5$ subflows pass through the link L_2. Here also, the throughput of any MPTCP algorithm for link L_2 should not be more than 5 Mbps, which is same as that of SPTCP. The throughputs of MPTCP subflows through link L_2 using the proposed OLBLIA, LIA. OLIA and BALIA algorithms are 4.93, 5.16, 5.08 and 5.18 Mbps, respectively, as shown in Table 2. Here, it can be seen that the existing algorithms i.e., LIA. OLIA and BALIA algorithms are aggressive towards the SPTCP because their throughputs are more than that of the SPTCP, i.e., 5 Mbps. So, these algorithms degrade the performance of SPTCP, which is against the TCP-friendliness. For our OLBLIA algorithm, the throughput of MPTCP is 4.93 Mbps, which is very close to 5 Mbps. Further, the throughput of the SPTCP should be as close as possible to 5 Mbps. However, the throughput of the SPTCP for the LIA, OLIA, and BALIA algorithms are 4.34, 4.42, and 4.36 Mbps, respectively; whereas, the throughput of the SPTCP for our proposed OLBLIA is 4.94 Mbps. Thus, our algorithm performs better than the existing algorithms.

In *case 3 of second scenario*, there are 25 MPTCP subflows and 10 SPTCP flows in which $X_1 = 20$ subflows of the MPTCP user pass through the link L_1 and the rest $X_2 = 5$ subflows pass through the link L_2. Here, the throughput of any MPTCP algorithm for link L_2 should not be more than $10 * 5/15 = 3.33$ Mbps. The throughputs of MPTCP subflows through link L_2 using the proposed OLBLIA, LIA. OLIA and BALIA algorithms are 3.24, 2.05, 2.05, and 2.04 Mbps, respectively, as shown in Table 2. Here, it can be seen that the existing algorithms i.e., LIA. OLIA and BALIA algorithms are under-utilize the MPTCP subflows; whereas, the proposed OLBLIA utilizes them in a better way. Further, the throughput of the SPTCP users should be as close as possible to $10 * 10/15 = 6.66$ Mbps. However, the throughput of the SPTCP for the LIA, OLIA, and BALIA algorithms are 7.93, 7.92, and 7.94 Mbps, respectively; whereas, the throughput of the SPTCP for our proposed OLBLIA is 6.64 Mbps. Thus, our algorithm performs better than the existing algorithms.

We now discuss third scenario that has two cases.

In *case 1 of third scenario*, there are 10 MPTCP subflows and 10 SPTCP flows in which X_1 = 5 subflows of the MPTCP user and Y_1 = 5 flows of the SPTCP users pass through the link L_1 and the rest X_2 = 5 subflows of MPTCP user and Y_2 = 5 flows of the SPTCP users pass through the link L_2. Here, the throughput of any MPTCP algorithm for each link L_1 and L_2 should not be more than 10 * 5/10 = 5 Mbps. The throughputs of the MPTCP subflows through link L_1 using the proposed OLBLIA, LIA. OLIA and BALIA algorithms are 4.92, 5.32, 5.18, and 5.22 Mbps, respectively, and through the link L_2 are 4.94, 5.34, 5.21, and 5.20, respectively, as shown in Table 2. Here, it can be seen that the existing algorithms i.e., LIA. OLIA and BALIA algorithms are aggressive towards the SPTCP because their throughputs are more than 5 Mbps. So, these algorithms degrade the performance of the SPTCP, which is against the TCP-friendliness. For our OLBLIA algorithm, the throughput of MPTCP through the links L_1 and L_2 are 4.92 and 4.94 Mbps, respectively, which are very close to 5.0 Mbps. Further, the throughput of the SPTCP should be as close as possible to 10 * 5/10 = 5.0 Mbps for each of the links L_1 and L_2. However, the throughput of the SPTCP for the OLBLIA, LIA, OLIA, and BALIA algorithms through the link L_1 are 4.93, 4.69, 4.72, and 4.32 Mbps, respectively, and through the link L_2 are 4.92, 4.61, 4.72, and 4.74 Mbps, respectively. Thus, our algorithm performs better than the existing algorithms, i.e., LIA, OLIA, and BALIA.

