Automated Identification of BBR Traffic based on Packet Inter-Arrival Times Analysis

Ziad Tlaiss*†, Alexandre Ferrieux*, Isabel Amigo†, Isabelle Hamchaoui*, Sandrine Vaton†
IMT Atlantique, Lab-STICC laboratory (Brest, France) †
Orange Labs Networks (Lannion, France)*
Emails: †{firstname.lastname}@imt-atlantique.fr, *{firstname.lastname}@orange.com

***Abstract*—The Internet is a complex and constantly evolving system, and congestion control algorithms play a crucial role in ensuring its functioning by managing network performance. These algorithms regulate the flow of data within a network and optimize data transmission for efficiency and effectiveness. They do this by continuously estimating available network resources and adjusting the data transmission rate accordingly.**

For network operators, identifying the congestion control algorithms being used on their network is essential to gain valuable insights into network performance and device behavior. This information can help them gain a better understanding of how the network is being utilized and which algorithms are most effective in different scenarios. With a clear understanding of the congestion control algorithms in use, they can make decisions about network design, configuration, and management.

Nowadays, over 85% of total Internet traffic is TCP traffic. TCP uses different congestion control algorithms, of which BBR and CUBIC represent 73% of the total TCP traffic. In this work, we present a method for automatically identifying BBR traffic on the Internet. Our method relies on analyzing packet inter-arrival times, specifically comparing the distribution of packet inter-arrival times during the Slow-Start state of a BBR capture with those of a CUBIC capture. We introduce a model that allows us to detect the silence period after packet bursts that are present in almost all non-BBR congestion control algorithms. This method is characterized by a very simple frontend signal processing that exploits the algorithms' core principles, allowing for a tiny parameter space dimension (two), which is sufficient for robust discrimination: an error rate of 4.1% was obtained on a test dataset independent from training.

Index Terms: Packet trace, TCP variants, packet inter-arrival times, Slow-Start state, Congestion Control Algorithms, BBR.

I. Introduction

Today's Internet is a crowded highway, with multiple services and applications competing to provide the best quality of experience to users while sharing the same network infrastructure. The Internet traffic is segmented into packets, each representing an individual unit of data. When the network's capacity cannot handle the transmission of all the traffic from source to destination, the network becomes congested [1]. To ensure reliable transmission between the server and user and to successfully send all the data over the network, the Transmission Control Protocol (TCP) was introduced in the late 1980s [2]. To offer the best experience to each Internet user in a fair manner for others, Congestion Control Algorithms (CCAs) were introduced [3] as a refinement of the initial TCP algorithm. CCAs control the number of packets that can be transmitted over a network at any given time, depending on the network conditions. They typically do this by controlling the data transmission so as not to exceed the so-called congestion window (cwnd) size [4]. The cwnd is a TCP state variable that represents the maximum amount of unacknowledged data that can be tolerated by the sender. The cwnd size is continuously determined by the CCA and adjusted based on network conditions. Depending on the type of CCA, the cwnd size is modified to achieve the best performance while taking into consideration factors such as packet losses, delay, Packet Delay Variation (PDV), etc.

CCAs can be classified into three different types: loss-based, delay-based, and rate-based. Loss-based CCAs use packet losses to determine the cwnd size: a loss event is the primary congestion signal, leading to deciding a reduction in cwnd. The most well-known loss-based CCA is CUBIC [5]. Delay-based CCAs use the variation of latency as the congestion signal. VEGAS is the best-known CCA in that category [6]. Finally, the so-called Bottleneck Bandwidth and Round-Trip-Time (BBR) was introduced more recently. BBR is a rate-based CCA: it uses an estimation of the available bandwidth along with the round-trip-time (RTT) to build a model of the network characteristics to determine the packet sending rate [7].

Several studies have found that TCP represents 85% of total internet traffic [8], [9]. Currently, CUBIC and BBR are estimated to represent more than 73% of the Internet traffic [10]. Due to the different methods of estimating the cwnd and detecting congestion, significant differences in throughput can be observed with CUBIC and BBR even when they share the same network. These differences are strongly correlated with certain network conditions that give one of them an advantage over the other [11]. For this reason, network operators are interested in monitoring the growth and percentage of traffic for each of CUBIC and BBR CCAs, to optimize their network infrastructure.

