Low Priority Congestion Control for Multipath TCP

Yuan Zhang†, Jian Li‡*, Jiayu Yang‡, Yitao Xing‡, Rui Zhuang‡, Kaiping Xue†‡*
† Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027 China
‡ School of Cyber Security, University of Science and Technology of China, Hefei, Anhui 230027 China
* Corresponding author, lijian9@ustc.edu.cn, kpxue@ustc.edu.cn

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9686024

***Abstract*—Many applications are bandwidth consuming but may tolerate longer flow completion times. Multipath protocols, such as multipath TCP (MPTCP), can offer bandwidth aggregation and resilience to link failures for such applications, and low priority congestion control (LPCC) mechanisms can make these applications yield to other time-sensitive ones. Properly combining the above two can improve the overall user experience. However, the existing LPCC mechanisms are not adequate for MPTCP. They do not take into account the characteristics of multiple network paths, and cannot ensure fairness among the same priority flows. Therefore, we propose a multipath LPCC mechanism, i.e., Dynamic Coupled Low Extra Delay Background Transport, named DC-LEDBAT. Our scheme is designed based on a standardized LPCC mechanism LEDBAT. To avoid unfairness among the same priority flows, DC-LEDBAT trades little throughput for precisely measuring the minimum delay. Moreover, to be friendly to single-path LEDBAT, our scheme leverages the correlation of the queuing delay to detect whether multiple paths go through a shared bottleneck. Then, DC-LEDBAT couples the congestion window at shared bottlenecks to control the sending rate. We implement DC-LEDBAT in a Linux kernel and experimental results show that DC-LEDBAT can not only utilize the excess bandwidth of MPTCP but also ensure fairness among the same priority flows.**

***Index Terms*—multipath TCP, congestion control, less-than-best-effort service**

I. Introduction

Among the diverse Internet applications, some widely used applications such as software updates, online system backup, and peer-to-peer applications are bandwidth consuming but not time-sensitive. The best-effort service provided by traditional congestion control mechanisms, aiming to have a fair share of bandwidth on a common bottleneck, is not adequate for such applications at the network-wide level. A less-than-best-effort (LBE) service provided by low priority congestion control (LPCC) mechanisms is an attractive choice for these applications [1]. LPCC mechanism utilizes only the excess bandwidth by detecting congestion early and yields to high priority flows when sharing bottlenecks. This makes LPCC mechanisms competent to transfer background traffic and promising to improve network utilization. Among all the LPCC mechanisms, Low Extra Delay Background Transport (LEDBAT) [2] is the most popular one and similar algorithms have been implemented and deployed in Apple devices for software updates and BitTorrent's uTP protocol.

However, with the popularity of bandwidth consuming applications, the applications using LPCC mechanisms may starve for a long time. Although these applications may tolerate longer completion times than others, unbearable waiting can also affect the user experience. For example, Alice is using WiFi to update software with LPCC mechanisms while Bob is enjoying high-definition live video streaming via the same WiFi access point. Due to the limited bandwidth provided by the access point and the low priority characteristics of LPCC mechanisms, Alice has to wait until Bob finishes watching.

With the prevalence of multihomed devices, using multiple network interfaces to expand the available bandwidth is a workable solution to the above problem. As an extension of regular TCP, Multipath TCP (MPTCP) makes it possible to use multiple network paths for more efficient resource utilization [3]. Besides, as an IETF standard, MPTCP is already commercialized by major software vendors, including KT and Apple. Using MPTCP to provide an LBE service can offer bandwidth aggregation for background traffic and significantly reduce flow completion times. Meanwhile, MPTCP can offer resilience to link failures for background traffic transport. Further, a multipath LBE service can take full advantage of the excess bandwidth without having a big impact on other high priority flows, just like a bandwidth scavenger.

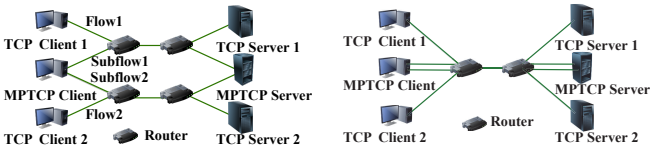
However, the existing LPCC mechanisms are not adequate for MPTCP. Specifically, these approaches do not take into account the characteristics of multiple network paths and cannot guarantee the fairness among same priority flows. For example, as the most popular LPCC mechanism, LEDBAT limits the queuing delay not to exceeding a predefined threshold to utilize the excess bandwidth, and it relies heavily on the correct measurement of the base delay to achieve its low priority characteristics. Unfortunately, LEDBAT lacks a mechanism to measure the base delay correctly, thus it suffers from latecomer's unfairness [4], [5]. Besides, LEDBAT does not take into account the multipath characteristics. When a single-path flow shares bottlenecks with more than one multipath subflow, these subflows take more resources than the single-path flow.