In *case 2 of third scenario*, there are 25 MPTCP subflows and 10 SPTCP flows in which X_1 = 15 subflows of the MPTCP user and Y_1 = 5 flows of the SPTCP users pass through the link L_1 and the rest X_2 = 10 subflows of the MPTCP user and Y_2 = 5 flows of the SPTCP users pass through the link L_2. Here, the throughput of any MPTCP algorithm for link L_1 should not be more than 10 * 15/20 = 7.5 Mbps and, for link L_2 it should not be more than 10 * 10/15 = 6.66 Mbps. The throughputs of the MPTCP subflows through the link L_1 using the proposed OLBLIA, LIA, OLIA, and BALIA algorithms are 7.47, 7.02, 6.94, and 7.07 Mbps, respectively, as shown in Table 2. Here, it can be seen that the existing algorithms i.e., LIA. OLIA and BALIA algorithms under-utilize the MPTCP subflows; whereas, our OLBLIA algorithm utilizes in a better way. The throughputs of the MPTCP subflows through the link L_2 using the proposed OLBLIA, LIA. OLIA and BALIA algorithms are 6.59, 6.98, 6.92, and 7.01, respectively. Here, it can be seen that the existing algorithms i.e., LIA. OLIA and BALIA algorithms are aggressive towards the SPTCP because their throughputs are more than the 6.66 Mbps. So, these algorithms degrade the performance of the SPTCP, which is against the TCP-friendliness. For our OLBLIA algorithm, the throughput of the MPTCP is 6.59 Mbps, which is very close to 6.66 Mbps. Further, the throughput of the SPTCP should be as close as possible to 10 * 5/20 = 2.5 Mbps for link L_1 and 10 * 5/15 = 3.33 Mbps for link L_2. However, the throughput of the SPTCP for the OLBLIA, LIA, OLIA, and BALIA algorithms through the link L_1 are 2.41, 2.63, 2.62, and 2.61 Mbps, respectively, and through the link L_2 are 3.29, 2.88, 2.86, and 2.84 Mbps, respectively. Thus, our algorithm performs better than the existing algorithms.

3.3 TCP-Friendliness

Here we discuss the friendliness of our proposed OLBLIA algorithm and compare its performance with that of the BALIA, OLIA, and LIA algorithms that involves the number of MPTCP subflows and SPTCP flows. It is assumed that all MPTCP subflows and SPTCP subflows are long-lived. In *case 1 of first scenario* (which has one MPTCP subflow and one SPTCP flow, sharing the common link L), the OLBLIA algorithm does not take more bandwidth than the SPTCP although it gives the throughput equal to that of SPTCP (i.e., 4.87 Mbps) as shown in Table 2. Thus, the OLBLIA shows good friendliness towards the SPTCP. In other scenarios including the rest all cases, the proposed OLBLIA algorithm shows better TCP-friendliness (as discussed in subsection B) than the BALIA, OLIA, and LIA as depicted in Table 2.

The friendliness behavior of the OLBLIA, BALIA, OLIA, and LIA algorithms for *case 1 of second scenario* is shown graphically in Fig. 4a–h. Figures 4a, c, e, g show the throughputs of the MPTCP subflows and SPTCP flows passing through different links and the corresponding average throughputs are shown in Fig. 4b, d, f, h. Here, the 'MPTCP throughput' refers to the overall throughput of the MPTCP user by including the throughputs of all the MPTCP subflows through each link, while the 'MPTCP Link L_1', 'MPTCP Link L_2', and 'SPTCP throughput' are the throughputs of all the MPTCP subflows passing through the links L_1, link L_2, and all SPTCP flows passing through the link L_2, respectively.

It can be seen from Fig. 4 that the throughput of 'MPTCP Link L_2' and 'SPTCP throughput' are much closer for the OLBLIA than that of the BALIA, OLIA, and LIA. It is because the number of SPTCP flows and the number of MPTCP subflows passing through the link L_2 are equal, and the OLBLIA balances the data load by distributing the data through the MPTCP subflows that passes through the link L_2 so that they do not take more bandwidth than the SPTCP flows (which can be atmost equal to the SPTCP flows). On the other hand, the BALIA, OLIA, and LIA algorithms are not as friendly as the OLBLIA because they take more bandwidth than the SPTCP flows. Thus, the proposed OLBLIA is more TCP-Friendly than the LIA, OLIA, and BALIA algorithms. We have carried out several experiments by varying different parameters and got the results of similar nature as shown in Table 2. Therefore, due to the similar trend in the results in almost all the scenarios, we have not shown all of them graphically.