This work presents a method to automatically differentiate between TCP traffic controlled by CUBIC or BBR. We focus on the so-called Slow-Start (SS) state phase, i.e. the beginning of the connection, which features a well-defined, predictable behavior in both CCAs. Given a TCP connection to classify, we isolate the SS phase; then we track and compare the emission patterns over this period, where our analysis has revealed a significant difference between CUBIC and BBR. To differentiate between the CCAs, we primarily focus on

the distribution of packet inter-arrival times. Then, to get better robustness to noise, we introduce a resampling of the distribution, that leads to a much clearer discrimination.

Using the cumulative distribution function (CDF) of packet inter-arrival times with this resampling, we obtain a highly separable representation, making it possible to classify by simply computing the position of the CDF with respect to a decision point. This decision point is computed in the training phase. To assess our method we used two independent (training and test) datasets, of hundreds of labeled captures each. The total error rate obtained was 2.6% on the training set, and 4.1% on the test set.

The remainder of this paper is organized as follows. In Section II we develop the motivation for this work. In Section III, we survey the related work on the identification of CCA variants. An overview of BBR and CUBIC CCAs is given in Section IV. Our model for identifying BBR CCA is presented in Section V. An evaluation of our method is presented in Section VI. The advantages and challenges of our method are discussed in Section VII. Finally, a conclusion is given in Section VIII.

II. Motivation

The main objective of our approach is to detect whether or not pacing is used in a TCP connection. In a paced TCP, instead of sending new packets immediately after receiving an acknowledgment, the packets are held back for a certain duration, which results in less bursty TCP traffic [12]. In other words, TCP pacing is used to evenly space data sent into the network over an entire round-trip-time; in this case, data is not sent in a burst. To measure the fraction of TCP traffic that is paced, we illustrate this using BBR and CUBIC CCAs. BBR, with its novel approach both to congestion detection and sending rate control, is a TCP CCA that paces. On the other hand, CUBIC, a loss-based algorithm that reduces its sending rate in case of high levels of packet loss, is a TCP CCA that does not pace. Recent studies show the impact of pacing on improving TCP performance [13], especially in the case of shallow buffers (i.e. small buffer size in routers). Hence, the number of paced TCP traffic might increase in the future to cover all TCP CCA flavors.

For network operators, understanding whether the TCP in use on their network is paced or not is crucial. For instance, detecting whether pacing is employed within operators' networks helps them measure the burstiness of the resulting TCP traffic. This burstiness might have an impact on how network buffers are dimensioned; operators need to adjust buffer sizes and network management strategies accordingly. For example, the guidelines outlined in [14] might no longer be applicable. Additionally, bursty traffic can lead to sudden spikes in network utilization, potentially causing congestion. By identifying bursty traffic, operators can implement measures to smooth out traffic patterns, thereby preempting congestion-related issues.

III. Related work

A large number of papers have been published on the identification and inference of TCP CCA. This related work section focuses on those more directly related to our work.

Padhye et al. [15] developed the TCP Behavior Inference Tool (TBIT), which performs active measurements to infer various TCP behaviors such as the initial window and congestion window (cwnd) of a remote Web server. TBIT can also detect which of the following CCAs is running on a Web server: Reno, New Reno, Reno Plus, or Tahoe.

Yang et al. [16] proposed an active CCA identification approach that uses a random forest algorithm to classify the CCA variants of a Web server. The classification is based on two features: the multiplicative decrease parameter applied when a loss is detected during the SS state and the window growth function driving the congestion avoidance state. The authors were able to identify several famous CCAs, such as NewReno, BIC, VEGAS, and CUBIC.

Mishra et al. [10] developed Gordon, an active tool that measures the congestion window size and identifies TCP CCA variants among websites. Gordon measures the cwnd and then analyzes the reaction of the TCP variants to packet losses to classify them. In particular, depending on the decrease factor after a loss or/and the increase factor during the congestion avoidance state, the TCP variant is identified. To do this they do not rely on common active measurements, but manipulate the client to force the server to react against several scenarios, generating a considerable amount of traffic. For rate-based CCAs such as BBR, which does not change its cwnd after a loss, the no-loss reaction is used to determine the CCA as BBR or unknown.

