This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

TCP FlexiS: A New Approach To Incipient Congestion Detection and Control

Qian Li

***Abstract*—Best effort congestion controls strive to achieve an equitable distribution of network resources among competing flows. However, fair resource allocation becomes undesirable when a bandwidth/delay sensitive application shares a bottleneck with a greedy background application. Less than Best Effort (LBE) Congestion Control Algorithms (CCA) are specially designed for background applications, which do not have strict bandwidth/delay requirements. LBE CCAs give foreground applications higher priority in resource allocation by only utilizing spare bandwidth. This can greatly improve network utility at times of congestion. We propose FlexiS – a Flexible Sender side LBE CCA. Unlike most conventional LBE CCAs, which use queue size based congestion detectors and linear rate controllers, FlexiS employs a queue trend based congestion detector and a cubic increase multiplicative decrease rate controller. We have compared FlexiS with LEDBAT and LEDBAT++. Extensive emulation and preliminary Internet tests showed that: (1) FlexiS has comparatively low impact on concurrent best effort TCP flows; (2) it scales to a wide range of available bandwidths; (3) FlexiS flows in aggregation can efficiently utilize available bandwidth; (4) contending FlexiS flows can, in most cases, equally share available bandwidth; (5) it adapts to route changes quickly; and (6) it maintains low priority even when AQM algorithms or shallow buffers are deployed.**

***Index Terms*—Lower than Best Effort, LBE, Low Priority, Congestion Control Algorithms.**

I. Introduction

The Internet is a packet switched network. Packets of different applications share a common set of network resources including transmission medium, switching and routing devices and device buffers. Packets that cannot be transmitted immediately are placed in a buffer and are usually served in First Come and First Served (FCFS) order. Packets are dropped when the buffer is full. The most widely adopted mechanism for network resource allocation so far has been based on end points through Congestion Control Algorithms (CCA). A CCA determines at what rate a transmitter should be sending, which further determines the bandwidth share of a flow (a sequence of packets that have the same source IP, source port, destination IP, destination port and protocol). Fairly sharing network resources among competing flows has been one of the design goals of many CCAs. These CCAs are usually referred to as Best Effort (BE) CCAs. The dominant BE CCAs use loss as a signal of congestion.

However, allocating network resources fairly among competing flows is not always desirable. Consider the following scenario. While Alice is enjoying Video on Demand (VoD) streaming (APP A), a software update (APP B) is initiated by a server to transfer 1 GB of software patches to Bob's computer in the background. APPs A and B share the same bottleneck and both use BE CCAs. The fair competition of APP B makes APP A lose a portion of bandwidth, which causes stalls during video play back. Consequently, the Quality of Experience (QoE) of Alice is degraded. In comparison, if APP B employs a Less than Best Effort (LBE) CCA that can detect APP A and never acquires bandwidth more than what is available, the minimum bandwidth requirements of APP A can be guaranteed and the QoE of Alice will not be affected. In the meantime, Bob will not notice the increased download time of software patches since the software update runs in the background. Clearly, priority-based resource allocation is more preferable over fair allocation in the above scenario.

In reality, scenarios similar to the one illustrated above are not uncommon. When QoS sensitive foreground applications (such as VoIP, online gaming, video streaming and web browsing) compete fairly with QoS tolerant background applications (e.g., software update, client-to-cloud backup, inter-data-center synchronization and peer-to-peer file sharing), the QoE of foreground application users will be degraded. LBE CCAs are specially designed for background applications. They opportunistically scavenge spare bandwidth (henceforth, used interchangeably with available bandwidth) left by other applications, thus maximize QoE of all Internet users. Some LBE CCAs have been proposed in the literature [1] [2] [3] and Microsoft has incorporated two – LEDBAT++ [4] and rLEDBAT [5] – in its Windows Servers to limit bandwidth consumption of software updates and error reporting [6].

In this article, we propose a new LBE CCA named FlexiS – a Flexible Sender side LBE CCA. The objectives of FlexiS are: (1) low intrusion on concurrent foreground applications; (2) high bandwidth utilization; and (3) equally sharing Available Bandwidth (AB) between contending FlexiS flows. The objectives are listed in order of priority. FlexiS has been implemented as a Linux kernel module. The source code is available as open source software at [7].

Most existing LBE CCAs [1], [2], [3] and delay based CCAs [8], [9], [10], [11] need to estimate base delay in order to detect incipient congestion. Base delay is the time spent by a packet to traverse an unloaded route. In practice, it is very difficult to estimate. On the one hand, it can change with the change of route. If a route consists of wireless links, its base delay can change even when route is not changed [12]. On the other hand, a connection might not be able to observe the true base delay during its lifetime if the bottleneck is persistently overloaded. A wrong estimation of base delay can make an LBE CCA fail to meet its design objectives. For instance, LEDBAT has been shown to suffer from a latecomer advantage problem [3] [13] and to induce increasing Queuing Delay (QD)

Q. Li was with the Department of Informatics, University of Oslo, Oslo, Norway. E-mail: li_qian_pro@hotmail.com

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

until buffer overflow [14] due to wrong estimation of base delay.

We had been searching for an alternative incipient congestion detection technique. We discovered that an increasing trend in Round Trip Time (RTT) serves as a good indication of early congestion because congestion is usually accompanied by increased QD. Extensive evaluation showed that FlexiS does not suffer from problems related to inaccurate base delay estimation. In particular, it adapts to route changes quickly.

Recent LBE CCAs such as LEDBAT [3] and LEDBAT++ [4] use fixed delay TARGETs as congestion thresholds. This technique has two major drawbacks. On the one hand, a fixed amount of extra QD is added to the estimated base delay for most of the time. On the other hand, if the bottleneck has a small buffer or Active Queue Management (AQM) algorithms are deployed, packet loss can occur before the LBE CCAs reach their delay targets. This will make them as aggressive as some BE CCAs in certain scenarios [15]. With the ongoing effort on tackling buffer bloat, recent years witnessed rapid AQM deployment in home gateways [16]. It has become a must-have feature of LBE CCAs to retain low priority in the presence of AQMs.

FlexiS does not maintain a fixed QD. It reduces rate when RTT samples have an obvious increasing trend. Experiments showed that the extra QD induced by FlexiS is much smaller than that inflicted by LEDBAT and LEDBAT++ in most cases and that FlexiS can preserve low priority even with the presence of AQMs or shallow buffers. Further, backing off earlier can reduce the number of packets dropped by AQMs or buffer overflow, therefore improve bandwidth utilization in such situations.

Another problem with many conventional LBE CCAs [2] [1] [3] [4] is the use of linear controls for rate adaptation. The merit of linear control is simplicity and the potential of converging to fairness – the equal distribution of bandwidth between contending flows. It is proven by Chiu and Jain in [17] that Additive Increase Multiplicative Decrease (AIMD) is the most feasible and efficient linear control to realize fairness. Although theoretically sound, AIMD has various issues in practice. The typical application of AIMD is to increase congestion window (CWND) by one Maximum Segment Size (MSS) per RTT and reduce it by half. However, this application does not guarantee fairness between connections with different RTTs. Floyd proposed in [18] that all connections despite their RTTs should increase their rates by a constant amount of a packets/second during each second. However, it was later on substantiated that this increase function is difficult to successfully deploy in an operational network [19]. In addition to the fairness problem, AIMD used by standard TCP is also known to have low bandwidth utilization problem in large Bandwidth Delay Product (BDP) networks. It is shown in [20] that an unrealistically small Bit Error Rate (BER) is required for standard TCP to achieve full bandwidth utilization in such networks.

We evaluated a variety of rate increase/decrease functions. Finally, a cubic increase multiplicative decrease rate controller was chosen. To be specific, FlexiS increases sending rate using a cubic function of time and decreases rate by a fixed percentage point. FlexiS does not have a slow start phase, the same rate increase function is applied whenever rate should be increased. Experiments showed that with FlexiS' cubic increase function, the initial long delay and high loss caused by slow start can be avoided. Cubic increase makes FlexiS scalable across a wide range of ABs. And it improves bandwidth utilization in large BDP networks with random loss. Cubic increase multiplicative decrease together contribute to FlexiS' high intra- and inter-RTT fairness.

The rest of the paper is organized as follows. In section II we review related work. Section III elaborates on the design of FlexiS. Extensive evaluation results are presented in section IV. Finally, section V concludes the paper.

II. Related Work

In this section, we review previous work that either inspired FlexiS or that are similar to FlexiS in objectives or in design.

A. Work that Inspired FlexiS

The design of FlexiS' congestion detector is greatly influenced by Pathload [21], which is an AB estimator. A pathload sender transmits a fleet of UDP packet streams to a receiver with a predetermined stream rate and inter-stream interval. Upon the receipt of all probe packets and One Way Delay (OWD) samples of a stream, the receiver analyzes the trend in OWD using two statistics. If the majority of the streams in a fleet cause OWD to have an increasing trend, the rate of the fleet is considered higher than the AB, otherwise lower. AB can be discovered by sending out multiple fleets with different rates. Inspired by pathload, FlexiS uses an increasing trend in delay as the indicator of incipient congestion. Unlike pathload, FlexiS uses in-band TCP packets as probe packets and employs linear regression to determine the trend in RTT. Further, it uses RTT in congestion detection so that a sender does not need the support of a receiver.

B. Work Similar in Design

Similar to FlexiS, Some CCAs also use trend in delay or delay gradients in congestion detection. They differ from FlexiS mainly in how delay gradient is calculated and utilized. ImTCP [22] employs the two statistics technique proposed in [21] to estimate the trend in RTT. Probe Control Protocol (PCP) [23] derives the trend in OWD using least squares. CDG [24] uses the minimum or maximum RTTs measured in consecutive round trips to derive delay gradients. If the average of last n delay gradients is greater than zero, congestion is concluded. Timely [25] uses a positive delay gradient as an indicator of congestion. The delay gradient is a ratio between the average (EWMA) RTT difference between consecutive RTT samples and the minimum RTT. PCC-Vivace [26] and PCC-Proteus [27] use linear regression to generate RTT gradients, which is used as an independent variable of a utility function. In contrast, FlexiS uses a Theil-Sen estimator to determine the slope of the linear regression line of RTT samples. Slope exceeding a threshold is used to indicate congestion.

Some other CCAs employ similar increase functions as FlexiS. CUBIC [28] uses a cubic function of elapsed time

since last congestion event as its window growth function. It uses both the concave and convex profiles of a cubic function. Similarly, FlexiS also employs a cubic function of time elapsed since the start of the increase epoch for rate increase. But it only uses the convex profile of a cubic function. TCP-LoLa [29] employs two cubic functions for rate increase. One is CUBIC's CWND growth function. Another is a cubic function of time elapsed since the start of fair flow balancing. It is used to calculate a dynamic queue occupancy target to achieve fairness. This latter function resembles FlexiS' increase function no matter in the function itself but also in objectives. Unlike CUBIC and TCP-LoLa, in order to speed up initial rate increase, FlexiS adds a first degree term to its increase function.

C. Work Similar in Objectives

The CCAs presented in this subsection all have LBE as their design objectives.

TCP Nice [1] and TCP-LP [2] are two early LBE CCAs. Both of them use (filtered) QD exceeding a certain percent of maximum QD as congestion indicator. CWND is halved when congestion is detected. Linearly increase functions are used to grow CWND. Eclipse [30] is a hybrid of three LBE CCAs. Its target QD calculation is based on TCP Nice and TCP-LP and its rate controller is based on LEDBAT.

ImTCP-bg [31] and TCP Westwood Low Priority [32] are LBE CCAs that are adapted from BE CCAs. The former uses the AB estimated by ImTCP to calculate an upper limit for CWND. The latter adds a backlog based incipient congestion detector to TCP Westwood.