A desired multipath LPCC mechanism should satisfy the following constraints: (1) A multipath low priority flow should make full use of the excess bandwidth on all available network paths and quickly yield to standard TCP flows when they are sharing bottlenecks. (2) A multipath low priority flow should remain fair to other flows of the same priority when they are sharing one or more bottlenecks, as required by the multipath congestion control constraint [6]: the use of multiple paths must not unduly harm the single-path flow at shared bottlenecks.

In this paper, we propose a novel multipath LPCC mech-

978-1-7281-8104-2/21/$31.00 ©2021 IEEE

anism named Dynamic Coupled LEDBAT (DC-LEDBAT). Firstly, DC-LEDBAT trades little throughput for a precise measurement of the base delay, which completely solves the problem of latecomer's unfairness inherited from LEDBAT. Secondly, inspired by shared bottleneck detection mechanisms designed for best-effort services [7], we design a queuing delay-based bottleneck detection mechanism specifically for low priority subflows. And then DC-LEDBAT uses a shared bottleneck-based coupled congestion window (CWND) management mechanism to satisfy the multipath congestion control constraint. We implement DC-LEDBAT in a Linux kernel with MPTCP v0.95 [8] and evaluate its performance in two scenarios from [7]: the Non-Shared Bottleneck (NSB) scenario and the Shared Bottleneck (SB) scenario. The main contributions of this paper are summarized as follows:



(a) Non-shared bottleneck (b) Shared bottleneck

Fig. 1. Evaluation scenarios.

- We analyze the benefits of MPTCP for background traffic transport and illustrate latecomer's unfairness and multipath unfairness through experiments to show that LEDBAT is not adequate for MPTCP.
- We propose a novel multipath LPCC mechanism named DC-LEDBAT. It consists of a mechanism for precisely measuring the base delay, a shared bottleneck detection mechanism that leverages the correlation of each subflow's queuing delay, and a shared bottleneck-based coupled CWND management mechanism for low priority flows' fairness.
- By implementing DC-LEDBAT in the Linux kernel, we evaluate its performance in two of the most common multipath scenarios. The results show that DC-LEDBAT can make full use of the excess bandwidth while remaining fair to the same priority flows.

II. Background and Related Work

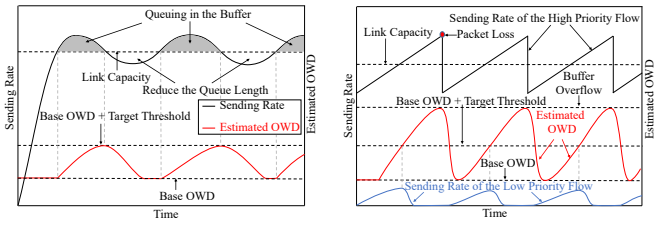
We first give a brief overview of LEDBAT [2]. LEDBAT is a delay-based congestion control mechanism, and it uses one-way delays instead of round-trip times (RTT) to adjust its congestion window. With the goal of providing LBE services, it uses a predefined queuing delay threshold to avoid congestion and packet losses. Meanwhile, in order to utilize the excess bandwidth more effectively, it uses a linear controller to adjusts its CWND $W(t)$ and keeps the queuing delay $q(t)$ as close to the threshold δ as possible. Unfortunately, LEDBAT lacks a mechanism to equalize resource utilization amongst LEDBAT flows [4], [5]. Carofiglio *et al.* [9] illustrated the latecomer's unfairness of LEDBAT due to its additive decrease approach and proposed a multiplicative decrease approach named fLEDBAT to alleviate this problem. However, we found that even fLEDBAT cannot solve the problem thoroughly. Latecomer's unfairness still exists in multipath scenarios, and we will illustrate it in Section III.