None of the previously mentioned works address the identification of BBR on its own by analyzing a packet capture, nor do they compare CUBIC and BBR traffic. Our method differs from these works in that it relies only on the analysis of bidirectional packet traces obtained from classical TCP downloads to identify BBR traffic. Unlike [15] and [10], we do not need to generate multiple traffic patterns emulating various perturbations, as our method consists of analyzing a single, passive capture of a full connection. We also differ from [16] as we do not use a heavyweight machine learning algorithm: indeed, our parameter space is extremely small, thanks to a novel, well-motivated feature extraction step that taps into the core design differences between the CCAs. And similarly to the aforementioned works, we do not need to process our data online, as our objective is to allow operators to quantify the amount of each CCA present on their networks and to detect a paced TCP traffic.

IV. Background: CUBIC vs BBR

In this section, we provide some background on CUBIC and BBR CCAs. We discuss the different states of each CCA, with a focus on the initial state: Slow-Start for CUBIC and Startup for BBR. We compare and contrast the similarities and differences between these two states.

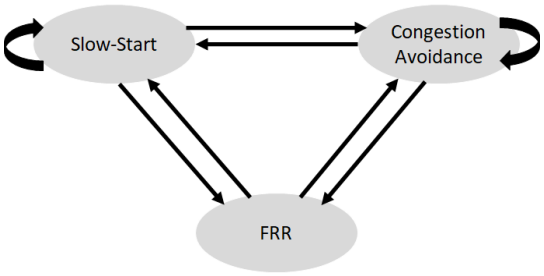


Figure 1: Finite State Machine of CUBIC.

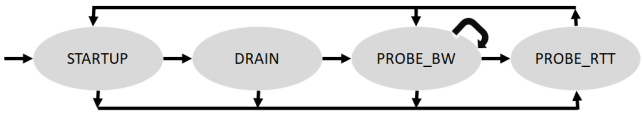


Figure 2: Finite State Machine of BBR.

A. Overview: CUBIC & BBR CCA states

1) CUBIC: CUBIC is a loss-based congestion control algorithm for TCP, that uses a cubic function to model the temporal evolution of the cwnd after a congestion event [5]. The behavior of the algorithm is represented by a finite state machine (FSM) with three states: Slow-Start (SS), Congestion Avoidance (CA), and Fast Retransmit & Fast Recovery (FRR). Figure 1 shows the FSM of the CUBIC CCA.

2) BBR: BBR is a congestion control algorithm introduced by Google in 2016 [7], designed to use available network resources more efficiently than CUBIC. BBR optimizes the sending rate of the packets based on estimated network conditions, with a focus on the Bandwidth-Delay Product (BDP), calculated as the measured average bandwidth multiplied by the minimum Round Trip Time (RTT). Figure 2 shows the FSM of BBR CCA, with four states: Startup, Drain, ProbeBW, and ProbeRTT.

B. Slow-Start and Startup

Now we're looking in more detail into the Slow-Start and Startup phases, which will later be shown to be of high relevance for our classification task.

1) CUBIC's Slow-Start (SS) state: During the SS phase, CUBIC increases its packet sending rate exponentially to quickly reach the bottleneck capacity. The amount of data transmitted between the sender and receiver is controlled by the minimum of the cwnd and the maximum receiver window (rwnd) [4]. The first packets are sent according to the Initial Congestion Window (ICW) and the cwnd size is doubled after every round-trip-time (RTT) upon reception of acknowledgments for the sent packets [17]. This is a result of how CUBIC operates, as it requires confirmation of received packets through acknowledgments before proceeding to send another burst of packets. In particular, this results in an ON/OFF traffic pattern emission as will be developed in Subsection V-B. Finally, upon a congestion signal reception, such as packet loss or increased latency, as indicated by Hystart [18], or when the SS threshold (ssthresh) is reached, CUBIC switches from the SS phase to the congestion avoidance state [19].

2) BBR's Startup state: Similarly, during its Startup phase, BBR performs a binary search and increases its sending rate exponentially by doubling the number of transmitted packets in each round-trip period. The Startup phase of BBR and the SS phase of CUBIC have a similar exponential increase in average rate, but they differ in the detailed emission mechanism. Unlike CUBIC, BBR does not wait for packet acknowledgments before sending new packets; it relies instead on estimates of available resources to drive a slowly-varying packet shaper. This results in a "smooth" emission pattern, without any of the macroscopic-scale "pauses" exhibited by CUBIC (see Subsection V-B for more details). Finally, once BBR determines that the pipe is full by comparison of in-flight with the estimated BDP, it exits the Startup phase and enters the Drain state, similar to CUBIC's transition to Congestion Avoidance.

