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

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

27-01-2022 / 10-02-2022

CITATION

Li, Qian (2022): TCP FlexiS: A New Approach To Incipient Congestion Detection and Control. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.19077161.v1>

DOI

[10.36227/techrxiv.19077161.v1](https://doi.org/10.36227/techrxiv.19077161.v1)

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

Qian Li

Abstract—Bulk data transfer applications are greedy in nature in that they have virtually unlimited bandwidth requirements. If serviced by a best effort congestion control, they can create periodic packet loss, long delay and throughput degradation of cross traffic, which may disrupt the normal operation of concurrent QoS sensitive applications. A Lower than Best Effort (LBE) congestion control can make greedy applications only utilize bandwidth left by QoS sensitive applications, thus improve QoS. We propose FlexiS – a Flexible Sender side LBE congestion control mechanism. Unlike most existing LBE congestion controls, which use base delay in incipient congestion detection, FlexiS employs a trend of delay based congestion detector that is more robust to dynamic network environments, in which base delay is difficult to estimate. A non-linear increase and multiplicative decrease controller is used to adapt sending rate by FlexiS. Extensive emulation and Internet tests showed that: (1) FlexiS generally has much lower impact on coexisting best effort traffic than LEDBAT; (2) FlexiS is scalable across a wide range of available bandwidth; (3) aggregate FlexiS traffic can effectively utilize available bandwidth; (4) in most situations, FlexiS flows with the same or different RTTs can fairly share available bandwidth – achieving a fairness index greater than 0.99; (5) the performance of FlexiS is almost not affected by route change and active queue management.

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

I. INTRODUCTION

BULK data transfer applications such as client-to-cloud backup, software update, inter-data-center synchronization and peer-to-peer file sharing have virtually unlimited bandwidth requirements. If these greedy applications are serviced by a loss based Best Effort (BE) Congestion Control (CC), they can create periodic packet loss and long delay, which can disrupt the normal operation of delay and loss sensitive applications like live streaming video and Voice over IP. Furthermore, the throughput of bandwidth sensitive applications such as Video on Demand streaming might be reduced below their critical operation level due to the fair competition of the greedy flows generated by bulk data transfer applications.

A promising way of reducing disruption brought by greedy applications is to service them with Lower than Best Effort (LBE) CCs, which seek to utilize bandwidth left by QoS sensitive applications with which it shares a bottleneck link. In this paper, the data flows controlled by LBE CCs are referred to as low priority or LBE flows, and those governed by BE CCs are termed high priority or simply BE flows. An LBE CC can detect and respond to congestion earlier than a BE CC, and thus yields bandwidth to concurrent BE flows whose

bandwidth demand is growing. On the other hand, an LBE CC probes for available bandwidth repeatedly and consequently can absorb bandwidth newly released by high priority flows whose bandwidth demand is decreasing.

In this paper, we propose a novel LBE CC FlexiS – a Flexible Sender side LBE CC. The objectives of FlexiS are: (1) low intrusion on coexisting high priority flows; (2) high bandwidth utilization; (3) fairly sharing spare bandwidth between FlexiS flows. The objectives are listed in order of priority.

Most existing LBE CCs [1], [2], [3] and delay based CCs [4], [5], [6], [7] 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. It can change with the change of route between the source and destination of a flow. Furthermore, a flow might not be able to discover the true base delay if the bottleneck is persistently overloaded during its lifetime. A wrong estimation of base delay can make an LBE CC fail to meet its design objectives. For instance, when no cross traffic is running in the network, LEDBAT was shown to be unfair due to the failure in estimating base delay by late coming flows [8] and to induce increasing queuing delay until buffer overflow due to wrong base delay estimation [9].

We had been searching for an alternative to base delay based incipient congestion detection. Among all techniques evaluated, a trend of delay based congestion detector was chosen due to many of its merits. For example, it can detect congestion within very short period of time (in the order of milliseconds). Experiments showed that it can respond to congestion before an Active Queue Management (AQM) algorithm drops/marks packets in most cases. Due to the quick congestion response, FlexiS is much less intrusive to high priority flows than LEDBAT, especially when multiple LBE flows running simultaneously. Furthermore, the accuracy of the detector was shown not to be affected by route change.

Conventional LBE CCs [2] [1] [3] [10] use linear controls for rate adaptation. The merit of linear control is simplicity and possibility of converging to fairness. It is proven by Chiu and Jain in [11] 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 Round Trip Time (RTT) and reduce it by half. However, this AIMD application proved biased against long RTT flows. Floyd proposed in [12] 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 [13]. In addition to the fairness problem, AIMD used by standard TCP is also known to have low bandwidth utilization in high-speed long-latency networks.

We evaluated a variety of rate increase/decrease functions. Finally, a non-linear increase multiplicative decrease rate controller was chosen because it provides the best trade-off between intrusion and utilization. To make this more concrete, a FlexiS sender increases sending rate with increasing acceleration and reduces rate by a fixed percentage. The amount of rate increase per RTT is in proportion to a connection's RTT and time elapsed since the start of the current rate increase epoch. FlexiS does not have a slow start phase, the same rate increase function is applied whenever rate should be increased. Our emulation experiments showed that this rate controller can realize high intra- and inter-RTT fairness in most cases. It makes FlexiS scalable across a wide range of available bandwidth. Compared to the slow start and congestion avoidance rate increase model, it can avoid the initial long delay and packet loss caused by slow start and can quickly absorb available bandwidth not only at the beginning of a connection but also in later stages, which is a desirable property of an LBE CC that needs to cope with greatly oscillating available bandwidth.

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

II. RELATED WORK

In this section, we review previous work that either inspired FlexiS or have similar objectives with FlexiS.

A. Work that inspired FlexiS

Pathload [14] is an available bandwidth 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 of 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 assumed higher than the available bandwidth, otherwise lower. Available bandwidth can be discovered by sending out multiple fleets with different rates. Inspired by pathload, FlexiS uses increasing trend of RTT as indicator of incipient congestion.

B. Less than Best Effort Congestion Controls

TCP nice [1] is a TCP Vegas-based LBE CC. It adds a new incipient congestion detector in addition to that of Vegas. At the end of every RTT, if more than a fraction (50% by default) of Queuing Delay (QD) measurements obtained from last RTT are greater than a threshold (by default 20% of the estimated maximum QD), TCP nice assumes that congestion is forming and halves cwnd. Otherwise, the congestion avoidance algorithm of TCP Vegas is used.

TCP-LP [2] detects early congestion using smoothed QD exceeding a certain percentage (by default 15%) of maximum QD. When incipient congestion is detected, TCP-LP halves its cwnd, then enters an inference phase, during which, cwnd is unchanged. If the sender detects another congestion in the inference phase, cwnd is reduced to 1 MSS. When no congestion is detected, cwnd is increased by 1 MSS per RTT.

TCP Westwood Low Priority [15] is an adaptation of TCP Westwood. Self-induced backlog exceeding a threshold is used as an indication of incipient congestion. Backlog is estimated as $cwnd - BWE \times RTT_{min}$, where BWE is the bandwidth share estimated. Cwnd is additively increased when no congestion is detected and is reduced to $BWS \times RTT_{min}$ on congestion.