With the prevalence of multihomed devices and the wide application of multipath protocols, some studies seek to combine the LPCC mechanism with multipath protocols. Adhari *et al.* [10] applied LEDBAT to SCTP and Montes *et al.* [11] used a traditional congestion control mechanism for the primary subflow and LPCC mechanisms for other subflows. Although these approaches can improve throughput by using multiple network paths, they may harm other low priority flows due to the lack of a mechanism specifically designed for fairness of the same priority flows.

One of the major challenges in the application of a multipath protocol is to be friendly to a single-path one. If we only focus on the advantages of multipath without considering the fairness constraints, then it would be contrary to the design principles of multipath protocols. To design a multipath LPCC mechanism, we get inspiration from bottleneck detection mechanisms designed for best-effort services. These mechanisms seek to ensure bottleneck fairness. The bottleneck fairness is that a multipath flow gets a fair share with a single-path flow at each individual bottleneck, which means a set of subflows only get one share when they are sharing a bottleneck. To satisfy bottleneck fairness, Hassayoun *et al.* [7] used RTTs and packet losses to detect shared bottlenecks, Ferlin *et al.* [12] leveraged the statistical properties of one-way delay (OWD) and interaction between the server and the client for detection, Wei *et al.* [13] introduced explicit congestion notification (ECN) signal as a basis and Hayes *et al.* [14] even used the method of wavelet transform to improve accuracy. Unfortunately, all these approaches are tightly coupled with best-effort services and they fall short in leveraging the characteristics of low priority subflows. For a multipath LPCC mechanism, we need a simpler and more targeted approach to detect shared bottlenecks.

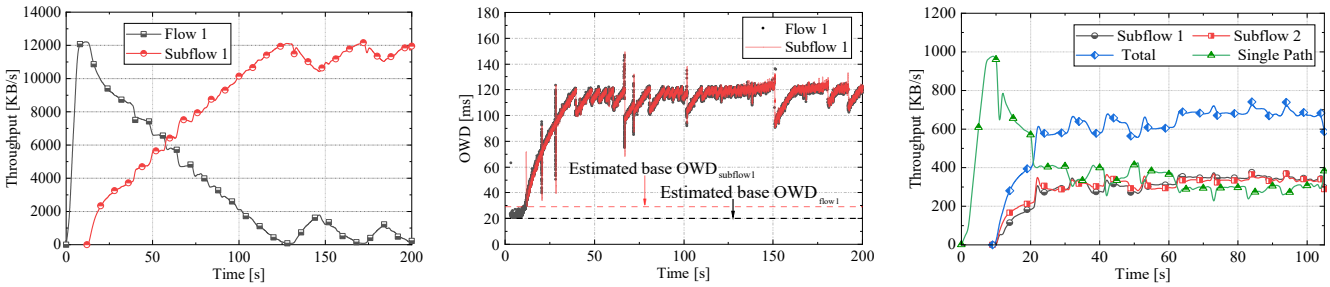
III. Motivation

In this section, we highlight the benefits and constraints faced by multipath background traffic transport and illustrate the problems of existing approaches through some experiments. We follow the standard of LEDBAT and set the threshold δ to 100 ms.



(a) Low priority flow: ideal case (b) Coexistence of different priority flows: ideal case

Fig. 2. Ideal low priority congestion control.



(a) Latecomer's unfairness (b) Estimated OWD (c) Multipath unfairness

Fig. 3. Unfairness among the same priority flows in the NSB scenario.

A. Combining the advantages of LPCC and multiple paths

Fig. 2 schematically depicts the ideal LPCC mechanism. Fig. 2(a) shows the sending rate and the estimated OWD for a low priority flow. The sender tries to make the sending rate close to link capacity while monitoring the OWD not to exceed the predefined threshold. And Fig. 2(b) shows the low priority flow competing with a high priority flow using a traditional, loss-based congestion control mechanism. An ideal LPCC mechanism should do no harm to high priority flows and improve the bandwidth utilization.

MPTCP can offer bandwidth aggregation, more efficient resource utilization, and resilience to link failures. Meanwhile, MPTCP is a transport layer protocol, which is transparent to upper-layer applications. All these characteristics are adequate for background traffic transport. A multipath LPCC mechanism can meet the diverse needs of users and further improve the overall user experience.