As they play similar operational roles, in the remainder of this paper, we call both of these states, for either CUBIC or BBR, the SS phase.

V. Methodology

In the present section, we propose an identification method based on the different behavior of the CCAs during the SS state. We first describe the nature of input data (Subsection V-A), then we elaborate on differences observed on BBR and CUBIC, with the highest contrast visible during the SS state, allowing us to characterize each of them based on the analysis of packet traces (Subsection V-B).

Based on this, in order to detect BBR on a given packet capture, we first need to automatically detect the end of the SS state. In prior work, we presented a method to that effect [20]; we briefly present it in Subsection V-C.

The core of our CCA discrimination method is then presented in Subsection V-D. The method exploits the distinguishing characteristics of emission patterns of each CCA during the SS state. In particular, it builds on the empirical distribution of packet inter-arrival times, with a specific resampling step. The automatic method is completed with the introduction of a decision point, calculated as explained in Subsection V-E.

A. Input data representation

In this work, the tcpdump [21] tool was used to capture TCP packet header traces together with their arrival times as shown in Figure 3. These captures have been processed, to extract important timestamped indicators. The most significant ones for the current task are:

- Sequence number (SEQ): identifies the first byte in a segment [22]. For a better match-up with the acknowledgments, we denote SEQ as the last byte of the transmitted

Time (sec)	SEQ (byte)
47.740081	38679095
47.740272	38680543
47.740291	38681991
47.740314	38683439
47.740322	38684887
47.740533	38686335
47.740551	38687783
47.765699	38689231
47.765717	38690679
47.765737	38692127
47.765745	38693575
47.810710	38695023
47.810728	38696471
47.810748	38697919
47.810756	38699367
47.810978	38700815

Time (sec)	ACK (byte)
47.739973	38616831
47.740098	38619727
47.740118	38622623
47.765639	38628415
47.810644	38644343
47.811003	38650135
47.833856	38655927
47.872981	38664615
47.873527	38673303
47.949871	38684887
47.949995	38687783
47.975502	38693575
48.020602	38709503
48.020855	38715295
48.043443	38721087
48.082784	38732671

Time (sec)	BIF (byte)
47.740081000	62264
47.740272000	57920
47.740291000	59368
47.740314000	60816
47.740322000	62264
47.740533000	63712
47.740551000	65160
47.765699000	60816
47.765717000	62264
47.765737000	63712
47.765745000	65160
47.810710000	50680
47.810728000	52128
47.810748000	53576
47.810756000	55024
47.810978000	56472

Figure 3: A packet trace showing the SEQ, ACK, and BIF with their arrival times.

segment plus one, i.e. $SEQ = sequence\,number + length$.

- Acknowledgment (ACK): informs the source about the sequence number of the next expected segment.
- Bytes in flight (BIF): indicates the amount of data bytes sent by the source but not yet acknowledged. BIF is not included in packet headers but can be deduced from SEQ and ACK values by deducing the last received ACK value from the last emitted SEQ value as shown in Figure 4. By definition of cwnd, the BIF value is bounded by the cwnd size at any given time.
- Round-Trip-Time (RTT): represents the delay between a packet emission and the reception of the corresponding acknowledgment. We can calculate the RTT value using the SEQ and corresponding ACK arrival times as shown in Figure 4.
- Packet inter-arrival time (INTRPKT): represents the elapsed time between the arrival of two consecutive packets.