3.4 Responsiveness

Here, we discuss the responsiveness of the proposed OLBLIA and compare it with that of the BALIA, OLIA, and LIA algorithms. The process involves varying the numbers of MPTCP subflows and SPTCP flows, using the dynamic environment by starting the SPTCP flows at 60th second and terminating at 80th second, and the MPTCP subflows starting from the beginning (i.e., 0th second) and continuing till end of the simulation (i.e., 200th second). The OLBLIA algorithm has better responsiveness than the BALIA, OLIA, and LIA algorithms for all three scenarios discussed in subsection A, as depicted in Table 3.

In *scenario 1*, 20 MPTCP subflows and 5 SPTCP flows have been considered, and all the MPTCP subflows and SPTCP flows pass through a common link. The OLBLIA provides much better convergence time than the BALIA, OLIA, and LIA algorithms as their respective convergence times are 1, 19, 24, and 25 s. In *scenario 2*, 10 MPTCP subflows and 5 SPTCP flows have been considered in which $X_1 = 5$ MPTCP subflows pass through the link L_1 and $X_2 = 5$ MPTCP subflows pass through the link L_2, while all the SPTCP flows pass through the link L_2. Here also, the OLBLIA provides better convergence time than the BALIA, OLIA, and LIA algorithms as their respective convergence times are 15, 25, 110, and 28 s. In *scenario 3*, 10 MPTCP subflows and 10 SPTCP flows have been considered in which $X_1 = 5$ subflows of the MPTCP user and $Y_1 = 5$ flows of the SPTCP users pass through the link L_1 and the rest $X_2 = 5$ subflows of MPTCP user and $Y_2 = 5$ flows of the SPTCP users pass through the link L_2. In this case also, the OLBLIA provides better convergence time than the BALIA, OLIA, and LIA algorithms as their respective convergence times are 10, 32, 64, and 31 s. Since the proposed OLBLIA algorithm has better

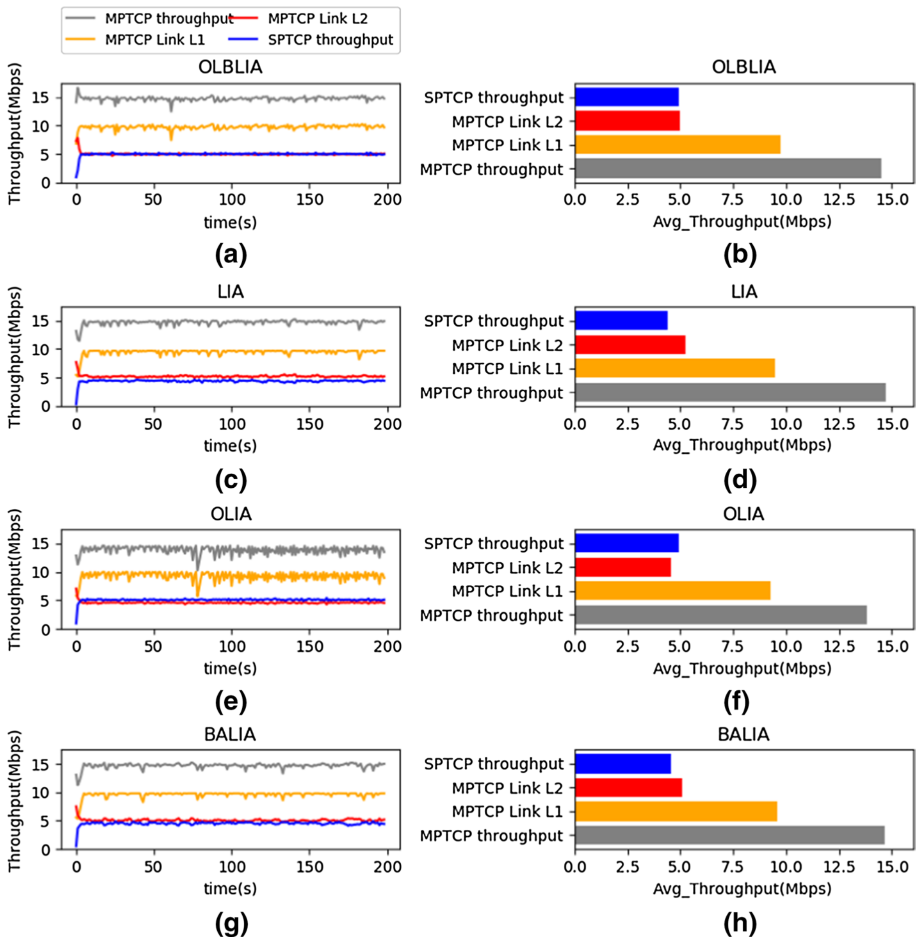

Fig. 4 Comparison of TCP-friendliness in terms of: **a** throughput using OLBLIA, **b** average throughput using OLBLIA, **c** throughput using LIA, **d** average throughput using LIA, **e** throughput using OLIA, **f** average throughput using OLIA, **g** throughput using BALIA, and **h** average throughput using BALIA