B. The Quest for Fairness of the Low Priority Flows

An important criterion for designing a congestion control mechanism is to ensure fairness of the same priority. Due to the characteristics of LPCC mechanisms, the aggressive slow start phase is replaced by an additive increase approach. Besides, the additive decrease approach cannot reduce the queue length rapidly at bottleneck links. LEDBAT suffers from latecomer's unfairness. Carofiglio *et al.* [9] proposed a multiplicative decrease approach named fLEDBAT. To verify whether fLEDBAT is adequate for MPTCP, firstly, we conduct experiments in the NSB scenario. The links are set to 100 Mbps and RTT (from each client to the corresponding server) is 40 ms, which means the base OWD is 20 ms. The single-path fLEDBAT flow 1 starts at 0 s and the multipath flow starts at 10 s. The experiment is repeated many times and several of them show obvious latecomer's unfairness. The latecomer's throughput after 50s shows much higher than the original flow in Fig. 3(a) and we can see from Fig. 3(b) that sometimes even multiplicative decrease cannot correctly measure the base delay. Secondly, we conduct experiments in the SB scenario with 8 Mbps links (Fig. 1(b)) to reveal the unique issue of multipath with the same conditions as the former. To avoid the influence of latecomer's unfairness, we set the base OWD measured in the algorithm as a fixed value (same as the base OWD). As shown in Fig. 3(c), the multipath flow takes up twice the bandwidth of the single-path flow.

In order to make better use of MPTCP for background traffic transport and meet the corresponding constraints, it is urgent to design an adequate multipath LPCC mechanism for MPTCP.

IV. DC-LEDBAT: System Design and Implementation

A. Overview

DC-LEDBAT combines a set of mechanisms to solve the issues mentioned in Section III. It is designed based on LEDBAT, which consists of a base OWD measurement mechanism, a shared bottleneck detection mechanism, and a shared bottleneck-based coupled congestion control mechanism to make full use of the multipath excess bandwidth and ensure fairness among the same priority flows.

In order to completely avoid latecomer's unfairness, we introduce a Probe-OWD phase, which is inspired by BBR [15] and successful application of multipath BBR [16]. In this phase, we make a trade-off between throughput and the precise measurement of the base delay. We set the CWND to the initial value in the Probe-OWD phase, which quickly reduces the queue length and makes the corresponding subflow measure the correct base OWD. After introducing the Probe-OWD phase, there are two cases that cause a significant reduction in the queuing delay. The first one is multiplicative decrease and the second one is the Probe-OWD phase. We treat both of them as queue reduction signals. By analyzing the correlation between each subflow's queue reduction signals, we can simply and efficiently judge whether a set of subflows are sharing bottlenecks. To achieve bottleneck fairness mentioned in Section II, a newly designed coupled CWND management mechanism based on the total CWND and the ratio of each subflow's CWND to its RTT is used for the subflows within the same set. Detailed descriptions are presented in the following subsections.

B. Probe-OWD Phase

Whether it is an approach of a larger additive decrease gain or an approach of multiplicative decrease, the basic idea of them is to heuristically decrease the CWND to a more appropriate value when the queuing delay exceeds the predefined threshold δ. We follow this basic idea and make

some improvements. In absence of packet losses (if congestion losses occur, the main task of low priority flows is to quickly yield to standard flows rather than discuss fairness among low priority flows), a latecomer may mistakenly take the sum of the correct base OWD and δ as the estimated base OWD in the worst case. The queuing delay may approach 2δ as a result.

In order to precisely measure the base delay, what we should do is to make the queue depleted as soon as possible. Thus, we introduce a Probe-OWD phase into DC-LEDBAT. We use $TS_{baseOWD}$, reT, now, and TS_{PO} to denote the timestamp of the base OWD last time, the time threshold to re-measure the base OWD, the current timestamp, and the timestamp of the start time of the Probe-OWD phase. For a certain subflow, when $now - TS_{baseOWD} > reT$, it should enter the Probe-OWD phase to reduce the queue length and set $TS_{PO} = now$. The CWND is set to the initial value in the Probe-OWD phase and this phase lasts for 2δ to make sure that the queue is depleted.

The worst latecomer entering the Probe-OWD phase makes several low priority flows see the correct base OWD, and they update $TS_{baseOWD}$. This distributed coordination allows the entire system to get the correct base OWD. Besides, it can also make DC-LEDBAT robust to delay variations due to re-routing.