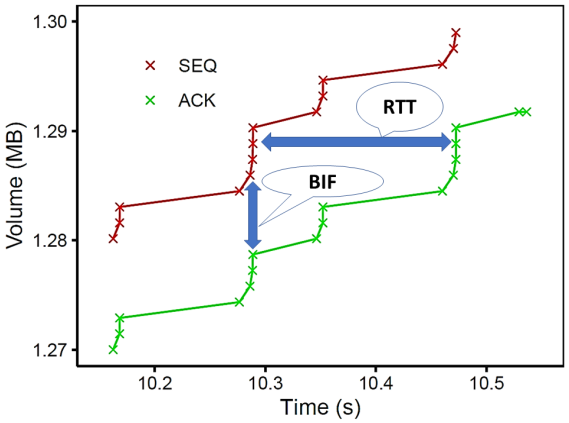


Figure 4: BIF and RTT calculation method

B. Characterization of CUBIC and BBR during SS state

In the context of a connection, it is important to note that only the source endpoint possesses explicit information regarding the currently employed CCA flavor (CUBIC or BBR). As a consequence, our study relies on controlled sources, specifically, our laboratory servers, for trace data to calibrate our recognition algorithm. Extension of this method to real-life traffic from third-party servers will be considered in section VI.

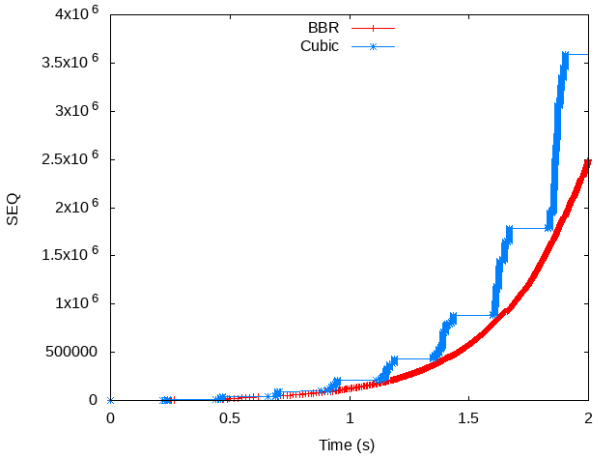


Figure 5: Time-sequence graph of real CUBIC and BBR connections during SS

Through the examination of traces, it becomes evident that the initial stages of a connection reveal distinct emission patterns between BBR and CUBIC. In Figure 5, we provide an illustrative example that highlights these differences in emission patterns during SS.

CUBIC typically exhibit bursty emissions, whereas BBR displays a smoother emission pattern. This disparity arises from a fundamental distinction between the two CCAs: BBR operates on a rate-based mechanism, whereas CUBIC utilizes a window-based approach. In other words, CUBIC employs a burst-like behavior where it sends its entire window of packets at once and then waits for acknowledgments. This results in the observable pattern of packet bursts followed by periods of silence in Figure 5. This behavior is inherent to CUBIC, as it waits for acknowledgments to ensure the successful reception of sent packets before proceeding with the transmission of another burst. On the other hand, BBR follows a rate-based approach and paces its emissions accordingly. As a consequence, the ON/OFF pattern seen in CUBIC traffic is absent from BBR, which thus appears extremely smooth in comparison.

After exiting the SS state, CUBIC also tends to behave smoothly, as it is subject to the so-called ”ack clocking” phenomenon [4], due to bottleneck-induced pacing, as shown in Figure 6.

This precisely happens when reaching the bottleneck capacity, typically on the SS exit. Hence, the clear difference in ”burstiness” observed during the SS tends to be blurred afterward: the SS phase thus presents the optimal period for discrimination between CUBIC and BBR. Therefore, we isolate the SS state for each connection and focus on char-

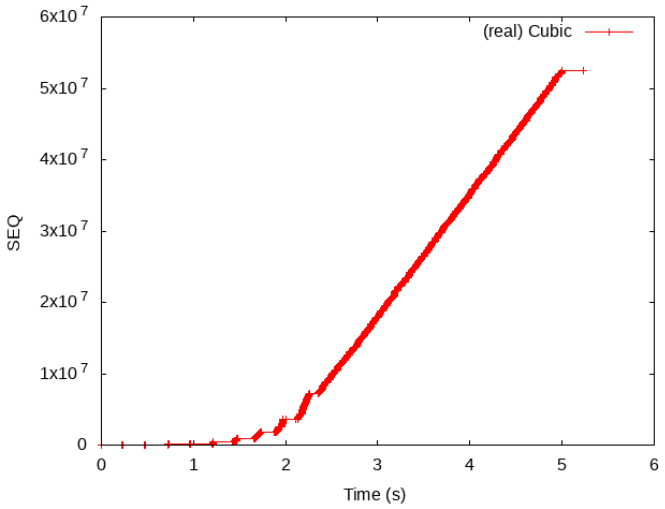


Figure 6: SEQ against time for a CUBIC capture - smooth behavior after exiting the SS state.