FLOWER [33] adjusts CWND with a fuzzy controller. DA-LBE [34] is a deadline aware LBE framework that is capable of making any non-LBE CCAs to have LBE behavior and the degree of LBEness can be changed according to remaining time to deadline.

D. Work Used for Comparison

Low Extra Delay BAckground Transport (LEDBAT) [3] and LEDBAT++ [4] are two recent LBE CCAs. Both published specifications with IETF and carry real world traffic in the Internet. In this article, FlexiS is compared with them.

LEDBAT uses QD exceeding a TARGET (100 ms by default) as an indication of congestion. If QD is below TARGET, CWND is additively increased, otherwise additively decreased. The amount of increase/decrease is in proportion to off_target = (TARGET - QD) / TARGET. QD is the difference between the current OWD and base_delay. LEDBAT adjusts CWND with the following function. CWND = CWND + GAIN $\times$ off_target $\times$ bytes_newly_acked $\times$ MSS / CWND. GAIN determines the rate at which the CWND responds to changes in QD. Base delay is the minimum OWD observed during past a few minutes.

LEDBAT++ improves LEDBAT using a number of techniques. It replaces OWD with RTT in QD calculation. Slow start quits when QD exceeds 3/4 TARGET to reduce initial intrusion. LEDBAT's linear controller is replaced by an additive increase multiplicative decrease controller, which is borrowed from fLEDBAT [35]. The authors of fLEDBAT propose to use constant increase to replace proportional increase to speed up convergence and improve efficiency. Additive decrease is attributed to the cause of unfairness and is replaced by multiplicative decrease. In LEDBAT++, the pre-configured increase GAIN is replaced by a dynamic GAIN which is in proportion to a connection's RTT. LEDBAT++ periodically reduces CWND to 2 packets in order to discover a true base RTT.

III. Design

A. Overview

FlexiS uses an increasing trend in RTT to indicate congestion. On the receipt of the i_{th} ACK, an RTT sample d_i and a timestamp t_i are obtained. t_i is the sending time (in ms) of the acknowledged data packet. (t_i, d_i) are put into a data structure called RTT sack (denoted as D) as one data point. If enough data points are gathered in D, the slope of the linear regression line of all data points in D is derived and the oldest data point (t_1, d_1) is removed. If the slope is equal to or greater than a threshold θ, RTT is considered to have an increasing trend and FlexiS will reduce its CWND by a fixed percentage point. Otherwise, CWND will be increased according to a CUBIC function of elapsed time. FlexiS halves its CWND when packet loss is detected. It does not distinguish algorithmically between slow start and congestion avoidance. Algorithm 1 shows a pseudo-code of the main logic of FlexiS.

Algorithm 1 The main logic of FlexiS

Variables
L_D: length of D in ms
N_D: size of D in number of data points
S_D: the slope of the regression line of data points in D
σ: the minimum number of data points in D required to make a trend analysis. default: 3
τ: the minimum length of D required to make a trend analysis. default: 60 ms
θ: the minimum slope that makes FlexiS reduce rate. default: 30
On the receipt of the i_{th} ACK
put (t_i, d_i) into D
$L_D \leftarrow t_i - t_1$
if $L_D < \tau$ **then**
  CWND is unchanged
else
  if $N_D < \sigma$ **then**
    increase CWND per Equations 1 & 2 (Section III-C)
  else
    calculate S_D
    if $S_D \geq \theta$ **then**
      decrease CWND per Equation 3 (Section III-D)
    else
      increase CWND per Equations 1 & 2 (Section III-C)
    end if
  end if
  remove the oldest data point (t_1, d_1) from D
end if

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

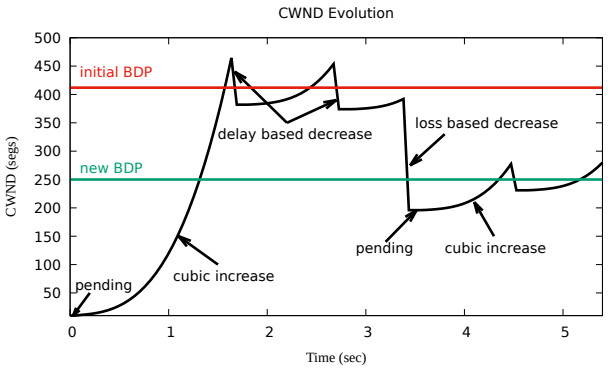


Fig. 1. A Sample Evolution of FlexiS' Congestion Window

From a high level view, a FlexiS sender iteratively goes through three phases: pending, increase and decrease. This can be best illustrated with a CWND evolution schematic as is shown in Fig. 1. In the pending phase, the sender is not qualified to make a trend analysis due to insufficient observation duration ($L_D < \tau$). In this phase, it puts every newly received RTT sample into D and keep CWND unchanged. A newly established connection always starts from the pending phase. As soon as $L_D >= \tau$, the pending phase terminates and the sender goes into either the increase or decrease phase. If there are not enough data points ($N_D < \sigma$) in D, FlexiS simply increases CWND. If sufficient data points are obtained, the trend in RTT is derived. If the trend is non-increase, the sender enters the increase phase, otherwise, it enters the decrease phase (in Fig. 1, FlexiS always enters the increase phase after pending). An increase phase is also termed an increase epoch, which terminates when congestion is detected (at seconds 1.6, 2.7, 3.4 and 4.5 in the schematic). The sender will then go into the decrease phase, in which, CWND is decreased and D is emptied. Due to the need of replenishing D, a decrease phase is always followed by a pending phase.

Each component of FlexiS is designed to support fairness. The design ensures that all contending FlexiS flows (1) make congestion decisions based on the same criteria; (2) increase rates at the same speed; and (3) decrease rates by the same percentage points. In the rest of the article, we will refer to these as the necessary conditions to fairness. Next, we elaborate on each component of FlexiS.

B. Congestion Detection

FlexiS detects congestion based on the following definition. *Congestion is the state of sustained overload of a bottleneck*. According to the definition, the two prerequisites for congestion are "overload" and "sustained". "Overload" can be manifested by an increasing trend in RTT, in that when a link is overloaded, queue will increase over time, which makes RTT have an increasing trend. θ is a threshold for "overload". Any slope below θ will not trigger the congestion response of FlexiS. "Sustained" imposes a minimum duration for "overload". τ is a threshold for "sustained". That is, if the network is overloaded for at least τ ms, FlexiS will take congestion response. τ and θ are RTT-independent constants. This can make flows with different RTTs detect congestion based on the same criteria, which is the first necessary condition to fairness.

The trend in RTT can be quantified using the slope of the regression line of the data points in D. We use a Theil-Sen estimator [36] [37] to estimate the slope. The Theil-Sen slope is the median of slopes of lines connecting all distinctive pairs of points in D. Let $P_i = (t_i, d_i)$ and $P_j = (t_j, d_j), t_i < t_j$ be any two data points in D, S_{ij} be the slope of the line connecting points P_i and P_j. We have $S_{ij} = (d_j - d_i)/(t_j - t_i)$, and $S_D = \text{median}(S_{ij})$. In our Linux implementation, slopes are magnified 1000 times because fractional numbers are not supported by the kernel. When the magnified Theil-Sen slope is greater than θ, the RTT samples are deemed to have an increasing trend.

The space and computation complexity of our Linux kernel implementation of the Theil-Sen estimator are both $O(n^2)$. When sending rate is high, we get quite a number of data points in D. Under such circumstances, the computation time of S_D is not negligible. In order to reduce overhead, we employ a technique, which compresses all RTT samples with the same timestamp into one data point P. The RTT of P is the median of all RTT samples being compressed. The timestamp of P is the common timestamp. After compression, there can be at most τ samples in D and the space and computation overhead caused by the Theil-Sen estimator can be neglected.

C. Rate Increase

FlexiS increases its sending rate with a cubic function of time elapsed since the start of an increase epoch E. Fig. 1 illustrates the operation of the increase function. And Equations 1 and 2 give the definition of FlexiS' increase functions. The same functions are used for both initial ramp up and congestion avoidance. In the following equations, RTT is measured in unit of seconds, timestamps in milliseconds, rate in packets per second and CWND in packets. $d_{min}(t1, t2)$ denotes the minimum RTT observed between time t1 and t2.

$$r = (\frac{t_{cur} - t_0}{\alpha})^3 + \frac{t_{cur} - t_0}{\beta} + \frac{w_0}{d_0} \quad (1)$$

where r is sending rate, α and β are increase factors. t_0 is the start time of E and t_{cur} is the current time. w_0 is the CWND value at t_0. $d_0 = d_{min}(t_{pend}, t_0)$ is the RTT estimate made at time t_0. t_{pend} is the start time of the preceding pending phase of E. Because RTT usually do not increase significantly in the pending phase (otherwise rate would have been reduced), $d_{min}(t_{pend}, t_0)$ is pretty close to the actual RTT at time t_0. t_0, w_0 and r_0 are recalculated at the beginning of every increase epoch.

$$w = r \times d_{cur} \quad (2)$$

where $d_{cur} = d_{min}(t_{pend}, t_{cur})$ is the RTT estimate made at t_{cur}. The actual RTT at t_{cur} can be higher than d_{cur} if the network is congested at the time. This will make the actual sending rate lower than the theoretical value at times of congestion. However, this side effect is desirable because it can make FlexiS less intrusive when congestion is forming

but the sender is still not aware of it. This is especially helpful when a connection's RTT is high.

FlexiS' rate increase function has a first degree term. It is used to speed up the initial rate increase. A single cubic term makes rate increase too slow when elapsed time is small, which has two undesired effects. On the one hand, overly slow rate increments can adversely affect bandwidth utilization. On another hand, in some rare cases, the slow rate increase makes RTT increase so slowly that it cannot be detected by FlexiS. Adding a first degree term to the function can solve both problems.

α and β determine the curvature of the increase function. The smaller the values the more aggressive FlexiS is. Very aggressive increase will inflict high intrusion and overly conservative increase can harm utilization. The default values for α and β are 100 and 10 respectively. They are proven experimentally to have better trade-off between intrusion and utilization.

We will show next that the increase functions meet all design objectives of FlexiS. First, it is low intrusive. During initial ramp up, it makes rate increase in proportion to CWND, which ensures that the QD induced by FlexiS is bounded to a proportion of an RTT. After a congestion event, elapsed time is reset to zero and FlexiS starts from small increments again. This behavior gives the bottleneck sufficient time to drain its queue. It can also delay the occurrence of the next congestion event, therefore reduce intrusion to foreground traffic. Second, the cubic function allows FlexiS to speed up over time, which makes it possible to converge to very large ABs and significantly reduces convergence time compared to linear increase. Finally, using elapsed time as input makes contending flows with different RTTs increase at the same speed provided that they enter their respective increase epochs at the same time. This is the second necessary condition to fairness. When flows enter their increase epochs at different times, they can increase at different speeds. However, this asynchronous state will end after the first congestion event.

D. Rate Decrease

Upon congestion, FlexiS reduces CWND per Equation 3. It is shown experimentally that the simple decrease function has better worst case scenario performance than more sophisticated decrease functions.

$$w' = max(w \times \gamma, 2) \tag{3}$$

where w' and w are CWND values after and before reduction, $0 < \gamma < 1$ is the decrease factor. The max operator ensures that CWND is never reduced below 2 MSS. This is to mitigate the adverse effect of delayed ACKs.

The default value for γ is 0.85. γ should be set in conjunction with α and β. When α and β are decreased, so should γ so as to counter-act the increased aggressiveness and vice versa. In order to meet the third necessary condition to fairness, all flows should set their decrease factors to the same value.