C. Shared Bottleneck Detection for DC-LEDBAT

After introducing the Probe-OWD phase in DC-LEDBAT, we focus on the issue shown by Fig. 3(c). To achieve bottleneck fairness mentioned in Section II, we leverage the queuing delay to detect whether subflows share bottlenecks. Since the additive decrease approach cannot reduce the queuing delay in time, we adopt the multiplicative decrease approach in DC-LEDBAT. And we can estimate the queuing delay $q(t)$ as follows:

$$q(t) = OWD_{now} - baseOWD, \tag{1}$$

where OWD_{now} denotes the OWD estimated by the most recently arrived packet. When there are only low priority flows, $q(t)$ will show regular changes. Typically, when $q(t) < \delta$, the additive increase of the CWND causes $q(t)$ to approach δ and when $q(t) > \delta$, the multiplicative decrease of the CWND makes $q(t)$ fall rapidly below δ (Fig. 3(b)). In addition, the Probe-OWD phase makes $q(t)$ close to 0. We treat multiplicative decrease (MD) and the Probe-OWD phase as queue reduction signals. When DC-LEDBAT subflows share bottlenecks with standard TCP flows, there will be packet losses and $q(t)$ will far exceed δ. So we treat the queuing delay continuously greater than δ and packet losses as congestion signals. DC-LEDBAT only does the shared bottleneck detection when there is no congestion signal. When a certain subflow i generates a queue reduction signal, DC-LEDBAT monitors the status of all other subflows. For any other subflow in this multipath connection, we assume that it shares the same bottleneck with the subflow i if either of the following situations occurs:

We use TS_{qrMD}, TS_{qrPO}, and $q(t+1)$ to denote the timestamp when queue reduction signals (MD and Probe-OWD) occur and the estimated queuing delay within next RTT.

- **MD queue reduction signal occurs at the same time**. Since the RTT of each subflow may be different, here "same" is a broader concept. For example, subflow i and subflow j have their own RTT. At $TS_{qrMD,i}$, subflow i generates the MD queue reduction signal. If $TS_{qrMD,j} - TS_{qrMD,i} < RTT_j$, we think these two subflows are sharing bottlenecks.
- **The queue reduction signals of a certain subflow lead to a reduction in the queuing delay of any other subflow that does not have these signals**. Due to the existence of the additive increase, the queuing delay of a subflow that traverses independent bottlenecks does not reduce in absence of congestion and queue reduction signals. We still illustrate this case with two subflows (i and j). At $TS_{qrMD,i}$ or $TS_{qrPO,i}$, subflow i generates a queue reduction signal. If subflow j does not generate a queue reduction signal in $TS_{qrMD,i} + RTT_j$ or $TS_{qrPO,i} + RTT_j$ and $q(t+1)_j < q(t)_j$, subflow i and subflow j are sharing bottlenecks.

In order to reduce the misjudgment, we need to double-check the result. If some subflows are judged to be sharing bottlenecks for the first time, DC-LEDBAT will repeat the above steps. If both judgments are sharing bottlenecks, DC-LEDBAT will couple these subflows. Otherwise, DC-LEDBAT will continue to monitor the status of these subflows until the last two judgments are consistent. Based on this result, DC-LEDBAT decides whether to couple or decouple the subflows. Besides, in response to changes in bottlenecks, DC-LEDBAT will re-execute the shared bottleneck detection after any subflow enters the Probe-OWD phase.

D. Congestion Window Management

DC-LEDBAT adjusts the CWND in the same manner as fLEDBAT before the shared bottleneck detection mechanism works:

$$W(t+1) = \begin{cases} \frac{1}{2}W(t) & \text{if packet loss,} \\ W(t) + \alpha\frac{1}{W(t)} & \text{if } \Delta(t) \geq 0, \\ W(t) + \alpha\frac{1}{W(t)} + \zeta\frac{\Delta(t)}{\delta} & \text{if } \Delta(t) < 0, \end{cases} \tag{2}$$

where $\Delta(t) = \delta - q(t)$ denotes the estimated distance from the predefined threshold and ζ denotes the multiplicative decrease gain [9].