ImTCP-bg [16] uses the available bandwidth A estimated by ImTCP [17] to calculate an upper limit $maxcwnd$ for cwnd. $maxcwnd = \bar{A} \times RTT_{min}$, where $\bar{A}$ is the exponentially weighted moving average of A . If the smoothed RTT $\overline{RTT}$ and the observed minimum RTT RTT_{min} satisfies $\overline{RTT}/RTT_{min} > \delta$, $\delta > 1$, cwnd is reduced to $cwnd \times RTT_{min}/\overline{RTT}$.

Low extra delay background transport (LEDBAT) [3] uses QD exceeding a target $TARGET$ (100 ms by default) as an indication of incipient congestion. On the receipt of an ack, cwnd is increased additively if QD is below $TARGET$ and decreased additively if QD exceeds $TARGET$. To be more specific, cwnd is updated with the following equations. $cwnd = cwnd + GAIN \times off_target \times bytes_newly_acked \times MSS/cwnd$, $off_target = (TARGET - queuing_delay)/TARGET$ and $queuing_delay = current_delay - base_delay$. 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.

fLEDBAT [18] modifies the cwnd update function of LEDBAT. It proposes to use constant increase to replace the original proportional increase to speed up convergence and improve efficiency. The original additive decrease is attributed to as the cause of unfairness and is replaced by multiplicative decrease.

FLOWER [19] replaces LEDBAT's linear controller with a fuzzy controller. The fuzzy controller takes as input current QD, maximum QD, QD error and change in the error and outputs the change to be added to the current cwnd.

LEDBAT++ [10] improves LEDBAT using a number of techniques. It replaces OWD with RTT in QD calculation. Slow start quits when QD exceeds 3/4 target QD to reduce initial intrusion. The linear controller is replaced by additive increase multiplicative decrease control and the pre-configured increase GAIN is replaced by a variable which is in proportion to a flow's RTT. LEDBAT++ periodically reduces cwnd to 2 packets in order to discover a true base RTT.

Spring et al. proposed a receiver side LBE CC policy [20] that aims to prioritize incoming TCP connections during congestion by controlling each connection's receive buffer size (consequently affecting receiver's advertised window). It is shown that when the last mile is the bottleneck, limiting the size of the receive buffer of long lived bulk transfer

connections during congestion can improve the performance of higher priority connections.

Mehra et al. proposed a receiver side LBE CC [21] that regulates an incoming TCP connection's bandwidth share to a target rate calculated based on priority preferences given by users. The actual throughput of a connection is controlled by adjusting the advertised window and the delay in sending ack.

Hayes et al. proposed a deadline aware LBE framework [22] that is capable of making any non-LBE CCs to have LBE behavior and the degree of LBEness can be changed according to remaining time to deadline.

Sync-TCP [23] and PERT [24] emulate active queue management algorithms from end hosts with TCP CCs. Although they were not designed to be LBE, they exhibit LBE behavior when they coexist with loss-based BE CCs.

Some delay based CCs such as CARD [4], TRI-S [5], DUAL [6], TCP-Vegas [7], CIM [25] and TCP-BFA [26] aim to avoid creating long delays and packets losses. They also manifest signs of LBE.

C. Non-Congestion Control LBE Services

A Lower Effort (LE) Per Hop Behavior (PHB) [27] for differentiated service was standardized by IETF. The specification specifies that ideally, LE packets should be forwarded only when no packet with any other PHB is awaiting transmission. A network scheduler [28] that is capable of scheduling background traffic to use idle bandwidth was proposed as a component of an operating system. Key et al. [29] emulated an LBE transport on the application layer. A more exhaustive survey on end-to-end LBE services was conducted by Ros et al. [9].

III. DESIGN

A. Overview

Trend of delay is used to detect incipient congestion by a FlexiS sender. An increasing trend of RTT signals congestion. Rate is decreased by a fixed percentage when congestion is detected, but otherwise is increased according to a cubic function of elapsed time. To be specific, on the receipt of each acknowledgment (ack), an RTT sample and a timestamp are obtained and are put into an observation window O. The timestamp is the sending time (in ms) of the acknowledged data packet. If enough RTT samples are gathered in O, the trend of these RTTs is derived and the oldest sample is removed from O. An increasing trend indicates congestion and cwnd will be therefore decreased. However, if the trend is non-increasing, cwnd will be increased. If the decision is increase, rate is updated using a cubic function of elapsed time. FlexiS does not have a slow start phase, rate is always increased with the same function. Rate is reduced by a fixed percentage when incipient congestion is detected. Algorithm 1 shows a pseudo-code of the main logic of FlexiS.

From a high level view, a FlexiS sender iteratively goes through three phases: pending, increase and decrease. In the pending phase, the sender does not have sufficient RTT samples to make a trend analysis, it puts every newly received RTT sample into O and keep cwnd unchanged. A newly established

Algorithm 1 The main logic of FlexiS

Variables

d_i : RTT measured by the i_{th} ack
 t_i : timestamp associated with d_i , in ms
O: data structure of observation window
 L_o : length of O in ms
 n : length of O in number of RTT samples
 τ : the minimum duration (in ms) required to make a trend analysis

On the receipt of the i_{th} ack

```

put  $(t_i, d_i)$  into O
 $L_o \leftarrow t_i - t_1$ 
if  $L_o < \tau$  then
    cwnd is unchanged
else
    if  $n < 3$  then
        increase cwnd
    else
        derive the trend of the RTT samples in O
        if trend is increasing then
            decrease cwnd
        else
            increase cwnd
    end if
end if
remove the oldest sample  $(t_1, d_1)$  from O
end if

```

connection always starts from the pending phase. As soon as $L_o \geq \tau$, the pending phase terminates and the sender goes into either the increase or decrease phase based on the trend of RTT. If the trend is non-increase, the sender enters the increase phase, in which cwnd is increased. An increase phase is also termed an increase epoch, which terminates when congestion is detected. The sender will then go into the decrease phase, in which, rate is decreased and O is emptied. Due to the need of replenishing O, a decrease phase is always followed by a pending phase.

Next we will dig deeper and elaborate on the building blocks of FlexiS.

B. Congestion Detection

Congestion is the state of sustained overload of a bottleneck. During congestion, a BE CC will contend for bandwidth until packet loss occur. An LBE CC should however never compete against BE CCs during congestion. Detecting congestion prior to packet loss thus becomes a major objective of an LBE CC. FlexiS uses increasing trend of RTT over a period of time as indication of incipient congestion. The rationale behind this is that loss is usually preceded by the increase of bottleneck queue and as the queue grows, RTT will exhibit an increasing trend. However, not every queue is a result of congestion. Some queues are caused by traffic bursts. Such queues are transient and should not be considered as an indication of congestion. Although it is difficult to distinguish between these two types of queues only by trend of RTT, a careful selection

of the parameter τ can greatly improve the performance of the congestion detector.

Trend of RTT can be derived using the slope of the regression line of the RTT samples in the observation window O . We use Theil-Sen estimator [30] [31] to estimate the slope. Theil-Sen slope of RTT samples in O is the median of slopes of lines connecting all distinctive pairs of points in O . Strictly speaking, let d_i be the RTT measured by the i_{th} ack, t_i be the timestamp associated with d_i , O be the observation window, n be the number of RTT samples in O , S_{ij} be the slope of the line connecting points (t_i, d_i) and (t_j, d_j) and S be the Theil-Sen slope. We have $O = ((t_i, d_i); 1 \leq i \leq n)$, $S_{ij} = (d_j - d_i)/(t_j - t_i)$, $t_i < t_j$, and $S = \text{median}(S_{ij})$. 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 a threshold θ , the RTT samples are deemed to have an increasing trend, and the sender will back off.