FlexiS only reduces CWND once per round trip time. CWND is halved (but not below 2 MSS) on packet loss detected by the fast retransmit and fast recovery algorithm and is reduced to 1 MSS when retransmission timer goes off. After any type of CWND reduction, D is emptied.

If fast recovery is entered when FlexiS is in rate reduction, CWND will be reduced twice. Upon the exit of fast recovery, FlexiS sets CWND back to the value after the first reduction. This technique can improve link utilization in the presence of AQM, shallow buffer or random loss.

E. Pacing

FlexiS uses pacing, which is a technique used to evenly space packets at specified intervals at time of sending. Pacing has two functions in FlexiS. On the one hand, it can greatly reduce delay variation and packet losses inflicted by FlexiS flows, therefore reduce intrusion to foreground flows. On the other hand, pacing can improve bandwidth utilization of FlexiS. It minimizes the probability that a FlexiS flow backs off before its rate reaches AB due to self-induced queues.

In Linux, per flow pacing can be realized by TCP or by the fair queue packet scheduler. In either way, we need to determine a desired pacing rate. However, in recent Linux kernels, a congestion control module that only implements congestion avoidance cannot ultimately update pacing rate. As a temporary work-around, we update pacing ratio R_p, which is used to calculate pacing rate by the kernel. R_p is the ratio between the current rate r and the rate in one RTT r'. It is updated per equation 4 right after each CWND update. If rate has just been increased, R_p is updated with the first sub-equation. Otherwise, the rate in one RTT would be equal to the current rate, so R_p should be 100%.

$$R_p = \begin{cases} \frac{(\frac{\Delta t + d_{cur}}{\alpha})^3 + \frac{\Delta t + d_{cur}}{\beta} + r_0}{r} \times 100, & \text{r increased} \\ 100, & \text{Otherwise} \end{cases} \tag{4}$$

where $\Delta t = t_{cur} - t_0$ is the time elapsed since the start of the current increase epoch, $d_{cur} = d_{min}(t_{pend}, t_{cur})$ is an estimate of the current RTT. At times of congestion, this estimate can be smaller than actual RTT, which results in reduced pacing rate. However, this estimation error makes the pacing rate more close to the actual sending rate in one RTT.

IV. Evaluation

In this section, we compare and analyze the performance of FlexiS, LEDBAT and LEDBAT++. The LBE CCAs were tested in a variety of scenarios in both emulated networks and in the Internet. In both emulation and Internet tests, background or LBE flows were generated by bulk data transfer applications written in C. LBE data packets were only sent in one direction. The direction in which LBE data (ack) packets travel is referred to as forward (reverse) direction. In emulation tests, foreground flows were all BE TCP flows. Tcpdump was used to capture packets, Tcptrace was used to analyze trace files and bash scripts were used to run the experiments and produce results.

A. Performance Metrics

This subsection defines performance metrics. Except retransmission rate and Jain's fairness index, all metrics are only used in emulation tests.

Convergence time CT is defined in equation 5 as the time taken for an LBE connection to increase its CWND to BDP for the first time since connection establishment. BDP of an LBE connection is defined as the product of AB and base delay.

$$CT = t_c - t_s \tag{5}$$

where t_s is the establishment time of the LBE connection and t_c is the time when CWND of the LBE connection reaches its BDP for the first time.

Bandwidth utilization U measures the throughput of one or more LBE flows as a percentage of AB. It is defined in equation 6.

$$U = \frac{x_l}{B_a} \times 100 \tag{6}$$

where $B_a = C - B_u$ is the average AB, C is the bottleneck link capacity and B_u is the average bandwidth consumed by BE flows. B_u is measured by bmon at the bottleneck Network Interface Card (NIC). x_l is the **aggregate throughput** of all LBE flows that share one bottleneck. A utilization greater than 100% indicates over-use of AB, which implies that the LBE flows are "stealing" bandwidth from BE flows.

Throughput degradation TD measures what percentage of throughput BE flows lose due to the contention of LBE flows. It is defined in equation 7.

$$TD = \frac{x_o - x_w}{x_o} \times 100, \tag{7}$$

where x_o is the average throughput of BE flows when they run alone and x_w is the throughput of the BE flows when LBE flows run in the background.

QD measures the cumulative time a packet spends waiting in various bottleneck queues on its round trip route. Let QD_i be the QD measured by the i_{th} packet. Then $QD_i = RTT_i - RTT_{base}$. Because numerous QD samples can be obtained during an experiment, the 90_{th} percentile ($P_{90}(QD)$) is used to indicate the overall degree of QD.

Extra QD ($\Delta P_{90}(QD)$) is the extra amount of QD added by LBE flows in addition to the original QD inflicted by BE flows. It is calculated as the difference between the 90_{th} percentile QD measured by a BE flow when no LBE flows present ($P_{90}(QD_o)$) and when LBE flows run in the background ($P_{90}(QD_w)$). Equation 8 gives its definition.

$$\Delta P_{90}(QD) = P_{90}(QD_w) - P_{90}(QD_o), \tag{8}$$

Retransmission rate RR is defined in equation 9. It measures the number of retransmitted packets as a percentage of all packets sent. It is used as an estimation of loss rate.

$$RR = \frac{n_r}{n_t} \times 100, \tag{9}$$

where n_r is the total number of packets retransmitted by all LBE (or BE) connections that share one bottleneck and n_t is the total number of packets sent by the same connections.

Extra retransmission rate (ΔRR) is the extra amount of RR experienced by BE connections due to the competition of LBE connections. It is defined in Equation 10.

$$\Delta RR = RR_w - RR_o, \tag{10}$$

where RR_o is the RR of BE connections when they run without LBE connections and RR_w is the RR of the BE connections when LBE connections run in the background.

Jain's Fairness Index [38] (JFI) is used to measure the degree of equitable distribution of AB among contending flows. It is defined in equation 11.

$$JFI = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}, \tag{11}$$

where x_i is the average throughput of the i_{th} LBE flow, n is the number of LBE flows sharing the same bottleneck. JFI is between 0 and 1. The larger the value the higher the fairness.

B. Bulk Data Transfer Applications

Bulk data transfer applications written in C were used to generate data for LBE connections in both emulation and Internet tests. Multiple applications were written for different tests. However, the core functions are the same – the sender pre-fills its send buffer with data, once it is connected to the receiver, it sends the data in its buffer to the receiver without pause. This can to a large extent prevent the sender from being limited by data when AB is high. The receiver simply discards all the data it receives from the sender.

C. Implementation of LEDBAT and LEDBAT++

We have adapted the LEDBAT Linux kernel module implemented by Silvio Valenti [39] for their study [40] to conform to RFC6817. The adapted code is available at [41]. In our implementation, OWDs are calculated using the timestamp values and timestamp echo replies embedded in returning ACKs at the sender. However, a direct subtraction of these values is not viable, because the two timestamps might be generated by clocks with different resolutions. A special algorithm is used to estimate the receiver's Timestamp Clock Resolution (TCR). The accuracy of the estimator is seriously affected by congestion on the forward direction path. LEDBAT with this estimator enabled has persistently low throughput in various situations. Because the receiver's TCR are known in all our emulation and Internet tests, they are hardwired in LEDBAT's code. This ensures that the performance of LEDBAT is not affected by the estimator in all tests. The default values for LEDBAT parameters are set as follows. TARGET = 100 ms. BASE_HISTORY = 10. The same GAIN was used for both CWND increase and decrease and was set to 1. Noise filter was set to NULL. And MIN_CWND = 2.

We also implemented a Linux kernel module for LEDBAT++, which was adapted from LEDBAT. The implementation conforms to LEDBAT++ draft version 1 [4] and is available at [42]. The draft proposes a per RTT CWND reduction equation. However, it is very difficult to implement in that the amount to be decreased is affected by QD, which changes with each ACK. Therefore, we implemented the per ACK CWND reduction equation proposed in [35], base on which the CWND update functions of LEDBAT++ were developed. In our implementation, the modified slow start is used after periodic slow down. The default values for LEDBAT++ parameters are set as follows. TARGET = 60 ms. BASE_HISTORY = 10. Min filter is used as current delay filter and its length is 4. And MIN_CWND = 2.

D. Emulation Tests

In this subsection, we study the performance of FlexiS with a network emulator.

1) Emulation Tools: Emulation tests were conducted on virtual networks emulated by the Common Open Research Emulator (CORE) [43] [44]. CORE builds a representation of a real computer network that runs in real time and provides an environment for running real applications and protocols. All virtual nodes on a physical host run the same kernel and share the same set of resources. The limiting factor of performance is the number of times that the operating system needs to handle a packet but not the number of hops and the size of the packets.

Our main emulation platform was a desktop PC with an Intel dual-core 2.9 GHZ processor and a 4 GB memory installed with Ubuntu Desktop 20.04, Linux kernel v4.5, and CORE 7.0. We did not observe obvious performance degradation in most of the tests except the scalability test (section IV-D6), in which we noticed lowered link utilization of all LBE CCAs when bottleneck capacity is high (due to hardware limitation) and higher intrusion of FlexiS when bottleneck capacity is low (because that CORE 7.0 does not emulate transmission delay). After redoing the scalability test on a PC with a 10-core 2.5 GHZ processor and a 16 GB memory installed with Ubuntu Desktop 22.04, Linux kernel v5.19, and CORE 9.0 (which supports transmission delay), the LBE CCAs have normal performance.

The Multi-Generator (MGEN) [45] is a packet-level traffic generator capable of generating UDP flows and responsive TCP connections. Standard TCP/IP socket is used to establish a TCP connection between specified source and destination hosts. The source host generates packets based on provided Packet Size (PS) and packet Inter-Departure Time (IDT), which can be explicitly specified by setting a statistical distribution or implicitly extracted from a binary trace file (called trace cloning).

2) Traces Used: Analytical traffic models – albeit easy to obtain and manipulate – cannot accurately model real world traffic [46]. Therefore, in the majority of the emulation tests, we used traces of real traffic to load the virtual networks. The MAWI traffic archive [47] hosts a huge number of Internet traffic trace files captured at various sample points of the WIDE backbone in Japan since 1999. This makes it possible to load our virtual networks without sophisticated processing of the original traces. Because no single trace can represent Internet traffic as a whole, we intentionally selected a large number of traces with diverged bit rates, burstiness and composition. The traces chosen were all captured at sample point F, which is a transit link of WIDE to its upstream ISP in the U.S. TABLE I lists all the trace files used by the emulation tests.

TABLE I
TRACES USED BY EMULATION TESTS

Name	Year	$\overline{R}$	s	Name	Year	$\overline{R}$	s
wide11	2007	11.19	1.80	wide42	2020	42.99	20.76
wide25	2020	25.55	7.58	wide47	2009	47.64	25.57
wide28	2008	28.54	11.23	wide52	2009	52.47	7.33
wide33	2008	33.64	5.59	wide74	2010	74.42	10.81
wide36	2008	36.11	9.79	wide94	2009	94.48	6.07
wide37	2009	37.66	6.80	wide108	2010	108.41	8.59

Year specifies in which year a trace was captured. $\overline{R}$ is the average rate of the trace in Mbps. And s is the standard deviation of rate in Mbps.

3) The Dumbbell Topology: Unless otherwise specified, all emulation tests were conducted on a dumbbell topology (Fig. 2), which is adapted from the one originally proposed in [48]. The OWD of each link are annotated by the link. They are largely the same as the ones used in [48]. The intention of choosing these OWDs is to make the RTTs of the paths (shown as a column on the right hand side of Fig. 2) fall within the typical Internet RTT range.