After the shared bottleneck detection mechanism groups the subflows, we need to carefully adjust the CWND. The increase of a subflow's CWND depends on three parameters: (i) the minimum RTT (RTT_{min}) of subflows in the same set, (ii) the total CWND of all subflows in this set, (iii) the sum of the ratio of each subflow's CWND to its RTT within the set. For a certain set S_n, a subflow $i \in S_n$ adjusts its CWND $W_i(t)$ for each ACK as follows (we only show the $\Delta(t) \geq 0$ case, the rest are the same as eq. (2):

$$W_i(t+1) = W_i(t) + \frac{\alpha_{S_n}}{W_{S_n}}, \tag{3}$$

where:

$$W_{S_n} = \sum_{i \in S_n} W_i(t), \tag{4}$$

$$\alpha_{S_n} = \frac{1}{RTT_{min} \sum\limits_{i \in S_n} \frac{W_i(t)}{RTT_i}}. \quad (5)$$

We give this window increase approach mainly by combining the existence of a global stable state of fLEDBAT mechanism and the constraints of MPTCP coupled congestion control mechanisms. It has been proved in [9] that the system will reach a stable state if all flows are in the following states (we assume that there are N flows at the bottleneck):

$$\frac{W_i^*(t)}{RTT_i^*} = \frac{C}{N}, \qquad and \qquad Q^* = C\delta + \frac{N\alpha\delta}{\zeta T}, \quad (6)$$

where $W_i^*(t)$ and RTT_i^* denotes the stationary value of the CWND and RTT of subflow i. Q, C, and T denote the queue length, the bottleneck capacity, and the propagation delay. Under the premise of the existence of a stable state, what we need to do further is to make the performance of a certain set of multipath subflows the same as a single-path one. To achieve this goal, we make the total CWND increase rate of each set of subflows equal to that of the best single-path fLEDBAT flow. Because a certain set of subflows have the same queuing delay when they are sharing a bottleneck, these subflows multiplicative decrease their CWND almost simultaneously. Coupled with the same increase rate as the best single-path fLEDBAT, the performance of these subflows in the system is the same as the best single-path one. Thus, we get the following equation:

$$\sum_{i \in S_n} \frac{\alpha_{S_n} W_i(t)}{W_{S_n} RTT_i} = \max_{i \in S_n} \frac{\alpha^{TCP}}{RTT_i}, \quad (7)$$

where $\frac{\alpha_{S_n} W_i(t)}{W_{S_n} RTT_i}$ denotes the CWND increase rate of subflow i, $\frac{\alpha_{S_n}}{W_{S_n}}$ denotes the increase value of the CWND for each ACK, RTT_i on the denominator is to normalize the time, and $W_i(t)$ on the numerator denotes how many ACKs arrive per RTT_i on subflow i. Generally, α^{TCP} is equal to 1 and we get eq. (5) by rearranging eq. (7).

V. Performance Evaluation

We evaluate the performance of DC-LEDBAT in our testbed with mini PCs. As shown in Fig. 1(a) and Fig. 1(b), the testbed has two most common topologies when using MPTCP. We implement DC-LEDBAT in the Linux kernel with MPTCP v0.95 [8] and use *tc* to set different network conditions.

A. Performance analysis of DC-LEDBAT

The primary evaluation index of an adequate multipath LPCC mechanism is yielding to traditional congestion control mechanisms when sharing bottlenecks. To verify this, we conduct an experiment in which the DC-LEDBAT flow coexists with high priority flows in the NSB scenario (Fig. 1(a)). The MPTCP client and server are connected via two 8 Mbps links. The multipath LEDBAT flow starts at 1 s, after which the two TCP clients and servers respectively transfer data through Cubic flow (standard TCP flow) for 25 s. And Fig. 4 shows that DC-LEDBAT can improve throughput by using multiple network paths while ensuring its low priority characteristics.

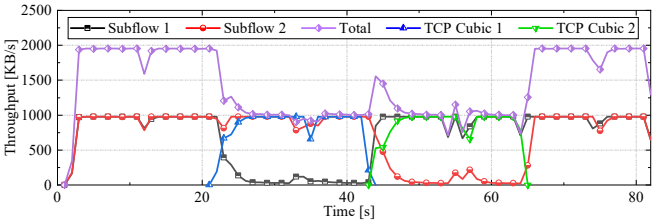


Fig. 4. Coexistence with high priority flows.