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

C. Increase and Decrease Functions

Sending rate r is increased according to a cubic function of time elapsed t since the start of the current increase epoch E . Equation 1 defines r as a function of t .

$$\begin{aligned} r &= \frac{t^3}{\alpha} + \frac{t}{\beta} + r_0 \\ t &= t_{cur} - t_0 \\ r_0 &= \frac{w_0}{d_{min}} \end{aligned} \quad (1)$$

, where α and β are increase factors. They decide the shape of the rate curve. t_0 is the start time of E and t_{cur} is the current time. w_0 and r_0 are the cwnd value and sending rate at t_0 , respectively. d_{min} is the minimum RTT observed since last cwnd reduction or since the establishment of the connection if no reduction has been done. It is updated on the receipt of each ack. t_0 , w_0 and r_0 are recalculated at the beginning of every increase epoch.

Cwnd w can be calculated from rate r by using equation 2.

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

Equation 3 is used to decrease cwnd when incipient congestion is detected.

$$w' = w \times \gamma \quad (3)$$

, where w' and w are cwnd values after and before reduction, $0 < \gamma < 1$ is the decrease factor.

Cwnd is halved on normal packet loss and is reduced to 1 MSS when retransmission timer goes off. After any type of cwnd reduction, the observation window O is emptied and d_{min} is set to positive infinity.

D. Pacing

Pacing is a technique used to evenly space packets at specified intervals. By the use of pacing, a FlexiS flow can avoid creating queues with its own packets at bottlenecks when its rate is no more than spare bandwidth. That makes queues the only indicator of congestion or the presence of non-FlexiS flows. In Linux, per flow pacing can be realized by TCP stack or by the fair queue packet scheduler. In either way, we need to determine a desired pacing rate. We use the rate in one round trip time as 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 P , which is used to calculate pacing rate by the kernel. P is the ratio between the current cwnd w and the cwnd in one RTT w' . It is updated according to equation 4 whenever cwnd is modified.

$$P = \begin{cases} \frac{w'}{w} \times 100 = \frac{r'}{r} \times 100 = \\ \frac{(\frac{t}{\alpha})^3 + \frac{t}{\beta} + r_0}{(\frac{t+d_{min}}{\alpha})^3 + \frac{t+d_{min}}{\beta} + r_0} \times 100, & \text{if increased} \\ \frac{w'}{w} \times 100 = 100, & \text{Otherwise} \end{cases} \quad (4)$$

, where r is the current sending rate and r' is the rate in one RTT. If cwnd was increased, P is updated with the first sub-equation. If cwnd was unchanged or just decreased, the cwnd value in one RTT would be equal to the current cwnd, so P is 100%.

E. Parameter Selection

We briefly mentioned some protocol parameters in the previous part of this section, next we discuss how to set these parameters.

α and β are increase factors. They together determine the shape of the increase curve. Smaller values result in faster increase and larger values slow down rate increase. β determines the initial increase speed of an increase epoch. It mainly affects queuing delay and bandwidth utilization. A large value gives the bottleneck enough time to drain the queued packets and a small value can improve bandwidth utilization. α governs the speed of later rate increase. It mostly affects intrusion and bandwidth utilization. A large α can reduce intrusion and utilization. Smaller values have the opposite effects. According to heuristics, the values of α and β (when t is in ms) that can make a good trade-off between intrusion and utilization are 100 and 10, respectively.

γ is the decrease factor. It determines how much cwnd will be retained on cwnd reduction. Because a FlexiS connection backs off when its rate is only slightly over spare bandwidth, cwnd does not need to be reduced by a large percent. Setting γ to 0.85 has been shown to yield an overall good performance.

τ and θ together control when to back off. Large values can delay the response to growing queues, which will increase intrusion if congestion is forming. However if the queue is transient, large values can improve utilization without causing too much intrusion. In contrast, smaller values result in quicker response to increasing queues, which can therefore reduce intrusion and utilization. With experiments, we evaluated all combinations of τ and θ with $\tau \in [10, 100]$, $\theta \in [10, 100]$ and $step = 10$. Finally, $\tau = 60$ ms and $\theta = 30$ were shown to have the best overall performance.

IV. EVALUATION

This section presents evaluation results. First, the definitions of performance metrics are given. Then, we briefly describe the implementation of LEDBAT, against which FlexiS was compared. Finally the results of emulation and Internet tests are presented.

A. Performance Metrics

Convergence time t is defined in equation 5 as the time taken for an LBE connection to increase its cwnd to its Bandwidth Delay Product (BDP) for the first time since connection establishment. BDP of an LBE connection is defined as the product of available bandwidth and base delay.

$$t = t_c - t_s \quad (5)$$

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

Bandwidth utilization u measures how much spare bandwidth is utilized by an LBE connection. It is defined in equation 6 as the ratio between throughput of the LBE flow and the available bandwidth.

$$u = \frac{t_l}{b_a} \times 100 \quad (6)$$

$$b_a = C - b_u$$

, where b_a is the average available bandwidth, C is the bottleneck link capacity and b_u is the average bandwidth consumed by high priority flows. b_u is measured by bmon [32] at the bottleneck Network Interface Card (NIC) with one-second reading intervals. t_l is the average throughput of LBE flows. A utilization greater than 100% indicates over-use of spare bandwidth, which implies that the LBE flows are "stealing" bandwidth from high priority flows.

Throughput degradation e measures how much throughput of the high priority flows is lost due to the contention of low priority flows. It is defined in equation 7.

$$e = \frac{T_o - T_w}{T_o} \times 100 \quad (7)$$

, where T_o is the average throughput of BE flows when they run without any concurrent LBE flows and T_w is the throughput of the BE flows when they share bottlenecks with LBE flows.

Queuing delay (QD) measures how much time a packet spends waiting in bottleneck queues. Let q_i be the QD measured by packet i . Then $q_i = d_i - d_{base}$, where d_{base} is the RTT of the unloaded route and d_i is the RTT measured by packet i . Because numerous QD samples can be obtained during a test, the 90_{th} percentile is used to indicate the overall degree of QD. It is the value which 90% of QD measurements are smaller than or equal to.

Retransmission rate r is defined in equation 8. It measures how many packets are retransmitted for every 1000 packets sent. It is used as an estimate of loss rate.

$$r = \frac{p_r}{p_s} \times 1000 \quad (8)$$

, where p_r is the total number of packets retransmitted and p_s is the total number of packets sent during a test.

Jain's fairness index [33] f is used to measure fairness between LBE connections. It is defined in equation 9.

$$f = \frac{(\sum_{i=1}^n t_i)^2}{n \sum_{i=1}^n t_i^2} \quad (9)$$

, where t_i is the average throughput of the i_{th} LBE connection, n is the number of LBE connections sharing the same bottleneck. f is between 0 and 1. The larger the value the higher the fairness.