In the dumbbell and other emulated networks, links are symmetrical. A link is denoted by the two nodes at both ends of it connected with a line segment as N1−N2. When there is only one path between two end hosts Hi and Hj, the path is denoted as Hi$\cdots$Hj. If there are more than one paths between two hosts, a path is denoted by all nodes on it connected with line segments. A flow is depicted by its source S and destination D connected with an arrow as S$\rightarrow$D.

The central link of the dumbbell R1−R2 is the bottleneck. Its capacity varies from one test to another. The capacity of peripheral links are 1 Gbps and they are invariant throughout all tests. The default queuing discipline (QDISC) is Tail-Drop First-In First-Out queue in unit of Packets (PFIFO). The bottleneck buffer defaults to 1.5 BDP of a 100 ms connection, which equals a maximum of 150 ms QD (chosen to be larger than LEDBAT's TARGET). LBE data and ACK flows are depicted in the graph.

Some tests require to load the dumbbell network with MAWI traces. In this case, two traces are needed with one (wideX) used as forward direction load and another (wideY) as reverse direction load. Because there are 9 paths in the network from one side to another, each of wideX and wideY is split into 9 sub-traces. During an experiment, MGEN is used to clone a live TCP connection from a sub-trace. In the end, there will be two TCP connections between each pair of hosts with their data traveling in reverse directions. Experiments showed

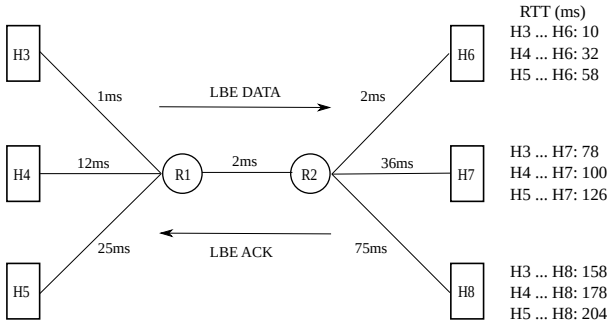


Fig. 2. Dumbbell topology adapted from [48]

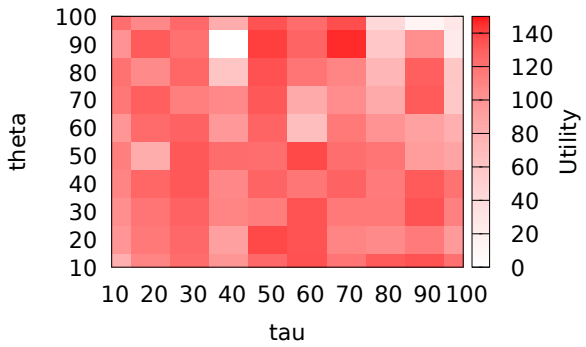


Fig. 3. Utility score of each τ and θ combination obtained from the first round selection

that the synthesized traffic at the bottleneck inherits the key characteristics of wideX (or wideY).

4) Tests Overview: All of the emulation tests were conducted on virtual networks emulated by CORE. The emulated routers used OSPF as routing protocol. Most of the BE flows were generated by MGEN, some by our bulk data transfer applications. TCP CUBIC was the default CCA for BE flows. LBE flows used one of the CCAs: FlexiS, LEDBAT without slow start (LEDBAT-BA), LEDBAT with slow start (LEDBAT-SS) and LEDBAT++. All LBE CCAs used their default parameter settings.

In each test, one network parameter (e.g. bottleneck capacity) is varied within a range. Other parameters are fixed. For each value in the range, an experiment is conducted. In each experiment, all LBE CCAs under investigation are tested in turn. In a typical experiment, the performance of the BE CCA (when it runs alone) is measured first. Then the BE CCA runs simultaneously with one of the LBE CCAs and the utilization and intrusion of the LBE CCAs are measured. When BE and LBE CCAs run simultaneously, the BE flows are started first. A break B is inserted between the start of BE and LBE flows. B is 60 seconds by default. The default starting interval I for LBE flows is 10 seconds. Unless otherwise specified, the performance measurement starts from the establishment of the first LBE flow and lasts until $300 + I$ seconds after the establishment of the last LBE flow. Each experiment was repeated for 10 times and the average is reported.

In all experiments, TCP send and receive buffers of the emulated nodes were adjusted so that they did not become the limiting factors of sending rate. CPU backlog queue was increased. tcp_no_metrics_save was enabled in order to make the performance of each LBE CCA independent of each other.

5) Parameter Selection: Before presenting the evaluation results, we would first describe how protocol parameters of FlexiS were selected.

Due to the large solution space, we used a method which progressively discards candidate values that yield bad performance. To be specific, we first set a range for each parameter based on common sense. For each candidate value within the chosen range, we evaluated them under a specific scenario. The candidate values that resulted in bad performance were excluded from further consideration. The rest of candidate values were further evaluated in other scenarios (different bottleneck capacities, QDISCs and loads etc.) until one candidate value was left for each parameter.

Using the above method, the default values for α, β and γ were set to 100, 10 and 0.85 respectively. Fig. 3 presents the results of the first round selection for τ and θ, each of which was varied within a range of [10, 100] with a step size of 10. α, β and γ were set to their default values.

The experiments were conducted on the dumbbell topology. The bottleneck capacity was set to 100 Mbps. QDISC of the bottleneck was PIE with a 15 ms target. This is different from the default QDISC. The intention is to ensure first and foremost that FlexiS does not fail when AQM is deployed. The forward direction load was wide42, and reverse direction load was wide25. Wide42 has a very high variation in rate. We chose it for the first round selection is to guarantee that FlexiS can meet its design objectives under harsh conditions like this. One FlexiS flow H4→H7 was investigated.

The depth of the color in the graph is determined by a numerical value called utility score ($uScore$), which is calculated as $uScore = U - TD^3 + 93$. The top 15% candidate values were chosen for further evaluation in other scenarios. $\tau = 60$, $\theta = 30$ gets a utility score of 134 in the first round selection, which ranks number 10 among all 100 combinations. It is finally chosen as the default because other combinations that have higher utility scores have lower scores in other scenarios. Due to space limitation, other selection results are not presented here.

6) Scalability Test: The goal of this test is to examine how well FlexiS scales to various ABs. The capacity of the bottleneck link was set to one of the values: 1, 5, 10, 50, 100, 500, and 1000 Mbps. The network was not loaded with any BE flows. One LBE flow H3→H6 was examined. The performance of the LBE CCAs are shown in Fig. 4. Utilization and QD were measured after convergence for 100 seconds.

Convergence time of all LBE CCAs increase with the increase of AB. LEDBAT-SS and LEDBAT++ take less time to acquire large amounts of bandwidth due to the use of slow start, while FlexiS and LEDBAT-BA are comparatively slower in acquiring large amounts of bandwidth.

LEDBAT (henceforth, used to represent both LEDBAT-BA and LEDBAT-SS) has high steady state utilization in all tested scenarios. FlexiS achieves equivalent utilization when bottleneck capacity is high ($\geq$ 50 Mbps) but has reduced

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

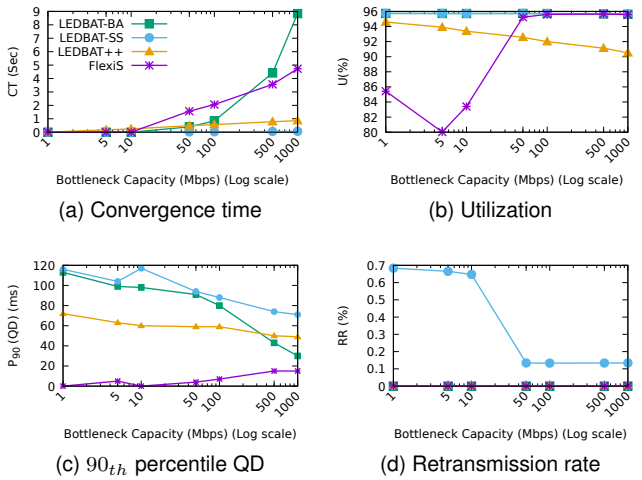


Fig. 4. Results of the scalability test

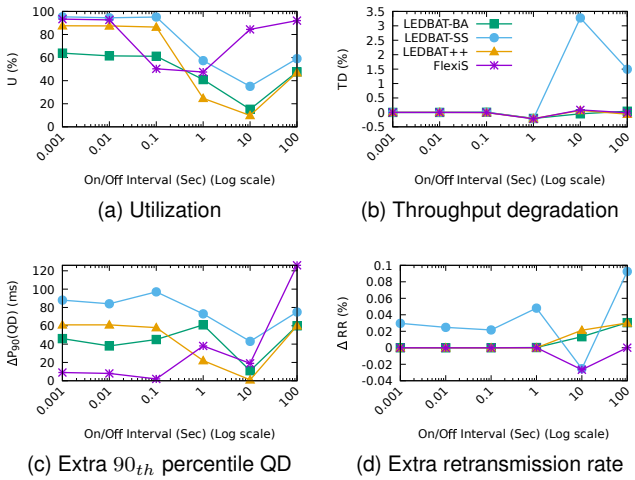


Fig. 5. Responsive test results

utilization otherwise. This is because when bottleneck capacity is small, the same amount of packets in the bottleneck queue can translate into higher QD, which will make FlexiS reduce its rate by multiple times until its rate is well below bottleneck capacity. And it may take a substantially long time for it to increase its rate back to the AB.

The $P_{90}(QD)$ induced by LEDBAT-SS and LEDBAT++ are around their respective targets. FlexiS induces the least amount of $P_{90}(QD)$ among all LBE CCAs ($max(P_{90}(QD) = 15$ ms). LEDBAT-SS suffers from higher RR than the rest of LBE CCAs due to the use of the unmodified slow start. In comparison, FlexiS has very low RR ($max(RR) = 0.00027\%$).

7) Responsiveness Test: The goal of this test is to examine how well FlexiS responds to changes in AB. Bottleneck capacity was set to 100 Mbps. The network was loaded with an on/off BE flow H4→H7, which was generated by MGEN as a responsive TCP connection. It transferred at a constant packet rate of 75 Mbps during the "on" period and 0 Mbps during the "off" period. The "on" and "off" periods had the same duration, which was set to one of the values: 0.001, 0.01, 0.1, 1, 10 and 100 seconds. One LBE flow H5→H8 was studied. The starting interval between BE and LBE flows was 10 seconds. The performance of the LBE CCAs are shown in Fig. 5.

The utilization of all LBE CCAs are affected by on/off intervals of the BE flow. FlexiS has lower utilization when the on/off intervals are 0.1 and 1 seconds. Other LBE CCAs have reduced utilization when the interval is equal to or greater than 1 second. When the on/off durations are 0.1 and 1 second, the "on" period is long enough to make FlexiS detect the rate increase of the BE flow and reduce its rate. In the meanwhile, the "off" period is too short to allow FlexiS to fully utilize bandwidth released during this period.

Except LEDBAT-SS, all LBE CCAs cause negligible throughput degradation of the BE flow. Generally, the extra $P_{90}(QD)$ induced by LEDBAT and LEDBAT++ have a positive correlation with their utilization. FlexiS induces low extra $P_{90}(QD)$ in most cases. All LBE CCAs add negligible retransmission rate to the BE flow.

LBE flows with different RTTs were also studied. FlexiS has similar performance irrespective of its RTT. Whereas, LEDBAT and LEDBAT++ are more affected by RTT. Their utilization and intrusion increase with the decrease of RTT. Due to space limitation, the results are not shown here.