We further conduct some experiments to show the scalability of DC-LEDBAT in the NSB scenario, and the links are set to 10 Mbps. We use multiple processes on the MPTCP client to simultaneously download files from the MPTCP server. Each connection has two subflows, and we test the coexistence of up to 5 multipath connections. As shown in Fig. 5, DC-LEDBAT can keep good fairness of the same protocol. However, as the number of connections increases, the overall bandwidth utilization decreases slightly. This is mainly due to the introduction of the Probe-OWD phase, and it will be studied in the next subsection.

B. Solve the problem of latecomer's unfairness

In this subsection, We conduct some experiments to prove that the Probe-OWD phase is superior in solving the problem of latecomer's unfairness. In contrast to the previous experiment shown in Fig. 3(a) and Fig. 3(b), the network conditions are set to the same parameters. Besides, we set reT to 10s (the same as BBR [15]). The single-path fLEDBAT flow 1 starts at 0s and the DC-LEDBAT flow starts at 15 s. Subflow 1 shares a bottleneck with flow 1 and the result is shown in Fig. 6. Compared with Fig. 3(a), it is obvious that DC-LEDBAT has no latecomer's unfairness and our design goal is achieved by introducing the Probe-OWD phase.

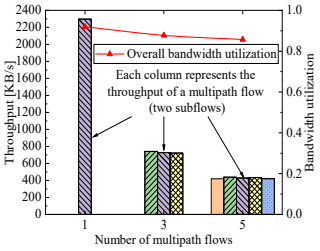


Fig. 5. Fairness of multiple DC-LEDBAT flows.

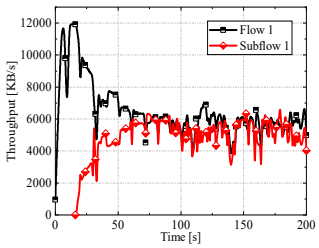


Fig. 6. Throughput comparison between DC-LEDBAT subflow 1 and fLEDBAT flow.

The Probe-OWD phase trades bandwidth for precise measurement of base OWD. This experiment is to show that the Probe-OWD phase does not have a big impact on throughput. We continue to use the same network conditions, and we separately test the performance of fLEDBAT and DC-LEDBAT in the NSB scenario. We conduct the experiment 10 times and average the results. In each experiment, we use the MPTCP client and server to transfer data for 200 s. As shown in Table I, the ratio of the total throughput of the above two mechanisms

is 97.8%, and it can be concluded that the Probe-OWD phase has just a minor impact on throughput.

TABLE I
THROUGHPUT COMPARISON BETWEEN FLEDBAT AND DC-LEDBAT.

Throughput / Mechanism	Subflow 1	Subflow 2	Total
fLEDBAT	94.00 Mbps	93.94 Mbps	187.94 Mbps
DC-LEDBAT	91.92 Mbps	91.86 Mbps	183.78 Mbps

C. Bottleneck fairness achieved by DC-LEDBAT

Sharing bottlenecks between subflows is a common situation when using MPTCP. In order to illustrate DC-LEDBAT can achieve bottleneck fairness, we conduct experiments in both SB (Fig. 1(b)) and NSB (Fig. 1(a)) scenarios. To compare with Fig. 3(c), we set the network conditions to be the same. The links are set to 8 Mbps and RTT is 40 ms.

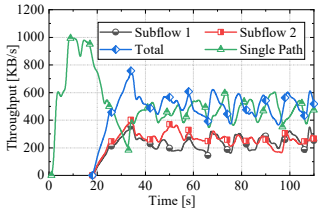


Fig. 7. Throughput comparison in the SB scenario.

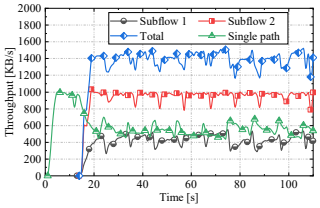


Fig. 8. Throughput comparison in the NSB scenario.

As shown in Fig. 7, from about 40 s, DC-LEDBAT's two subflows and single-path LEDBAT flow achieve a dynamic equilibrium. Further, we calculate the average throughput from 40 s to 100 s, the average throughput of subflow 1, subflow 2, total of DC-LEDBAT, and single-path fLEDBAT is 236.14KB/s, 263.11KB/s, 499.25KB/s, and 475.73KB/s. Due to the shared bottleneck-based coupled congestion control mechanism, DC-LEDBAT judges that subflow 1 and subflow 2 share a same bottleneck and controls their CWND increase rate. Consequently, DC-LEDBAT shows the multipath-friendliness that fLEDBAT does not possess.