B. Implementation and Configuration of LEDBAT

We have adapted the LEDBAT Linux kernel module implemented by Silvio Valenti et. al. [34] to conform to RFC6817. In our implementation, one way delay is estimated only at the sender by subtracting send timestamp from receive timestamp. The timestamps need to be converted into the same unit before subtraction, because the sender and receiver can use different clock frequencies to generate the timestamps. A frequency estimator is used to estimate remote host's timestamp generation frequency. Because its accuracy is seriously affected by congestion on the data path, it was disabled for all emulation tests, in which all emulated nodes have the same clock frequency. It was only turned on for Internet tests.

In all tests, the TARGET of LEDBAT was set to 100 ms, BASE_HISTORY was set to 10 and noise filter was set to NULL. The same GAIN was used for both cwnd increase and decrease and was set 1. For all emulation tests, two versions of LEDBAT were tested – LEDBAT without slow start (denoted as LEDBAT-BA) and LEDBAT with slow start (denoted as LEDBAT-SS). For Internet tests, slow start and remote timestamp clock frequency estimation were enabled and this version of LEDBAT is referred to as LEDBAT-HZ.

C. Emulation Tests

Emulation tests were conducted on a PC with Ubuntu desktop 20.08 and Linux kernel v4.5 installed. Nine groups of tests were run to evaluate FlexiS from various aspects. All tests were conducted on virtual networks emulated by the Common Open Research Emulator (CORE) [35], which runs in real time and provides an environment for running real applications and protocols. The emulated routers used OSPF as routing protocol. All links were symmetrical, i.e., they had the same bandwidth and delay in both directions. BE cross traffic was generated by the Multi-Generator (MGEN) [36], which is capable of generating real-time traffic patterns and cloning binary trace files captured using tcpdump. The MAWI traffic archive [37] hosts a huge number of Internet trace files captured at various sample points of the WIDE backbone since 1999. We selected several trace files captured at sample point F to load the virtual networks. The chosen traces have diversified rates and burstiness. TCP cubic was the default CC for all BE flows. Low priority flows were generated by a client/server based application written in c. The client sends packets filled with random bytes at its maximum speed and the server silently discards all received packets. Low priority flows were controlled by FlexiS, LEDBAT-BA and LEDBAT-SS in turn in separate tests and the performance of the three LBE CCs were compared. Unless otherwise noted, all performance measurements started from the start of the first LBE flow and ended $300 + I$ seconds after the start of the last LBE flow. I is the starting interval of LBE flows when multiple LBE flows are investigated, and it is otherwise 0. Each test was repeated for 10 times and the results are the averages. Adequate intervals were inserted between tests in order to let the packets of the previous test leave the network.

The default buffer sizes of the emulated nodes were insufficient for large BDP flows. They were adjusted so that they did not become the limiting factor of sending rate. The expanded buffers were the core read and write buffers, TCP read and write buffers, device input buffers and netem delay buffers. Large send and receive offload were disabled. `tcp_no_metrics_save` was enabled so that a new test was not affected by old ones.

In the following text, a link in the emulated network is denoted by the two nodes at both ends of the link as (N_1, N_2) . A path is denoted by all nodes on the path. For example, a path consisting of four nodes can be denoted as (N_1, N_2, N_3, N_4) . A flow is depicted by its source S and destination D connected by an arrow as $S \rightarrow D$.

1) *Single Bottleneck*: This section studies the scenario when there is only one bottleneck in the network. The tests were conducted using a dumbbell topology (Fig. 1), which is adapted from the one originally proposed in [38].

The bandwidth of all peripheral links of the dumbbell topology are 1 Gbps. The bandwidth of the central link (R1,R2) varied from one test to another. The central link is the bottleneck. The OWD of a link is annotated in the graph by the link. The default router queue is FIFO in unit of packets. The bottleneck buffer defaults to 1.5 BDP of a 100 ms flow, which equals a maximum of 150 ms QD (chosen to be larger

Fig. 1. Dumbbell topology adapted from [38]

Fig. 2. Results of scalability tests

than LEDBAT's TARGET). The default direction for LBE data packets is from west to east as depicted in the figure.

a) *Scalability Tests*: The goal of the tests in this group is to examine how well a FlexiS connection adapts to various available bandwidth. The capacity of the bottleneck link was set to 1, 5, 10, 50, 100, 500, and 1000 Mbps. The bottleneck buffer was set to 1.5 BDP of a 10 ms flow. The network was not loaded with any BE traffic, so the bottleneck capacity was just the available bandwidth. For each bottleneck capacity under investigation, three tests were run. In each test, an LBE flow, which took path (H3,R1,R2,H6), using one of the aforementioned LBE CCs was examined. The LBE flow started at time 0 and ended at 100 seconds after convergence. The performance of the LBE CCs are shown in Fig. 2.

The convergence time of the LBE CCs increase with the increase of available bandwidth. Compared to LEDBAT-SS, FlexiS spends more time ramping up when available bandwidth is large. LEDBAT-BA is the slowest in acquiring large amounts of bandwidth and what is worse is that its convergence time increases with the increase of its RTT (not shown in the figure). Bandwidth utilization, QD and retransmission rate were measured after convergence. LEDBAT-BA and LEDBAT-SS have similar performance when these metrics are applied. They have low utilization, long QD and high retransmission rate when available bandwidth is 1 Mbps but quite good

Fig. 3. Responsive tests. RTT of the LBE connection: 10 ms. In Fig. 3c and 3d, solid triangles connected by line segments are the 90_{th} percentile QDs and retransmission rates measured by the on/off flow when it was running alone. The rest of data points are the same metrics measured by the on/off flow when it was running simultaneously with one of the LBE CCs – FlexiS (asterisk), LEDBAT-BA (solid square) and LEDBAT-SS (solid circle).

performance when there are more available bandwidth. FlexiS has pretty consistent performance for all available bandwidth and induce less QD and packet loss than LEDBAT-BA and LEDBAT-SS.

b) Responsiveness Tests: The goal of this group of tests is to examine how quickly a FlexiS connection responds to changes in available bandwidth. Bottleneck link capacity was set to 100 Mbps. A BE on/off flow generated by Mgen was used to alter the available bandwidth. Its sending rate was 75 Mbps during the “on” period and 0 Mbps during the “off” period. The duration for the “on” and “off” periods were the same, which were set to 0.001, 0.01, 0.1, 1, 10 and 100 seconds. For a specific on/off duration D, four tests were run. In the first test, an on/off flow with duration D ran alone without any concurrent LBE flows in the network. In the rest of tests, an on/off flow with the same on/off duration D ran simultaneously with an LBE flow that used one of the LBE CCs. The on/off flow was started first and the LBE flow was started 10 seconds later. The BE flow always used the path (H4,R1,R2,H7) and the LBE flow used one of the paths (H3,R1,R2,H6), (H4,R1,R2,H7) and (H5,R1,R2,H8). The results are shown in Fig. 3, 4 and 5.

When the on/off duration of the BE flow are 0.1 and 1 second, a FlexiS flow has low bandwidth utilization irrespective of its RTT. This is probably because these duration are long enough to make the FlexiS sender to detect the BE flow when the latter is turned “on” and in the meantime they are also short enough such that the BE flow is unable to utilize the bandwidth released by the FlexiS flow. The utilization of LEDBAT-BA and LEDBAT-SS are affected by flow RTT, the use of slow start and on/off duration. When on/off duration is 1 and 10 seconds, long RTT ($> 10ms$) LEDBAT-BA and LEDBAT-SS flows have low utilization. The use of slow start

Fig. 4. Responsive tests. RTT of the LBE connection: 100 ms.