8) Forward Direction Load Test: This test studies how FlexiS performs with varying levels of traffic in its forward direction. Two groups of experiments were conducted. In both groups, the bottleneck link capacity was set to 100 Mbps. The network was loaded with MAWI traces. The forward direction load was one of wide11, wide33, wide52, wide74, wide94 and wide108. The reverse direction load was always wide11. These traces all have low variation in rate. The intention is to ensure load stability while we are introducing burstiness to the network. The first group of experiments examined the performance of one ($n = 1$) LBE flow H4→H7. While the second group investigated nine ($n = 9$) LBE flows, which used three different paths: H3$\cdots$H6, H4$\cdots$H7, and H5$\cdots$H8. There were three flows on each path. Fig. 6 shows the results. Utilization for the 108% loaded scenario is not shown in the figure since there is no AB left in this case and utilization cannot be calculated.

The utilization of a single FlexiS or LEDBAT++ flow declines drastically from around 90% to slightly over 40%. In a different test, we replaced MAWI traces with constant packet rate BE flows. The results showed that FlexiS has a pretty stable utilization (around 87%) for all loads. This indicates that FlexiS' low utilization in this test is caused by the burstiness of the cross traffic. When a packet burst arrives, FlexiS yields all demanded bandwidth to the cross traffic. When the burst leaves, FlexiS cannot timely absorb all bandwidth released by the burst because it does not leave large backlog in bottleneck buffer and it increases its rate conservatively after rate reduction. We measured the bandwidth wasted (B_w) by FlexiS and they do not differ greatly for different loads. However the utilization reduces significantly because when B_w is given, utilization reduces with the decrease of AB. Fig. 6b shows that multiplexing can improve utilization of both FlexiS and LEDBAT++.

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

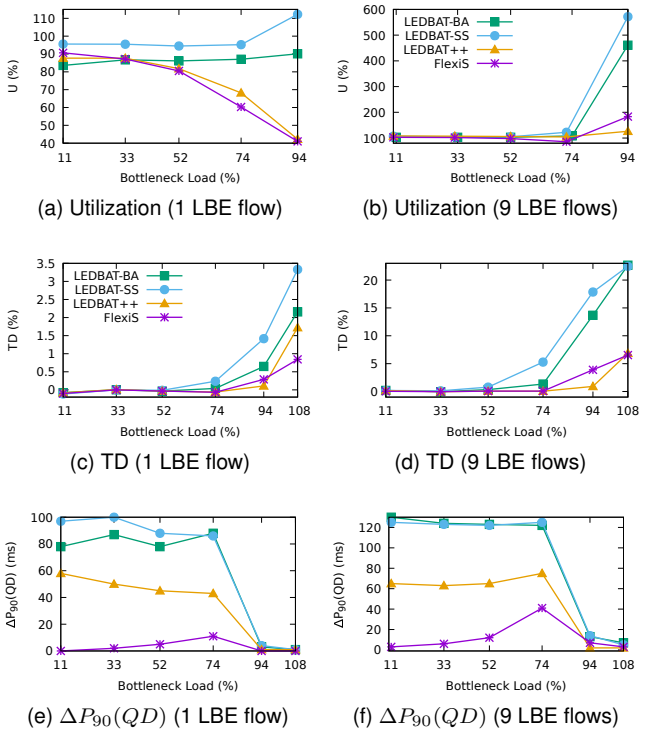


Fig. 6. Impact of forward direction load on the performance of LBE CCAs. NB the Y axes of figures on the left and right hand side columns have different ranges.

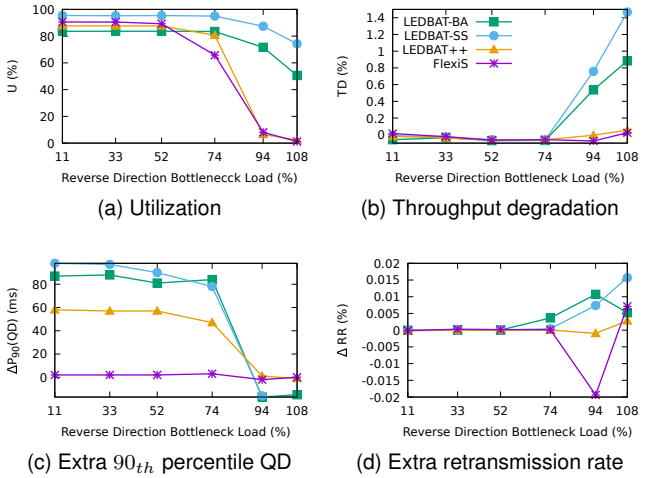


Fig. 7. The impact of reverse direction load on the performance of LBE CCAs

A single FlexiS flow has negligible impact on BE flows. Nine FlexiS flows inflict higher TD when the bottleneck is saturated but do not increase $\Delta P_{90}(QD)$ significantly. The higher TD (6.5% at 108% load) is caused by two factors: (1) FlexiS does not decrease CWND below 2 MSS; (2) when FlexiS cannot obtain σ RTT samples when $L_D \geq \tau$, it will increase CWND.

Increasing the number of LEDBAT connections can also increase their intrusion. When the bottleneck is 108% loaded, nine LEDBAT-BA connections take away almost one fourth of bandwidth (22.6%) from BE flows. LEDBAT and LEDBAT++ inflict higher extra $P_{90}(QD)$ than FlexiS in most cases. Retransmission rate of the BE flows is not seriously affected by the LBE CCAs. In particular, FlexiS inflicts nearly no extra RR in most cases. Therefore, the graphs are omitted.

9) Reverse Direction Load Test: This test examines the performance of FlexiS when traffic level in the reverse direction varies. The bottleneck link capacity was set to 100 Mbps. The network was loaded with MAWI traces. The forward direction load was always wide11 and the reverse direction load was one of the traces: wide11, wide33, wide52, wide74, wide94 and wide108. One LBE flow H4→H7 was studied. The results are shown in Fig. 7

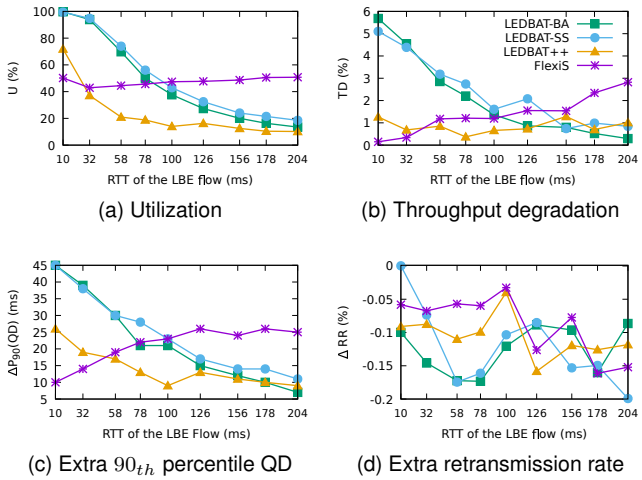
All LBE CCAs have high utilization when the reverse direction load is low ($<= 52\%$). FlexiS and LEDBAT++ have more throughput drop when the load in the reverse direction increases. In another test, we replaced MAWI traces with constant packet rate BE flows, and discovered that the utilization of all LBE CCAs do not drop until the reverse direction bottleneck gets overloaded (110% load), in which case, FlexiS' utilization drops mildly to 71%. This suggests that on the one hand, FlexiS responds to reverse direction congestion, and on the other, its utilization is affected by the burstiness of the cross traffic. Due to the use of OWD in congestion detection, LEDBAT is less affected by reverse direction congestion. However, this is only achievable in controlled environments, where the receiver's TCR is known. In practice, its utilization is determined by the accuracy of the TCR estimator.

Because the forward direction BE flows are nearly not affected by the LBE CCAs, intrusion measures how intrusive of the LBE CCAs to the reverse direction BE flows. FlexiS has negligible impact on the BE flows, while LEDBAT and LEDBAT++ inflict slightly higher intrusion.

10) RTT Test: The goal of this test is to investigate the impact of RTT on the performance of FlexiS. We have tested the LBE CCAs with constant packet rate BE cross traffic and real traffic traces with low variation in rate, in this test, we will test them with traces that have higher variation in rate. Therefore, we used wide42 (stddev = 20.76) as the forward direction load and wide25 as the reverse direction load. In order to facilitate cross-test comparison, this pair of traces were also used in most of the tests presented in the rest of this subsection. The capacity of the bottleneck link was set to 100 Mbps. One LBE flow was examined and it took one of the nine paths in turn. Please refer to Fig. 2 for the RTT of each path. The results are shown in Fig. 8.

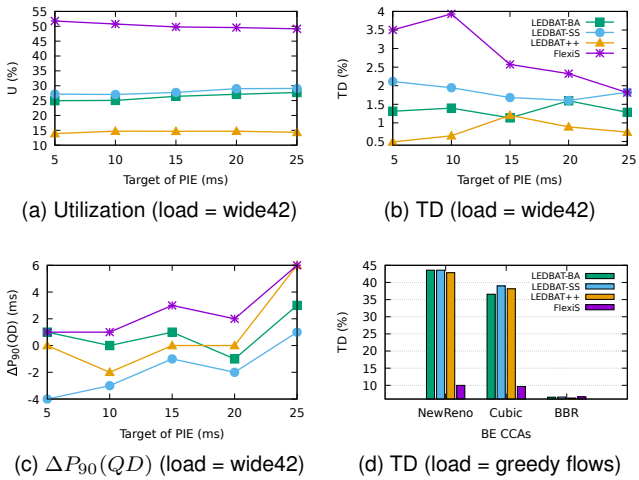
FlexiS maintains a quite stable bandwidth utilization (around 50%) irrespective of its RTT. In contrast, the utilization of LEDBAT and LEDBAT++ decrease significantly with the increase of RTT. The reason is that FlexiS adopts an RTT-independent increase function while LEDBAT uses an RTT-dependent increase function. The utilization of all LBE CCAs decline due to the increased burstiness of the cross traffic. This can be easily observed by comparing the utilization of 100 ms connections in Fig 8a with those when load is 33% or 52% in Fig 6a. The reason in the case of FlexiS is that

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441



(a) Utilization (b) Throughput degradation (c) Extra 90_{th} percentile QD (d) Extra retransmission rate

Fig. 8. The impact of RTT on the performance of LBE CCAs



(a) Utilization (load = wide42) (b) TD (load = wide42) (c) $\Delta P_{90}(QD)$ (load = wide42) (d) TD (load = greedy flows)

Fig. 9. The impact of AQM on the performance of LBE CCAs. The default target of PIE is 15 ms.

the increased amplitude of oscillation of AB causes more bandwidth wastage. To be specific, as a response to a bigger AB reduction FlexiS will release more bandwidth, but as a reaction to a lager AB increase it will not absorb more because the AB will not remain high for long enough duration to allow FlexiS to do so.

The intrusion of FlexiS increases moderately with the increase of RTT. This is caused by the difference in feedback delay for connections with different RTTs, and by the fact that FlexiS keeps increasing its rate before the arrival of congestion feedback. In comparison, the intrusion inflicted by LEDBAT increases with the decrease of RTT.

11) AQM test: This test studies how Active Queue Management (AQM) algorithms affect the performance of FlexiS. LEDBAT has been shown to have a reprioritization problem – it becomes as aggressive as TCP NewReno – when AQM is deployed at the bottleneck [15]. In this test, we study if FlexiS also suffers from similar problems. In the first group of experiments, the capacity of the bottleneck link was set to 100 Mbps. Wide42 was used as forward direction load and wide25 as reverse direction load. The QDISC of the bottleneck router was set to PIE [49]. Limit of PIE was set to 1250 packets, which corresponds to a 150 ms hard limit on QD. Target of PIE was set to 5, 10, 15, 20 or 25 ms. One LBE flow H4→H7 was studied. Fig. 9a, 9b, and 9c show the results.