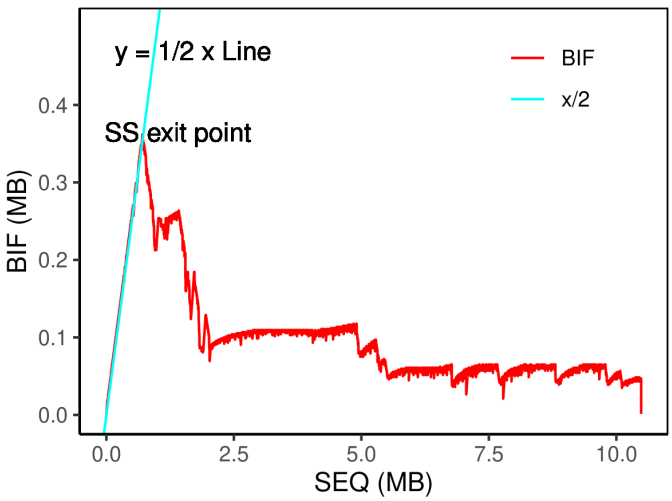


Figure 7: Automatically detecting SS state exit time with slope 1/2 method [20].

acterizing the emission patterns, specifically the burstiness, during this period. To evaluate these patterns, our approach involves analyzing the distribution of packet inter-arrival times (INTRPKT), derived from packet captures.

C. BIF vs SEQ representation for SS detection

In this work, we aim to efficiently identify the CCA variant by focusing on the SS state, as the ON/OFF pattern can easily be detected during this stage as illustrated before in Subsection V-B. To automatically recognize the SS phase, we will utilize the BIF versus SEQ representation introduced in [20]. The main benefit of this representation lies in its predictable shape and slope during the binary search phase.

Indeed, it can be shown that, during a phase of exponential rate growth, the representation shows a strong, linear correlation between BIF and SEQ. Moreover, if the rate is doubling at each RTT, the slope can be predicted, with $BIF = \frac{SEQ}{2}$. If the BIF value falls below the separation line $y = \frac{x}{2}$ without returning, as shown in Figure 7, it indicates that the SS state has ended and the CCA transitions to the next phase. For more details, we refer the reader to [20]. By taking advantage of the SS-detection method, it is possible to build a "BBR vs. CUBIC" classifier for TCP connections, which is both extremely cheap in training and runtime computational resources

D. Modelling the distribution of packet inter-arrival times

To get an intuition of the statistical properties of the INTRPKT distribution, we present the Probability Density Functions (PDF) for a BBR (red) versus CUBIC (blue) connection in Figure 8. Notably, these distributions demonstrate a distinct dissimilarity in shape. Unlike BBR, the CUBIC distribution is typically heavy-tailed, indicating a higher probability of longer inter-arrival times.

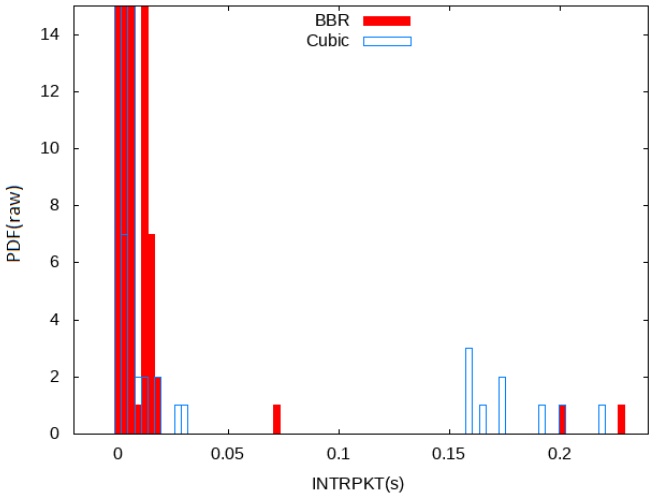


Figure 8: The PDF of INTRPKT

1) CUBIC distribution: The primary peak observed in the CUBIC inter-arrival distribution corresponds to the short INTRPKTs within bursts, while the heavy tail represents the longer INTRPKTs resulting from periods of silence between two bursts. If we denote N as the average number of packets in a burst, then the probability for an INTRPKT to belong to a burst is $\frac{N}{N+1}$, while the probability for it to belong to a silence between two bursts is $\frac{1}{N+1}$. We can then express the PDF of the INTRPKT as a mixture distribution f, as shown in this equation:

$$f(x) = \frac{1}{N+1}g(x) + \frac{N}{N+1}h(x) \quad (1)$$

Here, $g(x)$ is the PDF of the OFF period, while $h(x)$ is the PDF of packet interarrival times within bursts.