Fig. 5. Responsive tests. RTT of the LBE connection: 204 ms.

can improve utilization. The throughput and retransmission rate of the on/off flow is almost not affected by FlexiS in all tests. In contrast, LEDBAT-SS takes away more bandwidth from the BE flow when on/off duration is long (≥ 1 sec). Retransmission rate of the on/off flow is not seriously affected by LEDBAT-BA and LEDBAT-SS. A FlexiS flow induces little extra QD except when the on/off duration is 100 seconds. In contrast, LEDBAT-BA and LEDBAT-SS flows induce higher extra QD than a FlexiS flow in most tests.

c) Data Path Load Tests: This group of tests were carried out to study how FlexiS performs with varying level of traffic on its data path. The bottleneck link capacity was set to 100 Mbps. The BE cross traffic was cloned from the traces listed in TABLE I. Date of Cap. specifies when the trace was captured. Dir. indicates the direction of the trace from the point of view of sample point F. Avg. Rate is the average rate (in Mbps) of the trace. And Stddev is the standard deviation of the rate. These uni-directional traces were extracted from the original

TABLE I
TRACES USED AS DATA PATH LOAD

Name	Date of Cap.	Dir.	Avg. Rate	Stddev
wide11	2007.01.01	upstream	11.19	1.80
wide33	2008.01.03	downstream	33.64	5.59
wide52	2009.09.03	downstream	52.47	7.33
wide74	2010.03.18	upstream	74.42	10.81
wide94	2009.05.12	upstream	94.48	6.07
wide167	2010.09.30	downstream	167.60	15.57

bi-directional traces provided by MAWI. Each trace file in the table was further split into nine sub-traces, each of which was used to load one of the nine west to east or east to west paths. Flows having the same source and destination IP addresses were put into one sub-trace. At the bottleneck, all nine data or ack direction flows were synthesized to form a single traffic that had great similarity to the original unidirectional traffic. Appendix A justifies, with experiments, the efficacy of this way of synthesis. Unless otherwise noted, BE cross traffic mentioned hereafter were generated in the same way.

The LBE data direction paths were loaded with one of the traces listed in TABLE I. The reverse direction paths were always loaded with wide11. As a result, the data path bottleneck was roughly 10%, 30%, 50%, 70%, 90% and 170% loaded and the ack direction bottleneck was always about 10% loaded. Two sets of tests were run. The first set of tests examined the performance of one ($n = 1$) LBE flow that took path (H4,R1,R2,H7). The second set of tests investigated the performance of nine ($n = 9$) concurrent LBE flows taking paths (H3,R1,R2,H6), (H4,R1,R2,H7) and (H5,R1,R2,H8). There were three LBE flows on each path. For each data path load and n combination, four tests were run. In the first test, the BE flows ran alone without any concurrent LBE flows. In the rest of tests, they ran simultaneously with n LBE flows. BE flows were started first at once. LBE flows were started 60 seconds later at 10-second intervals. Fig. 6a, 6c, 6e and 6g illustrate the performance of the LBE CCs in the first set of tests. Fig. 6b, 6d, 6f and 6h show the results of the second set of tests.

The bandwidth utilization of a single FlexiS flow is affected by the load on the data path. Its utilization declines drastically when load exceeds 50%. This is the result of the frequent packet buildup in bottleneck buffer during high load. In Fig. 6b, we can see that FlexiS's bandwidth utilization is greatly improved by increasing the number of concurrent flows. A single FlexiS flow has negligible impact on BE traffic and increasing the number of flows does not significantly increase intrusion. Both LEDBAT-BA and LEDBAT-SS flows have high utilization and high intrusion. When bottleneck load is 90%, nine LEDBAT-SS flows can over-utilize available bandwidth and achieve 600% utilization! In most cases, LEDBAT-BA and LEDBAT-SS have much higher impact on BE flows than FlexiS. In particular, the throughput of BE flows are reduced by approximately 18% and 22% by nine LEDBAT-SS flows when the bottleneck is 90% and 170% loaded, respectively.

Fig. 6. Impact of data path load on the performance of LBE CCs. Pay attention to the different ranges of Y axes used by figures on the left and right hand side columns. Left: performance of one LBE flow H4→H7. Right: performance of nine LBE flows taking three different paths (H3,R1,R2,H6), (H4,R1,R2,H7) and (H5,R1,R2,H8). Each path was shared by three LBE flows.

d) ACK Path Load Tests: This group of tests examine the performance of FlexiS when traffic level on its ack path varies. The bottleneck link capacity was set to 100 Mbps. The data direction paths were always loaded with wide11. The ack direction paths were loaded with one of the traces listed in Table I. As a result, the ack path bottleneck was roughly 10%, 30%, 50%, 70%, 90% and 170% loaded and the data path bottleneck was always about 10% loaded. For each ack path load, four tests were run. In the first test, BE flows ran alone without any concurrent LBE flows. In the rest of tests, the BE flows ran simultaneously with an LBE flow. BE flows were started first at once. Then an LBE flow H4→H7 was started 60 seconds later. The results are shown in Fig. 7

Congestion on ack path greatly affects the ability of FlexiS to effectively utilize the available bandwidth on the data path. This is because FlexiS uses trend of RTT to detect congestion. The utilization of LEDBAT-SS and LEDBAT-BA are also

Fig. 7. The impact of ack path load on the performance of LBE CCs. Only the performance of one LBE flow H4→H7 is shown.

TABLE II
TRACES USED BY RTT TESTS

Name	Date of Cap.	Dir.	Avg. Rate	Stddev
wide42	2020.12.27	downstream	42.99	20.76
wide25	2020.12.27	upstream	25.55	7.58

reduced to 72% and 48%, respectively, when ack path load is 170%. Because the LBE CCs have almost no impact on data direction BE traffic, intrusion to ack direction BE traffic was measured instead. A FlexiS flow has little impact on ack direction BE traffic. In comparison, the reverse direction BE traffic's throughput can be reduced by acks of LEDBAT-BA or LEDBAT-SS flows and their RTTs are increased by queues created by LBE flow on the data path.

e) RTT Tests: The goal of this group of tests is to investigate the impact of flow RTT on the performance of FlexiS. The capacity of the bottleneck link was set to 100 Mbps. The traces used to load the network is listed in Table II. Wide42 was used as the data direction load and wide25 was used as the reverse direction load. The LBE flow under investigation took one of the nine west to east paths. For each RTT d examined, four tests were run. In the first test, BE flows were the only traffic in the network. In the rest of tests, the BE flows ran concurrently with an LBE flow with RTT d . BE flows were started first together. The LBE flow was started 60 seconds later. The results are shown in Fig. 8.

FlexiS has similar bandwidth utilization for all flow RTTs, while the utilization of LEDBAT-BA and LEDBAT-SS flows decrease with the increase of their RTTs. This is because a FlexiS flow increases cwnd in proportion to its RTT but LEDBAT flows with different RTTs increase cwnd by the same amount as long as off-target is the same. Due to the same reasons, large RTT FlexiS flows cause slightly higher intrusion whereas short RTT LEDBAT connections are more intrusive.