The utilization of the LBE CCAs do not change noticeably with the adjustment of PIE's target. If we compare the utilization of 100 ms connections in Fig 8a with those in Fig 9a, it is not difficult to observe that switching from PFIFO to PIE does not considerably alter the utilization of FlexiS and LEDBAT++, but reduce that of LEDBAT-SS and LEDBAT-BA by 35% and 32% respectively. The major reason for the difference is that FlexiS and LEDBAT++ respond to congestion earlier and more aggressively, so they can avoid many big rate reductions caused by packet loss.

FlexiS has slightly higher impact on the BE traffic compared to other LBE CCAs. Generally speaking, the LBE CCAs inflict lower TD but higher extra $P_{90}(QD)$ with the increase of PIE's target. Because they have little impact on the BE connections' retransmission rate, the corresponding graph is not shown.

In the second group of experiments, the bottleneck capacity was set to 10 Mbps. QDISC was set to PIE with its target set to 15 ms. The network was loaded with one greedy BE flow H5→H7. It was generated by one of the bulk data transfer applications described in subsection IV-B. Its CCA was one of the following: NewReno, CUBIC and BBR. One LBE flow H5→H7 was studied. The starting interval between the BE and LBE flows was 10 seconds. The results are shown in Fig. 9d.

When the BE flow uses either NewReno or CUBIC, LEDBAT and LEDBAT++ transform into BE CCAs while FlexiS preserves its LBE property. The NewReno flow loses 43.58% and 42.84% of throughput, while the CUBIC flow loses 36.57% and 38.15% of throughput to the LEDBAT-BA and LEDBAT++ flow respectively. In comparison, TD inflicted by FlexiS on NewReno and CUBIC are 9.97% and 9.66% respectively. BBR is the most aggressive BE CCA among the three. No LBE CCA can take a noticeable fraction of bandwidth from it.

12) Fairness Test: This test assesses FlexiS' capability of fairly sharing AB between its own flows. Both intra- and inter-RTT fairness are examined. The first two groups of experiments examine intra-RTT fairness. In the first group, the network was unloaded. The goal is to examine fairness in an environment without any interference from cross traffic. In the second group, the network was loaded with wide11 in both directions. The goal is to study fairness in a more realistic environment and analyze how cross traffic affect fairness of the LBE CCAs. In both groups, the capacity of the bottleneck link was set to 20 Mbps. The bottleneck buffer was set to 4 BDP of a 100 ms connection. Fairness between three LBE flows was measured. All of them used the path H4$\cdots$H7. LBE flows were started at 30-second intervals. Fairness was measured for 600 seconds. The results are shown in Fig. 10.

FlexiS and LEDBAT++ achieve high intra-RTT fairness in both unloaded and loaded scenarios with FlexiS' $JFI > 0.99$ in both scenarios. While LEDBAT flows cannot fairly share AB when the network is unloaded, which is the result of the so

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

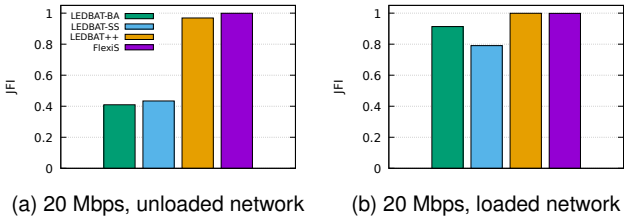


Fig. 10. Intra-RTT fairness of LBE CCAs

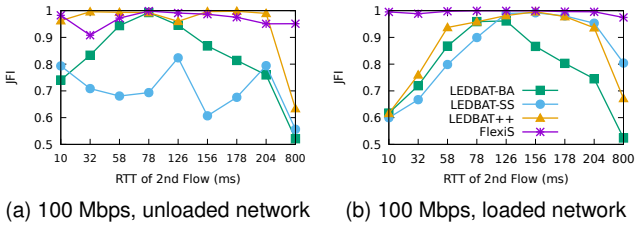


Fig. 11. Inter-RTT fairness of LBE CCAs. RTT of the first LBE flow: 100 ms

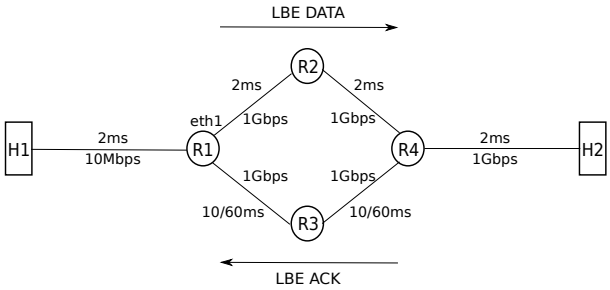


Fig. 12. A Diamond Topology

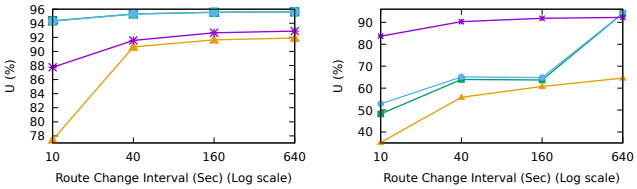


Fig. 13. Results of route change test. NB the Y axes of figures on the left and right hand side columns have different ranges.

called late comer advantage problem [3] [13] – the late coming flow takes the QD maintained by early coming flows as part of base delay, therefore become more aggressive than precedent flows. In the loaded scenario, BE flows introduce more delay dynamics, which gives the late coming flows opportunity to discover true base delay. Therefore fairness of LEDBAT is improved.

The third and fourth groups of experiments evaluate inter-RTT fairness. In the third group, the network was unloaded. In the fourth group, Wide42 was used as forward direction load and wide25 as reverse direction load. In both groups of experiments, the capacity of the bottleneck link was set to 100 Mbps. The bottleneck buffer was set to 4 BDP of a 100 ms connection. The fairness between two LBE flows was studied. Starting interval of LBE flows was 10 seconds in the third group and 30 seconds in the fourth group. Fairness was measured for 300 seconds. The first LBE flow had a fixed path H4$\cdots$H7 and the second LBE flow used one of the eight paths that is different from the one used by the first flow plus one more path (not shown in Fig. 2), which has an RTT of 800 ms (simulating a satellite link). Fig. 11 presents the results.

FlexiS has slightly lower JFI in an unloaded network. This is because that flows may take RTT samples at different times and with different frequencies. When the change in RTT is small (which is typical in an unloaded network), it may be detected by one flow (usually the one with a shorter feedback delay) but not another, which makes the former reduce rate more frequently and consequently obtain less bandwidth share. In a loaded network, packet bursts from cross traffic can make RTT increase more aggressively, which can be detected by all contending FlexiS flows.

Fairness of LEDBAT-BA is affected by the difference in RTT. The larger the difference, the lower the JFI. LEDBAT-SS has a very unstable performance in an unloaded network. Its JFI varies greatly between runs of the same experiment and between different experiments. Generally, cross traffic has no significant impact on LEDBAT-BA but can improve the fairness of LEDBAT-SS. In an unloaded network, LEDBAT++ flows having typical RTTs can achieve higher fairness. When a LEDBAT++ flow's RTT falls outside [7.5, 120] ms, it will not rate adjusted effectively, so fairness declines. Cross traffic has adverse impact on the fairness of LEDBAT++.

13) Route Change Tests: An Internet route between two end hosts may change during the lifetime of a TCP connection due to the moving of an end host or failure of a link. This test examines the robustness of FlexiS in the presence of route change. The topology used is diamond, which is illustrated in Figure 12.

The bottleneck link is H1$-$R1. It has a capacity of 10 Mbps. The capacity of the rest of links are 1 Gbps. The OWD of links are annotated in the graph. The bottleneck router buffer is such set that the maximum QD is 150 ms. There are two paths between H1 and H2. The first path is P1: H1$-$R1$-$R2$-$R4$-$H2. And the second is P2: H1$-$R1$-$R3$-$R4$-$H2. The default path is P1. The OWD of P1 is 8 ms. P2's OWD was set to 24 ms in the first group of experiments and to 124 ms in the second. The network was not loaded with any BE flows in all experiments.

A simple route change scenario was studied: a link is being announced alternatively as up and down by a router due to the malfunction of a NIC. In our specific case, eth1 of R1 is the malfunctioning NIC and R1$-$R2 is announced as up and down alternatively. When eth1 is down, route between H1 and H2 is automatically switched from P1 to P2 by the routers. When eth1 is brought up, P1 is used again. The initial state of eth1 was set to on and it was brought down and then up again during an experiment. Eth1 of R1 stayed in up or down for the same amount of time. Up/down duration F was set to 10, 40, 160 or 640 seconds. A single LBE flow H1$\rightarrow$H2 was studied. It ran for $3 \times F$ seconds. The results are shown in Fig. 13.

The utilization of FlexiS is almost not affected by the

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

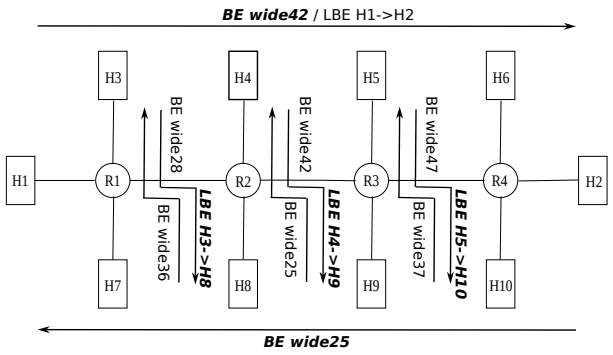


Fig. 14. A Parking Lot Topology

OWD of P2 (maintained between 84% and 93% for both scenarios) and only slightly impacted by up/down duration. In comparison, the utilization of LEDBAT and LEDBAT++ are seriously affected by OWD of P2. When up/down duration is 10 seconds, their utilization drop by approximately 50% when OWD of P2 is increased from 24 ms to 124 ms. This is because when the difference between the OWDs of P1 and P2 is greater than or equal to the targets of LEDBAT and LEDBAT++, the two LBE CCAs will take this difference as QD and keep decreasing rate until base delay is updated. Because base delay of LEDBAT and LEDBAT++ can be updated to the OWD of P2 when up/down duration is 640 seconds, their utilization is improved greatly. The retransmission rate of the LBE CCAs except LEDBAT-SS are negligible, so it is now shown here.

14) Multiple Bottlenecks Test: In a more realistic scenario, a data flow can traverse multiple congested gateways and wait in various queues for transmission. This test investigates the performance of FlexiS in such scenarios. The topology (Fig. 14) used is adapted from the parking lot topology originally proposed in [50]. The bottlenecks are the central links R1−R2, R2−R3 and R3−R4. The capacity of the central links are 100 Mbps and those of the peripheral links are 1 Gbps. Except links H1−R1 and R4−H2, which have 0 ms OWDs, the rest of links all have 10 ms OWDs. The buffer of all bottleneck routers are set to 2.5 BDP of a 60 ms connection, which corresponds to a maximum QD of 150 ms.

In the first group of experiments (flow notations are shown in bold italic font in Fig. 14), a BE flow H1→H2 (cloned from wide42) runs across multiple congested gateways. Another BE flow H2→H1 (cloned from wide25) runs in the reverse direction. Three LBE flows H3→H8, H4→H9, and H5→H10 pass through one of the gateways each. Fig. 15 presents the results.

The utilization of the LBE CCAs is not seriously affected by which bottleneck they traverse. LEDBAT has comparatively higher utilization than FlexiS and LEDBAT++ mainly because it increases faster after rate reduction in this specific scenario (RTT = 60 ms). In the meantime, it also inflicts much heavier intrusion on the BE flow. With similar intrusiveness, FlexiS achieves higher utilization than LEDBAT++.