Table 3 Responsiveness (SPTCP starts at 60th second and stops at 80th second)

	LIA	OLIA	BALIA	Proposed OLBLIA
Convergence time				
Scenario 1	25	24	19	1
Scenario 2	28	110	25	15
Scenario 3	31	64	32	10

convergence time than the BALIA, OLIA, and LIA algorithms in all scenarios, it provides better responsiveness.

The simulations have been carried out for responsiveness by considering a large number of cases in each of the three scenarios by varying the number of MPTCP subflows, SPTCP flows, RTT and link capacity, as given in Table 2. In almost all cases, similar types of the results have been obtained. Therefore, we have not shown all the results due to their repetitive nature.

The responsiveness of the OLBLIA algorithm along with that of the BALIA, OLIA, and LIA algorithms for *scenario 2* are shown graphically in Fig. 5. Figure 5a, c, e, g show the throughput of the MPTCP subflows and SPTCP flows through different links; and Fig. 5b, d, f, h show the total throughput of all the MPTCP subflows and SPTCP flows using the OLBLIA, LIA, OLIA, and BALIA, respectively. The 'Total throughput' shows the overall throughput of the network by including the throughput of all the MPTCP subflows and SPTCP flows through each link, while the 'MPTCP Link L_1', 'MPTCP Link L_2', and 'SPTCP throughput' are the throughputs of all the MPTCP subflows passing through link L_1, link L_2, and all SPTCP flows passing through the link L_2, respectively. As evident from Fig. 5, the OLBLIA algorithm gets its stable state very quickly (in 15 s) as compared to