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

Fig. 9. The impact of PIE's target on the performance of LBE CCs. The default target of PIE is 15 ms.

f) AQM tests: These tests study how Active Queue Management (AQM) affects the performance of FlexiS. The capacity of the bottleneck link was 100 Mbps. Wide42 and wide25 were used as data and ack direction traffic, respectively. The AQM of the bottleneck router was set to PIE [39]. Limit of PIE was set to 1250 packets, which corresponds to a 150 ms hard limit on queuing delay. Target of PIE was set to 5, 10, 15, 20 and 25 ms. For each target value, four tests were run. In the first test, the BE flows ran alone without any concurrent LBE flows. In the rest of the tests, the BE flows ran with one LBE flow H4→H7. BE flows were started first together. Sixty seconds later, an LBE flow was started. Fig. 9 shows the results.

The utilization of FlexiS roughly doubles that of LEDBAT-BA and LEDBAT-SS for any given target of PIE. This is because FlexiS responds to congestion much earlier than LEDBAT. In the meanwhile, FlexiS also has a little bit higher

Fig. 10. Intra- and Inter-RTT fairness of LBE CCs. Pay attention to the different ranges of Y axes of figures for intra- and inter-RTT tests.

impact on BE traffic than LEDBAT CCs.

g) *Fairness Tests*: This group of tests assess the fairness of FlexiS when the flows have the same or different RTTs.

For intra-RTT tests, the capacity of the bottleneck link was set to 20 Mbps. The bottleneck buffer was set to 4 BDP of a 100 ms flow. Fairness was studied with two sets of tests. The first set of tests studied flow fairness in an unloaded network and the second set of tests studied fairness in a loaded network. In the latter case, wide11 was used as BE traffic in both directions. In both sets of tests, the fairness of three LBE flows with the same CC and RTT were examined. All LBE flows took the path (H4,R1,R2,H7) and were started at 30-second intervals. For the second set of tests, BE flows were started first at once and the first LBE flow was started 60 seconds later. Fairness was measured for 600 seconds from 30 seconds after the start of the last LBE flow. The results are shown in Fig. 10c.

FlexiS flows have high fairness (0.999) in both loaded and unloaded networks. While LEDBAT-BA and LEDBAT-SS flows cannot fairly share available bandwidth when the network is unloaded, which is the result of the so called late comer advantage problem. Loading the network can improve the fairness of LEDBAT flows. The traffic bursts and the resultant packet loss introduced by cross traffic can help LEDBAT detect true base delay.

For inter-RTT tests, fairness was also evaluated with two sets of tests. In the first set of tests, bottleneck link capacity was set to 20 Mbps. Wide11 was used to load the network in both directions. In the second set of tests, the capacity of the bottleneck link was set to 100 Mbps. Wide42 and wide25 were used as data and ack direction loads, respectively. In both set of tests, the bottleneck buffer was set to 4 BDP of a 100 ms flow. The fairness of two LBE flows with the same CC but different RTT was examined. The first flow took the same path (H4,R1,R2,H7) in all tests and the second flow took one of the eight data direction paths that is different from the first flow's. For each RTT combination, three tests were run

Fig. 11. Diamond Topology

with each evaluating a different LBE CC. In each test, the BE flows were started first. Sixty seconds later, the first LBE flow H4→H7 was started. Thirty seconds later, the second LBE flow was started. Fairness was measured for 600 seconds from 30 seconds after the start of the second LBE flow. Fig. 10a and 10b present the results.

When bottleneck capacity is 20 Mbps and available bandwidth is roughly 9 Mbps, FlexiS flows can achieve high fairness (> 0.99) when the second flow's RTT is larger than 32 ms. Fairness declines to 0.92 and 0.86 respectively when the RTT of the second flow are 32 and 10 ms. This is probably because when a flow's BDP drops below a threshold, the accuracy of FlexiS's congestion detection and response is affected. When bottleneck capacity is increased to 100 Mbps and available bandwidth is increased to around 58 Mbps, the corresponding fairness index improved to 0.98 and 0.99 respectively. The fairness of LEDBAT-BA and LEDBAT-SS is mainly affected by the difference between the RTT of two competing flows. The smaller the difference, the higher the fairness.

2) *Route Change Tests*: Internet route between two end hosts may change during the lifetime of a TCP connection. This group of tests examine the robustness of FlexiS over failures such as route change. The topology used is diamond, which is illustrated in Figure 11.

The bottleneck link is (H1,R1), whose capacity is 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 queuing delay is 150 ms. There are two routes between H1 and H2. They are $P1 = (H1, R1, R2, R4, H2)$ and $P2 = (H1, R1, R3, R4, H2)$. The default route is P1.

This group of tests studied a simple route change scenario, when 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. Two sets of tests were run. In the first set of tests, OWD of P2 was 24 ms, which was 16 ms longer than that of P1. In the second set of tests, the OWD of P2 was 124 ms, which was 116 ms longer than that of P1. In both of the tests, the network was not loaded with any BE traffic. Eth1 of R1 stayed in up or down for the same amount of time. Four up/down duration

Fig. 12. Results of route change tests. Pay attention to the different ranges of Y axes used in figures on the left and right hand side columns.

were studied: 10, 40, 160 and 640 seconds. For each up/down duration D , three tests were run with each testing a different LBE CC. In each test, the initial state of eth1 was set to on and it was brought down and then up again during the test. At time zero, an LBE flow $H1 \rightarrow H2$ was started and ran for $3 \times D$ seconds. The results of the first set of tests are shown in Fig. 12a and 12c. Fig 12b and 12d show the results of the second set of tests.

The utilization of FlexiS is almost not affected by the RTT of P2 and only slightly impacted by up/down duration. The utilization of LEDBAT-BA and LEDBAT-SS are affected by both RTT of P2 and up/down duration. When OWD of P2 is 124 ms, the utilization of the two LEDBAT CCs decline dramatically with the decrease of up/down duration. This is because the difference between the OWD of P1 and P2 is greater than LEDBAT's TARGET and LEDBAT did not update base delay timely. There are a few seconds of delay from link status change to route update. Due to this delay, a TCP connection may experience packet loss. That is why the retransmission rate of FlexiS and LEDBAT-BA are non-zero. LEDBAT-SS has comparatively higher retransmission rate due to the use of slow start.

3) *Multiple Bottlenecks Tests*: In a more realistic scenario, a data flow can traverse multiple congested gateways and wait in various queues for transmission. This group of tests investigate the performance of FlexiS in such scenarios. The topology used (Fig. 13) is adapted from the parking lot topology originally proposed in [40]. The capacity of the peripheral links are 1 Gbps and that of the central links are 100 Mbps. The bottlenecks are the central links (R1,R2), (R2,R3) and (R3,R4). 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 flow, which corresponds to a maximum queuing delay of 150 ms.

Two sets of tests were conducted. For tests in the first set (Fig. 14), a BE flow cloned from wide42 ran through multiple

Fig. 13. Parking lot topology adapted from Fig. 10 in [40]

Fig. 14. Illustration of data flows for the first set of multiple bottlenecks tests

bottlenecks on the path (H1,R1,R2,R3,R4,H2). Another BE flow cloned from wide25 passed through the same bottlenecks in the reverse direction. There were four tests in the first set. In the first test, the BE flows ran alone without any concurrent LBE flows. In the rest of tests, the BE flows ran simultaneously with three LBE flows $H3 \rightarrow H8$, $H4 \rightarrow H9$ and $H5 \rightarrow H10$. All three low priority flows used the same LBE CC. The BE flows were started first together. Sixty seconds later, the LBE flows were started at 10-second intervals. The results are shown in TABLE III. U is the utilization of a certain LBE CC achieved at a specific bottleneck. TD, QD and RR are respectively the throughput degradation, 90_{th} percentile QD and retransmission rate measured by the data direction BE flow. The first row shows the results of the first test and the last three rows show the result of the rest of tests.