In the second group of experiments, 32 LBE flows run through multiple bottlenecks, with each bottleneck loaded with different BE traffic (annotated in Fig. 14 in normal font). The traces were so chosen that the amount of AB and variation in

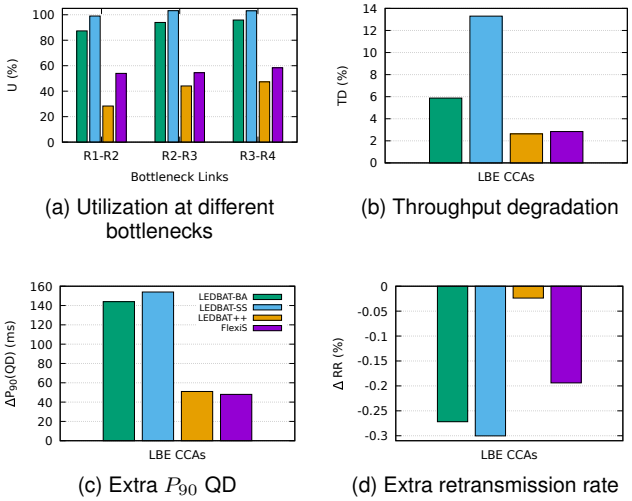


Fig. 15. Performance of LBE CCAs when BE flows run through multiple congested gateways

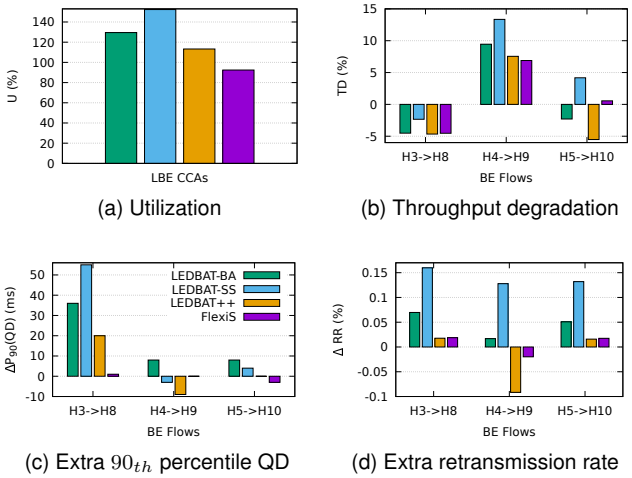


Fig. 16. Performance of LBE CCAs when 32 LBE flows run through multiple bottlenecks

AB at each bottleneck are different. The intention is to create a virtual network that bears a closer resemblance to the real Internet. Fig. 16 shows the results.

Thirty-two FlexiS flows have a utilization of 92.44%, a $max(TD)$ of 6.89%, a $max(\Delta P_{90}(QD))$ of 1 ms and a $max(\Delta RR)$ of 0.018%. All other LBE CCAs over-utilize AB. As a result, they are more intrusive to the BE flows.

The LBE CCAs have lower utilization and intrusion when only 1 flow runs across multiple bottlenecks. The results are not presented here due to space limitation.

E. Internet Tests

The goal of the Internet test is to study the performance of FlexiS in a realistic environment. The LBE CCAs were tested on two Internet paths – Data center to Data center (D2D) and Data center to Residence (D2R). On each end host, TCP send and receive buffers were expanded and CPU backlog queue limit was increased. Tcp_no_metrics_save was set to 1.

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

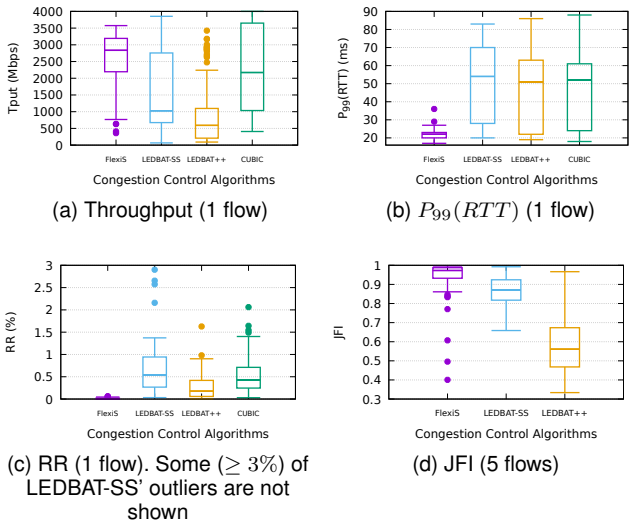


Fig. 17. Data center to data center communications

For the D2D path, the sender was a Linode virtual machine locating in Atlanta USA (referred to as atlanta.taht.net). The receiver was another Linode virtual machine in Newark USA (referred to as newark.taht.net). The observed minimum RTT of this path was 17 ms. The throughput was upper bounded by the sender's subscription plan to 4 Gbps. The two machines were installed with Ubuntu Desktop 22.10 and Linux kernel v5.19.0.

Two experiments were conducted on this path. The first experiment examines the throughput, RTT and retransmission rate of a single LBE flow. The second studies throughput fairness between five competing LBE flows of the same kind. The first experiment was carried out on 2023.03.26, 2023.03.28, 2023.03.30 and 2023.04.01. The dates were so chosen that both work days and weekend were sampled. On an experimental day, a random time was chosen to launch the experiment. Every hour, the sender made four 60-second bulk data transfers to the receiver using each of the CCAs in turn: FlexiS, LEDBAT-SS, LEDBAT++, and CUBIC. The starting order of the CCAs was randomized. There were at least 15 seconds of break between the run of each CCA.

The average throughput, 99_{th} percentile RTT and retransmission rate of each CCA obtained from all ninety-six runs of the experiment are summarized in a box-whisker plot shown in Fig. 17. The three bars of a box from bottom to top mark the 25th, 50th and 75th percentiles respectively. The whiskers extend to 1.5 times of inter-quartile range. The two bars at the ends of the whiskers are minimum and maximum values within the whisker delimited boundary. Values outside the boundary are considered as outliers, which are represented as solid dots in the figure.

As is shown in Fig. 17a, FlexiS is able to achieve comparatively higher throughput in most of the runs. The median throughput of FlexiS, LEDBAT-SS, LEDBAT++ and CUBIC are 2835, 1028, 590, and 2171 Mbps respectively. The inter-quartile ranges for the CCAs in the same order are 997, 2083, 889, and 2615 Mbps. After analyzing the trace files, we discovered that in most of the runs all CCAs except FlexiS encountered a series of packet losses after slow start, which forced them to reduce their rates well below AB and from then on further losses prevented them from increasing their rates to the AB. In contrast, by omitting slow start and responding to congestion earlier, FlexiS could avoid many long QDs (Fig. 17b) and packet losses (Fig. 17c), therefore could keep its throughput at a much higher value.

The second experiment was carried out on 2023.04.22, 2023.04.26, 2023.04.28, and 2023.05.01. On each experimentation day, a random time was chosen to start the experiment. Every hour, the sender woke up and established five connections with the receiver using one of the LBE CCAs in turn: FlexiS, LEDBAT-SS, and LEDBAT++. There was a 10-second starting interval between consecutive flows of the same CCA. The measurement for fairness started from 10 seconds after the establishment of the last connection and lasted for 60 seconds. The starting order of the LBE CCAs was randomized. There were at least 15 seconds of break between each LBE CCA. JFI of all ninety-six runs are presented in box-whisker plots, which is shown in Fig. 17d.

In most of the runs, FlexiS has higher JFI than other LBE CCAs. The median JFI of FlexiS is 0.97 and that of LEDBAT-SS and LEDBAT++ are 0.87 and 0.56 respectively. Based on our observation, some of the low JFIs obtained by FlexiS might be the result of flows not having the same bottleneck. We examined the trace file of the 69_{th} run, in which FlexiS obtained its minimum JFI of 0.4. We discovered that for the flow having the least bandwidth share, the minimum, maximum, average, and standard deviation of RTT are 21.6, 353.4, 127.7, and 103.2 ms as reported by Tcptrace. And the corresponding values for the flow with the maximum bandwidth share are 19.5, 73.4, 20.6, and 1.8 ms. This suggests that the flows might have used different paths or have been treated differently by the network. For other low JFI runs, the difference in RTT statistics are less obvious. However, due to the lack of control over the Internet paths under investigation, we cannot prove that the flows had the same bottleneck.

LEDBAT++ has very low JFI in most of the runs. Unlike FlexiS, LEDBAT++ obtains its highest JFI (0.97) when flows have remarkably distinctive RTT statistics. LEDBAT-SS has comparatively higher JFI than LEDBAT++. Considering that it also has higher RR, we believe that frequent packet losses and faster rate increase after reduction improved its fairness.

In the D2R experiment, newark.taht.net was the sender and a home PC locating in mainland China was the receiver. The observed minimum RTT was 218 ms and the throughput was limited by the receiver's subscription plan to 100 Mbps. The home PC was installed with Ubuntu Desktop 22.04 and Linux kernel v5.19.0. The experiment was carried out on 2023.04.10, 2023.04.12, 2023.04.14, and 2023.04.16. On an experimentation day, a random time was chosen to start the experiment. Every hour, the sender made four 60-second bulk data transfers to the receiver using each of the CCAs in turn: FlexiS, LEDBAT-SS, LEDBAT++, and CUBIC. The starting order of the CCAs was randomized. There were at least 15 seconds of break between the run of each CCA. Total 96 runs were conducted, among which, 11 are excluded from analysis,

This article has been accepted for publication in IEEE/ACM Transactions on Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNET.2023.3319441

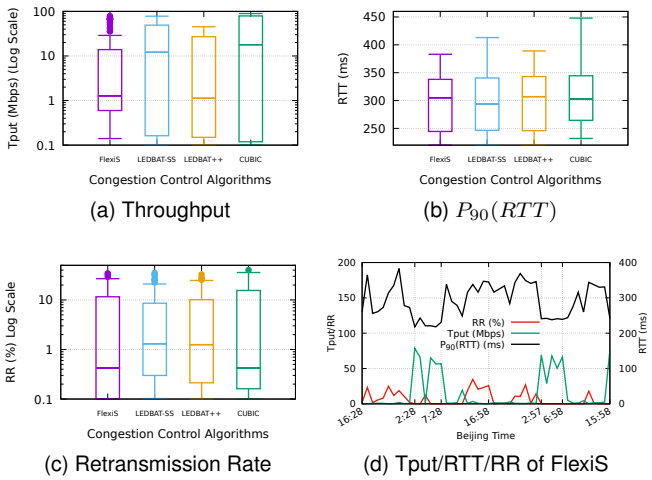


Fig. 18. Data center to residence communications

because in these experiments the sender could not get any ACK from the receiver. The results are summarized in Fig. 18.

FlexiS and LEDBAT++ have overall lower throughput compared to LEDBAT-SS and CUBIC. The median throughput of FlexiS and LEDBAT++ are 1.28 Mbps and 1.12 Mbps respectively and that of LEDBAT-SS and CUBIC are 12.1 Mbps and 17.9 Mbps respectively. The three LBE CCAs have similar RTT distributions: the $P_{90}(RTT)$ is greater than 246 ms ($>$ min(RTT)) in 75% of runs. And they have abnormally high retransmission rates in at least 25% of the runs. To be specific, the 75th percentiles of RR are 11.6%, 8.6%, 10.1%, and 15.5% for FlexiS, LEDBAT-SS, LEDBAT++, and CUBIC respectively. Because no abnormality was observed in the traces, we believe that the high QD and RR were the result of severe congestion.