Finally, to prove that DC-LEDBAT does not have misjudgment of shared bottleneck in the NSB scenario. We use the same network conditions in the SB scenario. The single-path flow 1 starts at 0 s and the DC-LEDBAT flow starts after a period of time. Fig. 8 shows that subflow 1 and flow 1 have an equal share of the bandwidth of the upper link in Fig. 1(a) while subflow 2 can make full use of the bandwidth of the lower link. This is in line with our design goals.

In summary, DC-LEDBAT not only maintains the low priority characteristics of LEDBAT, but also makes full use of the advantages brought by multiple network paths. Meanwhile, DC-LEDBAT satisfies bottleneck fairness of the multipath congestion control mechanism.

VI. CONCLUSION

In this paper, we gave the design criterions of multipath LPCC mechanisms and proposed a novel mechanism named DC-LEDBAT. DC-LEDBAT takes advantage of MPTCP's aggregation bandwidth and it also ensures fairness of the same priority at the connection level. DC-LEDBAT avoids latecomer's unfairness by introducing the Probe-OWD phase, which enables the subflows to precisely measure the base OWD. Then, it uses a shared bottleneck detection mechanism specifically for low priority flows by analyzing the queuing delay. And it ensures bottleneck fairness by using a dynamic coupled CWND management mechanism. Our experimental results showed that DC-LEDBAT is an adequate multipath LPCC mechanism for background traffic transport.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371 and Youth Innovation Promotion Association Chinese Academy of Sciences (CAS) under Grant No. Y202093.

REFERENCES

[1] D. Ros and M. Welzl, "Less-than-Best-Effort Service: A Survey of End-to-End Approaches," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 898–908, 2013.

[2] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)," *IETF, RFC6817*, 2012.

[3] A. Ford, C. Raiciu, M.Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," *IETF, RFC 6824*, 2013.

[4] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "LEDBAT: The New BitTorrent Congestion Control Protocol," in *Proceedings of the 19th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2010, pp. 1–6.

[5] G. Carofiglio, L. Muscariello, D. Rossi, and S. Valenti, "The Quest for LEDBAT Fairness," in *Proceedings of the 2010 IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE, 2010, pp. 1–6.

[6] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural Guidelines for Multipath TCP Development," *IETF, RFC6182*, 2011.

[7] S. Hassayoun, J. Iyengar, and D. Ros, "Dynamic Window Coupling for Multipath Congestion Control," in *Proceedings of the 19th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2011, pp. 341–352.

[8] C. Paasch, S. Barre, and et al., "Multipath TCP in the Linux Kernel." [Online]. Available: https://www.multipath-tcp.org

[9] G. Carofiglio, L. Muscariello, and et al., "Rethinking the Low Extra Delay Background Transport (LEDBAT) Protocol," *Computer Networks*, vol. 57, no. 8, pp. 1838–1852, 2013.

[10] H. Adhari, S. Werner, T. Dreibholz, and E. P. Rathgeb, "LEDBAT-MP – On the Application of "Lower-than-Best-Effort" for Concurrent Multipath Transfer," in *Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops (AINA)*. IEEE, 2014, pp. 765–771.

[11] I. Montes, R. Parmis, R. Ocampo, and C. Festin, *Multipath Bandwidth Scavenging in the Internet of Things*. Springer International Publishing, 2015, ch. Internet of Things. User-Centric IoT, pp. 297–304.

[12] S.Ferlin, Ã. Alay, T. Dreibholz, D. A. Hayes, and M. Welzl, "Revisiting Congestion Control for Multipath TCP with Shared Bottleneck Detection," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2016, pp. 1–9.

[13] W. Wei, K. Xue, J. Han, D. S. Wei, and P. Hong, "Shared Bottleneck-Based Congestion Control and Packet Scheduling for Multipath TCP," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 653–666, 2020.

[14] D. A. Hayes, M. Welzl, S. Ferlin, D. Ros, and S. Islam, "Shared Bottleneck-Based Congestion Control and Packet Scheduling for Multipath TCP," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2229–2242, 2020.

[15] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[16] J. Han, K. Xue, Y. Xing, J. Li, W. Wei, D. S. Wei, and G. Xue, "Leveraging coupled bbr and adaptive packet scheduling to boost mptcp," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2021.