FlexiS flows have similar utilization at three different bottlenecks. The utilization is not high due to the sensitivity of the congestion detector. LEDBAT-BA and LEDBAT-SS flows have higher utilization but also higher impact on the BE flow.

For tests in the second set (Fig. 15), BE flows ran on

TABLE III
UTILIZATION AND IMPACT OF LBE CCs AT DIFFERENT BOTTLENECKS

CC	U (%) (R1,R2)	U (%) (R2,R3)	U (%) (R3,R4)	TD (%)	QD (ms)	RR (%)
W/OUT LBE	N/A	N/A	N/A	N/A	109	5.09
FlexiS	54	54	58	2.84	157	2.09
LEDBAT-BA	87	94	96	5.87	253	2.37
LEDBAT-SS	99	103	103	13.30	263	3.15

Fig. 15. Illustration of data flows for the second set of multiple bottlenecks tests

TABLE IV

TRACES USED TO LOAD THE VERTICAL PATHS OF THE PARKING LOT TOPOLOGY

Name	Date of Cap.	Dir.	Avg. Rate	Stddev
wide28	2008.03.24	downstream	28.54	11.23
wide36	2008.03.24	upstream	36.11	9.79
wide42	2020.12.27	downstream	42.99	20.76
wide25	2020.12.27	upstream	25.55	7.58
wide47	2009.12.12	downstream	47.64	25.57
wide37	2009.12.12	upstream	37.66	6.80

the vertical paths with each passing through one bottleneck only. The trace files used to clone the BE flows are listed in TABLE IV. LBE flows ran through multiple congested gateways on path (H1,R1,R2,R3,R4,H2). Let N be the number of LBE flows run concurrently. We have $N = 1$ or $N = 32$. For each N , four tests were run. In the first test, the BE flows were the only traffic in the network. In the rest of tests, the BE flows ran simultaneously with N LBE flow(s). The BE flows were started first together. Sixty seconds later, the first LBE flow was started. The rest of LBE flows were started at 10-second intervals if any. The utilization of a single (or 32) FlexiS, LEDBAT-BA and LEDBAT-SS flow(s) are 41% (92.44%), 75% (129.52%) and 82% (152.32%) respectively. Clearly, increasing the number of LBE flows can increase utilization. For the case of LEDBAT-BA or LEDBAT-SS, 32 flows will over-utilize available bandwidth and hence cause higher intrusion to the BE flow. Fig. 16a, 16c and 16e show the intrusion inflicted by one LBE flow and Fig. 16b, 16d and 16f show the intrusion caused by 32 LBE flows.

A single FlexiS flow has negligible impact on the BE flows. Thirty-two FlexiS flows, however, can "steal" more bandwidth from the BE flows, but do not obviously alter QD or retransmission rate of the BE flows. For most of the cases, LEDBAT-BA and LEDBAT-SS have higher impact on BE flows than FlexiS.

D. Internet Tests

The goal of Internet tests is to study the performance of FlexiS in a more realistic environment. The sender was located in mainland China. The receivers were perfSONAR servers [41] located at different places in the world. TABLE V

Fig. 16. Performance degradation of the BE flow at different bottlenecks caused by 1 (left) or 32 (right) LBE flows traversing all bottlenecks.

TABLE V
PERFSONAR SERVERS USED IN THE INTERNET TESTS

ID	IP Address	Geo Location	mRTT
1	134.75.253.98	Hong Kong	85
2	195.221.27.242	CRIANN, France	212
3	129.88.196.129	Grenoble, France	187
4	195.251.201.190	Ioannina, Greece	201
5	128.111.174.171	California, USA	172
6	192.12.207.22	California, USA	171
7	130.18.0.31	Mississippi, USA	226
8	155.68.2.2	Pennsylvania, USA	224
9	198.17.13.11	Washington, USA	177
10	137.63.159.254	Kampala, Uganda	375

lists the perfSONAR servers used in our test along with the observed minimum RTT (mRTT, in ms). The experiments were conducted over a course of four days – 2021.12.03, 2021.12.05, 2021.12.07 and 2021.12.09. Each test started from noon of an experimentation day and lasted for 24 hours. Every two hours, the sender connected to the chosen perfSonar servers in turn and RTT and throughput were measured. For each server, ping was first used to measure RTT. An RTT test lasted for 60 seconds. Packet interval of ping was set to 0.02 seconds. Then the throughput of FlexiS, LEDBAT-HZ and cubic were measured using iperf. Each throughput test lasted for 60 seconds. The starting order of the CCs was randomized. There was a 5-second interval between tests.

Fig. 17. Internet tests: throughput of LBE flows compared to congestion score and best effort bandwidth share

Due to the lack of an accurate way of estimating available bandwidth of the Internet paths, RTTs are used as a rough indicator of congestion. Let d_{min_i} and d_{avg_i} be the minimum and average RTTs measured in the i_{th} RTT test, D_{MIN} be the minimum RTT measured during the entire four-day experiment. If the route between the sender and a receiver did not change during our tests and we could correctly measure its base RTT, the following is assumed. If d_{min_i} is larger than D_{MIN} , there was congestion on the route during the throughput tests following the i_{th} RTT test, otherwise, there was available bandwidth. However, route might have changed or we could not discover its base RTT during our experiment. In which case, the throughput of cubic can be used as a reference because it represents the best effort bandwidth share and an LBE CC should never use more than that share.

For easy comparison, we define a score s for congestion. $s = \frac{\sum_{i=1}^n (0.8 \times (d_{min_i} - D_{MIN}) + 0.2 \times (d_{avg_i} - D_{MIN}))}{n}$, where $n = 48$ is the number of RTT tests per perfSonar server. The smaller the score, the lighter the congestion and vice versa.

Fig. 17 shows the relationship between the average throughput of FlexiS and LEDBAT-HZ and the congestion score. For most of the time, the throughput of FlexiS is in inverse relationship with the congestion score. That is, when the score is high, the throughput is low, which implies that FlexiS could react to congestion correctly. In contrast, the average throughput of LEDBAT is less correlated to the congestion score. The average throughput of both FlexiS and LEDBAT are well below that of cubic for all tests.

V. OPEN ISSUES AND FUTURE WORK

We have identified a number of limitations of FlexiS during our evaluation.

First, bandwidth utilization of a single FlexiS connection is low for many situations. The main cause is the use of a very sensitive congestion detector. A future work is to investigate whether adjusting protocol parameters alone can improve utilization in the mean while does not exacerbate intrusion. As a temporary remedy, establishing multiple FlexiS connections between a sender and a receiver can greatly improve utilization.

Second, FlexiS cannot distinguish congestion on the data path from that on the ack path due to the use of trend of RTT as incipient congestion indicator. A possible solution is to use

trend of OWD as congestion detector. This requires either a more accurate remote timestamp clock frequency estimator or the support of the receiver. However, an accurate frequency estimator is not easy to devise and getting the support of the receiver having deployment difficulties.