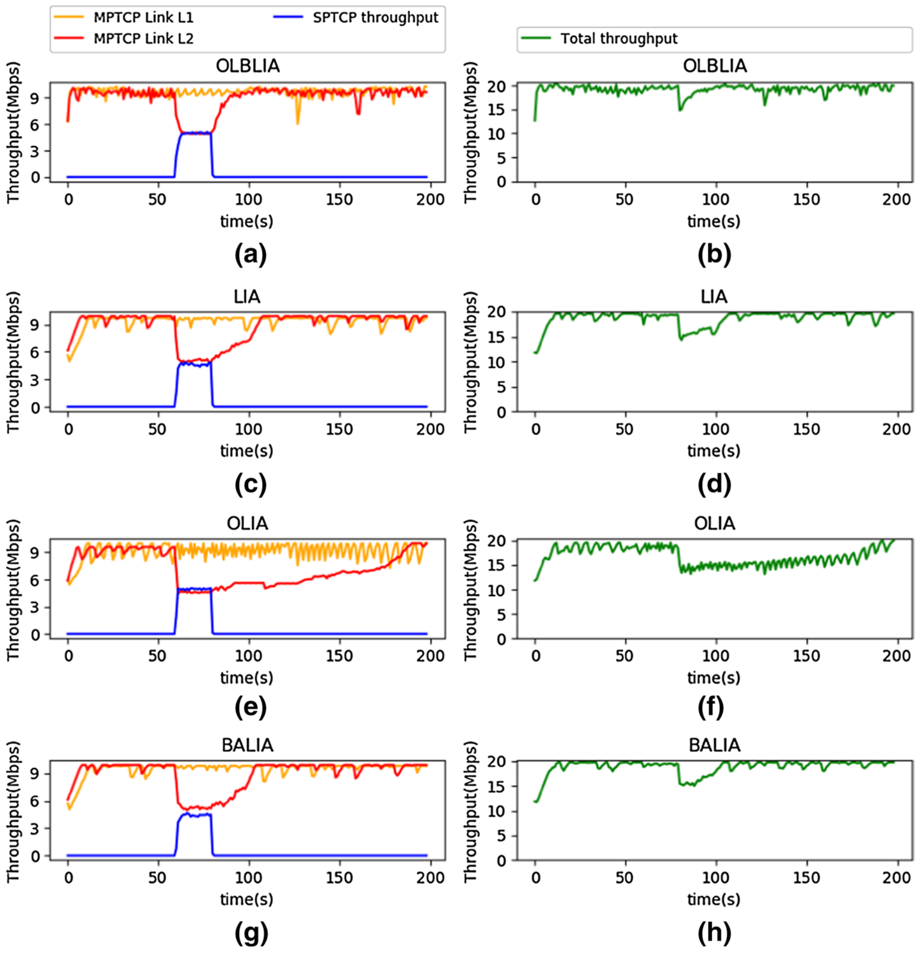

Fig. 5 Comparison of responsiveness in terms of **a** throughput using OLBLIA, **b** total throughput using OLBLIA, **c** throughput using LIA, **d** total throughput using LIA, **e** throughput using OLIA, **f** total throughput using OLIA, **g** throughput using BALIA, and (h) total throughput using BALIA

the LIA (in 28 s), OLIA (in 110 s), and BALIA (in 25 s) algorithms after terminating the 'SPTCP flows' (at 80th second). Thus, Fig. 5 and Table 3 show that the OLBLIA is more responsive than the BALIA, OLIA, and LIA algorithms. Thus, the experimental results demonstrate the effectiveness of our proposed OLBLIA in terms of throughput, responsiveness, TCP-friendliness and tradeoff among them, when compared with the BALIA, OLIA, and LIA algorithm.

4 Conclusion

The paper has analyzed the existing congestion control algorithms of MPTCP, with the intent of observing the current issues like load balancing, TCP-friendliness, responsiveness, and tradeoff among them. In order to overcome these issues, a new congestion control algorithm for MPTCP, OLBLIA, has been discussed. Experimentally and analytically, it has been shown that the OLBLIA outperforms the existing MPTCP algorithms. It allocates more data to a less congested path and less data to a more congested path. It achieves high throughput by reducing the loss of packets and provides better load balancing. It also tries to utilize the freely available bandwidth of a path in a better way that makes it more responsive. The overall performance of MPTCP increases without affecting the performance of SPTCP. Further, it provides better responsiveness and more throughput, while maintaining the TCP-friendliness.

Acknowledgements This work is supported by UPE-II, Jawaharlal Nehru University, Delhi. Authors like to acknowledge the Council of Scientific & Industrial Research (CSIR) for providing the Fellowship throughout this research.

References

1. Robinson, Y. H., & Julie, E. G. (2019). SMR: A synchronized multipath re-broadcasting mechanism for improving the quality of conversational video service. *Wireless Personal Communications, 104*(3), 1149–1173.
2. Laghari, A. A., He, H., Shafiq, M., & Khan, A. (2018). Application of quality of experience in networked services: Review, trend & perspectives. *Systemic Practice and Action Research*. https://doi.org/10.4108/eai.13-7-2018.160390.
3. Chaturvedi, R. K., & Chand, S. (2018). MPTCP over datacenter networks. In *2018 second international conference on inventive communication and computational technologies (ICICCT)* (pp. 894–898).
4. Ford, A., Raiciu, C., Handley, M., & Bonaventure, O. (2012). TCP extensions for multipath operation with multiple addresses, draft-ietf-mptcp-multiaddressed-09. *Internetdraft, IETF* (March 2012).
5. Scharf, M., & Ford, A. (2013). Multipath TCP (MPTCP) application interface considerations. RFC 6897, March.
6. Liu, Y., Neri, A., Ruggeri, A., & Vegni, A. M. (2016). A MPTCP-based network architecture for intelligent train control and traffic management operations. *IEEE Transactions on Intelligent Transportation Systems, 18*(9), 2290–2302.
7. Syariati, F. M., & Choi, K. W. (2019). Optimal concurrent multipath data transfer for bandwidth aggregation in heterogeneous mobile networks. *Wireless Personal Communications, 107,* 1383–1400.
8. De Schepper, T., Latré, S., & Famaey, J. (2019). Scalable load balancing and flow management in dynamic heterogeneous wireless networks. *Journal of Network and Systems Management*. https://doi.org/10.1007/s10922-019-09502-2.