Fig. 18d shows the evolution of throughput, $P_{90}(RTT)$ and retransmission rate of FlexiS during 48 hours between April 10 and 13. Other CCAs are not shown for visual clarity. It shows clearly that FlexiS has a diurnal behavior – it has high throughput, low RTT, and low RR in the late nights and the early mornings (period I) but the opposite performance in the rest of the days (period II). It is not difficult to discover that the RTT in period II is always higher than that in period I, which suggests that the network was congested in period II. Therefore, the low throughput of FlexiS in period II should be the result of yielding bandwidth to cross traffic at times of congestion. While the high throughput of LEDBAT-SS and CUBIC in period II should be the result of competition.

V. Conclusion

We have proposed a novel LBE CCA named FlexiS. It monitors RTT through a sliding observation window. If the trend in RTT within the window is increasing, it assumes that the network is congested, it will then reduce its rate by a fixed percent. It will otherwise increase its sending rate according to a cubic function of time.

Extensive emulation tests and preliminary Internet experiments showed that FlexiS has the following properties. First, In most cases, it inflicts very low intrusion to cross traffic. The only exception is when AB is very small or zero. Two solutions can be taken to mitigate this problem: (1) instead of increasing rate when $N_D < \sigma$, we can temporarily increase L_D until $N_D \geq \sigma$; and (2) decrease CWND below 2 MSS. We will leave the investigation into this issue as a future work.

Second, it scales to ABs that differ by several orders of magnitude. In the meantime, it also has a slower convergence time than slow start based LBE CCAs. The utilization of FlexiS is not affected by RTT, AQM and the amount of AB but by bottleneck capacity and the oscillation of AB. Usually, the smaller the bottleneck capacity, the lower the utilization. High frequency and amplitude of oscillation of AB can also reduce its utilization. However, the compromised utilization is an unavoidable trade-off for low intrusion. A temporary solution to improving utilization without significantly increasing intrusion is to establish multiple connections between a sender and a receiver.

Third, reverse direction congestion can reduce FlexiS' utilization in the forward direction, because RTT is used in congestion detection. One solution is to make routers to give ACK packets higher priority. But that will require the support of routers. Another solution is to replace RTT with OWD. However, OWD is difficult to estimate without the support of the receiver because the receiver's TCR is unknown to the sender. Further, the presence of non-zero clock skew and drift are more difficult to tackle. A third option would be to search for a variable to replace RTT or OWD in congestion detection. This variable should be easy to obtain and not affected by reverse direction congestion. We will leave it as a future work.

Fourth, FlexiS has high intra-RTT fairness in both loaded and unloaded networks regardless of bottleneck capacity. It also has high Inter-RTT fairness in most situations. However, in an unloaded network with small bottleneck capacity, FlexiS flows with large RTT difference will have reduced fairness.

Fifth, FlexiS preserves low priority when AQM is deployed at the bottlenecks.

Sixth, FlexiS adapts to route changes quickly. Its performance is almost not affected by RTT difference in alternative routes but slightly impacted by route change interval.

Seventh, because FlexiS employs a Theil-Sen estimator to estimate the trend in RTT, it has comparatively higher storage and computation demands.

Finally, it is a sender side only CCA, which makes it easy to deploy.

Acknowledgments

The author would like to thank the anonymous reviewers for their time and effort in providing valuable feedback to this paper. We truly appreciate the help of Dave Täht on making their Linode virtual machines atlanta.taht.net and newark.taht.net available to us for the Internet tests. The author would also like to thank Michael Welzl for the introduction of LBE services and the CORE emulator and for the suggestions and advice given to earlier design proposals of the LBE CCA. This work is a continuation of a previous effort in designing an LBE CCA, which was funded by the University of Oslo.

REFERENCES

[1] A. Venkataramani, R. Kokku, and M. Dahlin, "Tcp nice: a mechanism for background transfers," in *USENIX OSDI*, 2002.

[2] A. Kuzmanovic and E. W. Knightly, "Tcp-lp: low-priority service via endpoint congestion control," *IEEE/ACM Transactions on Networking*, 2006.

[3] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low extra delay background transport (rfc6817)," https://datatracker.ietf.org/doc/html/rfc6817, 2012.

[4] "Ledbat++: Congestion control for background traffic," https://datatracker.ietf.org/doc/html/draft-irtf-iccrg-ledbat-plus-plus-01, 2020.

[5] M. Bagnulo, A. Garcia-Martinez, G. Montenegro, and P. Balasubramanian, "rledbat: receiver-driven low extra delay background transport for tcp," https://datatracker.ietf.org/doc/html/draft-irtf-iccrg-rledbat-03, 2022.

[6] D. Havey, "Ledbat background data transfer for windows," https://techcommunity.microsoft.com/t5/networking-blog/ledbat-background-data-transfer-for-windows/ba-p/3639278.

[7] Q. Li, "Source code of flexis," https://github.com/tinalee77/FlexiS.

[8] R. Jain, "A delay based approach for congestion avoidance in interconnected heterogeneous computer networks," *ACM Computer Communication Review*, 1989.

[9] Z. Wang and J. Crowcroft, "A new congestion control scheme: Slow start and search (tri-s)," *ACM Computer Communication Review*, 1991.

[10] ——, "Eliminating periodic packet losses in 4.3–tahoe bsd tcp congestion control," *ACM Computer Communication Review*, 1992.

[11] L. Brakmo, S. O'Malley, and L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," in *ACM SIGCOMM*, 1994.

[12] H. Im, C. Joo, T. Lee, and S. Bahk, "Receiver-side tcp countermeasure to bufferbloat in wireless access networks," *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 2016.

[13] G. Carofiglio, L. Muscariello, D. Rossi, and S. Valenti, "The quest for ledbat fairness," in *GLOBECOM*, 2010.

[14] D. Ros and M. Welzl, "Assessing ledbat's delay impact," *IEEE Communications Letters*, 2013.

[15] Y. Gong, D. Rossi, C. Testa, S. Valenti, and M. D. Täht, "Fighting the bufferbloat: On the coexistence of aqm and low priority congestion control," *Computer Networks*, 2014.

[16] A. Flickinger, C. Klatsky, A. Ledesma, J. Livingood, and S. Ozer, "Improving latency with active queue management (aqm) during covid-19," 2022.

[17] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, 1989.

[18] S. Floyd, "Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic," *ACM SIGCOMM Computer Communication Review*, 1991.

[19] T. R. Henderson, E. Sahouria, S. McCanne, and R. H. Katz, "On improving the fairness of tcp congestion avoidance," in *IEEE GLOBECOM*, 1998.

[20] S. Floyd, "Highspeed tcp for large congestion windows," https://datatracker.ietf.org/doc/html/rfc3649, 2003.

[21] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," *IEEE/ACM TRANSACTIONS ON NETWORKING*, 2003.

[22] C. L. T. Man, G. Hasegawa, and M. Murata, "Imtcp: Tcp with an inline measurement mechanism for available bandwidth," *Computer Communications*, 2006.

[23] T. Anderson, A. Collins, A. Krishnamurthy, and J. Zahorjan, "Pcp: Efficient endpoint congestion control," in *3rd Symposium on Networked Systems Design and Implementation*, 2006.

[24] D. A. Hayes and G. Armitage, "Revisiting tcp congestion control using delay gradients," in *IFIP/TC6 NETWORKING 2011*, 2011.

[25] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "Timely: Rtt-based congestion control for the datacenter," in *SIGCOMM*, 2015.

[26] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "Pcc vivace: Online-learning congestion control," in *the 15th USENIX Symposium on Networked Systems Design and Implementation*, 2018.

[27] T. Meng, N. R. Schiff, P. B. Godfrey, and M. Schapira, "Pcc proteus: Scavenger transport and beyond," in *SIGCOMM*, 2020.

[28] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS Operating Systems Review*, 2008.

[29] M. Hock, F. Neumeister, M. Zitterbart, and R. Bless, "Tcp lola: Congestion control for low latencies and high throughput," in *IEEE 42nd Conference on Local Computer Networks*, 2017.

[30] H. Adhari, T. Dreibholz, S. Werner, and E. P. Rathgeb, "Eclipse: A new dynamic delay-based congestion control algorithm for background traffic," in *18th International Conference on Network-Based Information Systems*, 2015.

[31] T. Tsugawa, G. Hasegawa, and M. Murata, "Background tcp data transfer with inline network measurement," *IEICE Transactions on Communications*, 2006.

[32] H. Shimonishi, T. Hama, M. Y. Sanadidi, M. Gerla, and T. Murase, "Tcp westwood low-priority for overlay qos mechanism," *IEICE Transactions on Communications*, 2006.

[33] S. Q. V. Trang, E. Lochin, C. Baudoin, E. Dubois, and P. Gelard, "Flower – fuzzy lower-than-best-effort transport protocol," in *40th Annual IEEE Conference on Local Computer Networks*, 2015.

[34] D. A. Hayes, D. Ros, A. Petlund, and I. Ahmed, "A framework for less than best effort congestion control with soft deadlines," in *IFIP Networking Conference and Workshops*, 2017.

[35] G. Carofiglio, L. Muscariello, D. Rossi, C. Testa, and S. Valenti, "Rethinking the low extra delay background transport (ledbat) protocol," *Computer Networks*, 2013.

[36] H. Theil, "A rank-invariant method of linear and polynomial regression analysis. i, ii, iii," in *Koninklijke Nederlandse Akademie Wetenschappen*, 1950.

[37] P. K. Sen, "Estimates of the regression coefficient based on kendall's tau," *Journal of the American Statistical Association*, 1968.

[38] R. K. Jain, D. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *ACM Transaction on Computer Systems*, 1984.

[39] S. Valenti, "Source code of ledbat conforming to draft version 00," https://github.com/silviov/TCP-LEDBAT.

[40] D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "Ledbat: the new bittorrent congestion control protocol," in *Proceedings of 19th International Conference on Computer Communications and Networks*, 2010.

[41] S. Valenti and Q. Li, "Source code of ledbat conforming to rfc6817," https://github.com/tinalee77/LEDBAT.

[42] Q. Li, "Source code of ledbat++ conforming to draft version 01," https://github.com/tinalee77/LEDBAT-Plus-Plus.

[43] "Common open research emulator," https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/CORE/.

[44] J. Ahrenholz, "Comparison of core network emulation platforms," in *The 2010 Millitary Communications Conference*, 2010.

[45] "Multi-generator (mgen) network test tool," https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/MGEN/.

[46] V. Paxson and S. Floyd, "Wide area traffic: The failure of poisson modeling," *IEEE/ACM TRANSACTIONS ON NETWORKING*, 1995.

[47] "Mawi working group traffic archive," https://mawi.wide.ad.jp/mawi/.

[48] L. Andrew, C. Marcondes, S. Floyd, L. Dunn, R. Guillier, W. Gang, L. Eggert, S. Ha, and I. Rhee, "Towards a common tcp evaluation suite," in *International Workshop on Protocols for Fast Long-Distance Networks*, 2008.

[49] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional integral controller enhanced (pie): A lightweight control scheme to address the bufferbloat problem (rfc8033)," https://datatracker.ietf.org/doc/html/rfc8033, 2017.

[50] D. Hayes, D. Ros, L. Andrew, and S. Floydd, "Common tcp evaluation suite," https://datatracker.ietf.org/doc/html/draft-irtf-iccrg-tcpeval-01, 2014.

Qian Li received the B.S. degree in Computer Information Management from HeBei university, HeBei, China, in 2001, and the M.S. degree in Computer Science from Uppsala University, Uppsala, Sweden, in 2010. She is currently working toward a Ph.D. degree offered by the University of Oslo. Her research interests include congestion control algorithms, routing protocols, and intelligent environments.