Third, FlexiS currently does not reduce cwnd below 2 MSS, which means that the sending rate cannot be reduced below $R_{min} = 2 \times MSS/RTT$. When available bandwidth share is less than R_{min} , the intrusion of FlexiS can be high. And the degree of intrusion increases with the increase of the number of FlexiS flows. A future work is to investigate how to remove the lower limit of sending rate. Possible solutions are reducing cwnd below 2 MSS or keeping minimum cwnd at 2 MSS but reducing the size of segments below 1 MSS. This first solution will greatly reduce RTT sampling frequency, which results in delayed response to network state changes. The second solution cannot ultimately resolve the problem because packet headers also consume bandwidth; furthermore, it is not cost effective.

Finally, the space and time complexity of the current implementation of the Theil-Sen estimator are both $O(n^2)$, where $n \leq \tau$. A large τ may cause problems such as increased processing delay and decreased protocol performance. A future work is to devise a more efficient way of implementing the estimator.

APPENDIX

THE EFFICACY OF TRAFFIC SYNTHESIS

In this section, we verify, with experiments, that the flows cloned from sub-traces of a trace file can be synthesized into traffic that has great similarity to the original one. The dumbbell topology (Fig. 1) was used in this experiment. The bandwidth of the bottleneck link was 100 Mbps. Queuing discipline of the bottleneck was FIFO in unit of packets. The limit of the buffer was set to 1000 packets or 1.2 BDP of a 100 ms flow. Let wide36 denote the trace captured at sample point F of the WIDE backbone on 2007.01.01. Its input/output rate is 36.21 Mbps and standard deviation is 3.91 Mbps. It contains bidirectional data. It was first split into two unidirectional files based on flow directions. Each of them was further split into 9 sub-trace files. The nine downstream sub-traces were used to load the data direction paths and upstream sub-traces were used as ack direction loads. All 18 sub-traces were started at the same time. Tcpdump was used to capture packets at router R1 from the start of the sub-traces. The experiment lasted for 120 seconds. Tshark was used to generate byte I/O statistics in one second intervals from wide36 and from the pcap file captured at R1 of the emulated network. Figure 18 shows the byte I/O statistics of wide36 and the synthesized traffic. As we can see the latter has great similarity to the former.

ACKNOWLEDGMENTS

This work is partially supported by the University of Oslo. The author would like to thank Michael Weigl for the introduction of LBE services and the CORE emulator and the suggestions and advice given in the early stage of this work.

Fig. 18. A comparison of I/O graphs of the original and synthesized traffic

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)," 2012.
- [4] R. Jain, "A delay based approach for congestion avoidance in interconnected heterogeneous computer networks," *ACM Computer Communication Review*, 1989.
- [5] Z. Wang and J. Crowcroft, "A new congestion control scheme: Slow start and search (tri-s)," *ACM Computer Communication Review*, 1991.
- [6] —, "Eliminating periodic packet losses in 4.3-tahoe bsd tcp congestion control," *ACM Computer Communication Review*, 1992.
- [7] L. Brakmo, S. O'Malley, and L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," in *ACM SIGCOMM*, 1994.
- [8] "News from the internet congestion control world," *arXiv*, 2009.
- [9] D. Ros and M. Welzl, "Less-than-best-effort service: A survey of end-to-end approaches," *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 2013.
- [10] "Ledbat++: Congestion control for background traffic," 2020.
- [11] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, 1989.
- [12] S. Floyd, "Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic," *ACM SIGCOMM Computer Communication Review*, 1991.
- [13] T. R. Henderson, E. Sahouria, S. McCanne, and R. H. Katz, "On improving the fairness of tcp congestion avoidance," in *IEEE GLOBECOM*, 1998.
- [14] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," *IEEE/ACM TRANSACTIONS ON NETWORKING*, 2003.
- [15] 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.
- [16] T. Tsugawa, G. Hasegawa, and M. Murata, "Background tcp data transfer with inline network measurement," *IEICE Transactions on Communications*, 2006.
- [17] M. M. C. L. T. Man, G. Hasegawa, "Imtcp: Tcp with an inline measurement mechanism for available bandwidth," *Computer Communications*, 2006.
- [18] G. Carofiglio, L. Muscariello, D. Rossi, C. Testa, and S. Valenti, "Rethinking the low extra delay background transport (ledbat) protocol," *Computer Networks*, 2013.
- [19] S. Q. V. Trang, E. Lochin, C. Baudoin, E. Dubois, and P. G'elard, "Flower – fuzzy lower-than-best-effort transport protocol," in *40th Annual IEEE Conference on Local Computer Networks*, 2015.
- [20] N. T. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. Bershad, "Receiver based management of low bandwidth access links," in *IEEE INFOCOM*, 2000.
- [21] P. Mehra, A. Zakhori, and C. D. Vleeschouwer, "Receiver-driven bandwidth sharing for tcp," in *IEEE INFOCOM*, 2003.
- [22] A. P. D. A. Hayes, D. Ros and I. Ahmed, "A framework for less than best effort congestion control with soft deadlines," in *IFIP Networking Conference and Workshops*, 2017.
- [23] M. C. Weigl, K. Jeffay, and F. D. Smith, "Delay-based early congestion detection and adaptation in tcp: impact on web performance," *Computer Communications*, 2005.
- [24] S. Bhandarkar, Y. Z. A. L. N. Reddy, and D. Loguinov, "Emulating aqm from end hosts," in *ACM SIGCOMM*, 2007.
- [25] J. Martin, A. Nilsson, and I. Rhee, "Delay-based congestion avoidance for tcp," *IEEE/ACM Transactions on Networking*, 2003.
- [26] A. A. Awadallah and C. Rai, "Tcp-bfa: Buffer fill avoidance," in *IFIP High Performance Networking Conference*, 1998.
- [27] R. Bless, "A lower-effort per-hop behavior (le phb) for differentiated services (rfc8622)," 2019.
- [28] L. Eggert and J. D. Touch, "Idle time scheduling with preemption intervals," in *20th ACM Symposium on Operating Systems Principles*, 2005.
- [29] P. Key, L. Massouli'e, and B. Wang, "Emulating low-priority transport at the application layer: A background transfer service," 2004.
- [30] H. Theil, "A rank-invariant method of linear and polynomial regression analysis. i, ii, iii," in *Koninklijke Nederlandse Akademie Wetenschappen*, 1950.
- [31] P. K. Sen, "Estimates of the regression coefficient based on kendall's tau," *Journal of the American Statistical Association*, 1968.
- [32] "bmon - bandwidth monitor," <https://github.com/tgraf/bmon>.
- [33] R. Jain, D. M. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *ArXiv*, 1998.
- [34] 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.
- [35] "Common open research emulator," <https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/CORE/>.
- [36] "Multi-generator (mgen) network test tool," <https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/MGEN/>.
- [37] "Mawi working group traffic archive," <https://mawi.wide.ad.jp/mawi/>.
- [38] 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.
- [39] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional integral controller enhanced (pie): A lightweight control scheme to address the bufferbloat problem (rfc8033)," 2017.
- [40] D. Hayes, D. Ros, L. Andrew, and S. Floyd, "Common tcp evaluation suite," 2014, internet Draft.
- [41] "perfsnar," <https://www.perfsnar.net/index.html>.

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 delay based congestion control, routing protocols for emerging networks, and Internet traffic classification.