9. Chow, A. L., Yang, H., Xia, C. H., Kim, M., Liu, Z., & Lei, H. (2009). EMS: Encoded multipath streaming for real-time live streaming applications. In *2009 17th IEEE international conference on network protocols* (pp. 233–243).
10. Lu, Y. (2016). SED: An SDN-based explicit-deadline-aware TCP for cloud data center networks. *Tsinghua Science and Technology, 21*(5), 491–499.
11. Pang, J., Xu, G., & Fu, X. (2017). SDN-based data center networking with collaboration of multipath TCP and segment routing. *IEEE Access, 5,* 9764–9773.
12. Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., & Handley, M. (2011). Improving datacenter performance and robustness with multipath TCP. In *ACM SIGCOMM computer communication review* (Vol. 41, No. 4, pp. 266–277).
13. Ha, S., Rhee, I., & Xu, L. (2008). CUBIC: A new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review, 42*(5), 64–74.
14. Sastry, N. R., & Lam, S. S. (2005). CYRF: A theory of window-based unicast congestion control. *IEEE/ACM Transactions on Networking (TON), 13*(2), 330–342.
15. Ford, A., Raiciu, C., Handley, M., Barre, S., & Iyengar, J. (2011). Architectural guidelines for multipath TCP development. In *IETF, RFC 6182*.
16. Hurtig, P., Grinnemo, K. J., Brunstrom, A., Ferlin, S., Alay, Ö., & Kuhn, N. (2018). Low-latency scheduling in MPTCP. *IEEE/ACM Transactions on Networking, 27*(1), 302–315.
17. Blanton, E., & Allman, M. (2009). TCP congestion control. *IETF, Standards Track RFC 5681.*
18. Raiciu, C., Handley, M., & Wischik, D. (2011). RFC 6356, coupled congestion control for multipath transport protocols.
19. Paasch, C., Khalili, R., & Bonaventure, O. (2013). On the benefits of applying experimental design to improve multipath TCP. In *Proceedings of the ninth ACM conference on emerging networking experiments and technologies* (pp. 393–398).
20. Khalili, R., Gast, N., Popovic, M., & Le Boudec, J. Y. (2012). Performance issues with MPTCP. *draft-khalili-mptcpperformance-issues-04*.
21. Walid, A., Peng, Q., Hwang, J., & Low, S. (2016). Balanced linked adaptation congestion control algorithm for MPTCP. *Working Draft, IETF Secretariat, Internet-Draft draft-walid-mptcp-congestion-control-04*.
22. Khalili, R., Gast, N., & Popovic, M. (2013). Opportunistic linked-increases congestion control algorithm for MPTCP. *draft-khalili-mptcp-congestion-control-02*.
23. Kelly, F., & Voice, T. (2005). Stability of end-to-end algorithms for joint routing and rate control. *ACM SIGCOMM Computer Communication Review, 35*(2), 5–12.
24. Khalili, R., Gast, N., Popovic, M., & Le Boudec, J. Y. (2013). MPTCP is not pareto-optimal: Performance issues and a possible solution. *IEEE/ACM Transactions on Networking (ToN), 21*(5), 1651–1665.
25. Low, S. H. (2003). A duality model of TCP and queue management algorithms. *IEEE/ACM Transactions on Networking (ToN), 11*(4), 525–536.
26. Bertsekas, D. P. (1997). Nonlinear programming. *Journal of the Operational Research Society, 48*(3), 332–334.
27. Vo, P. L., Le, T. A., & Tran, N. H. (2019). mFAST: A multipath congestion control protocol for high bandwidth-delay connection. *Mobile Networks and Applications, 24*(1), 115–123.
28. Lee, J., Im, Y., & Lee, J. (2019). Modeling MPTCP performance. *IEEE Communications Letters, 23*(4), 616–619.
29. *Network Simulator (NS-3)*. http://www.nsnam.org. Accessed 2009.

Rajnish Kumar Chaturvedi received the B.Tech and M.Tech. degrees in Computer Science and Engineering from Dr. A.P.J. Abdul Kalam Technical University (formerly UPTU), Lucknow, India, in 2014, and is currently pursuing the Ph.D. degree in computer science at the Jawaharlal Nehru University, New Delhi, India. He has been with INMAS, DRDO, New Delhi, India, as a research scholar. His current research interests include congestion control in networking, multipath TCP, and transport protocols.

Satish Chand did his M.Sc. in Mathematics from IIT Kanpur, M.Tech. in Computer Science from IIT Kharagpur, Ph.D. in Computer Science from Jawaharlal Nehru University, Delhi. Presently he is working as Professor in the School of Computer and Systems Sciences, Jawaharlal Nehru University Delhi, India. His areas of interest are image processing, video processing, computer networks, sensor networks, etc.