2) BBR distribution: On the other hand, a typical BBR distribution displays a single peak primarily focused on small

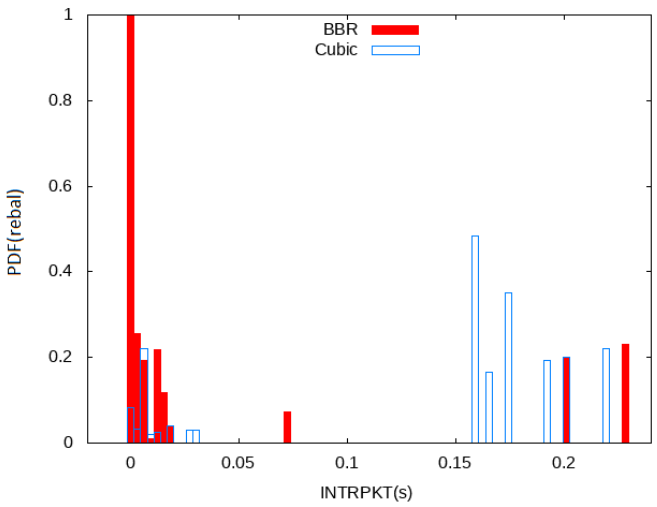


Figure 9: The PDF of INTRPKT after resampling (uniform over time)

values. This is because occurrences of large INTRPKTs are relatively rare in BBR due to its smooth emission pattern. Therefore, the discrimination task can be simplified to distinguish between single-peak distributions (BBR) vs. heavy-tailed distributions (CUBIC).

3) Expanding small contributions: In the CUBIC case, it can be noted in Eq. 1 that the large-INTRPKT component is dwarfed by the short-INTRPKT component. This is due to the significantly larger number of events within a burst compared to the relatively small number of pauses between bursts. However, in order to effectively discriminate between single-peak and heavy-tailed distributions, it is necessary to address this imbalance by amplifying the minority contribution, specifically the long-INTRPKT component.

Note that, as with any distribution, the CUBIC PDF (Eq. 1) can be approximated using an empirical distribution based on the measured INTRPKT values:

$$\hat{f}(x) = \frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{x_i}(x) \tag{2}$$

To magnify the minority contribution, we thus weight each observed INTRPKT value x_i by $w_i = \frac{x_i}{\sum_{j=1}^{T} x_j}$

So we now consider the rebalanced empirical distribution $\hat{f}_{bal}$:

$$\hat{f}_{bal}(x) = \sum_{i=1}^{T} w_i \mathbb{1}_{x_i}(x) = \frac{1}{\sum_{j=1}^{T} x_j}\sum_{i} x_i \mathbb{1}_{x_i}(x) \tag{3}$$

Notably, if the timescale is renormalized to $[0, 1]$ for the observation period (SS state), then $\sum_{j=1}^{T} x_j = 1$ and $\hat{f}_{bal}(x) = \sum_i x_i \mathbb{1}_{x_i}(x)$.

An interesting observation is that the resulting empirical distribution $\hat{f}_{bal}$ closely approximates the distribution of INTRPKTs that would be observed if sampling were uniform over time (Figure 9) rather than uniform over packets (Figure 8). Specifically, choosing each sampling instant uniformly over the observation period corresponds to a Poisson sampling approach. Due to the PASTA (Poisson Arrivals See Time Averages) property, the probability of a sampling instant occurring during a burst or a silence period is $\frac{T_{ON}}{T_{ON}+T_{OFF}}$ and $\frac{T_{OFF}}{T_{ON}+T_{OFF}}$, respectively, where T_{ON} (respectively T_{OFF}) is the average duration of a burst (respectively of a silence). Therefore $\hat{f}_{bal}(x)$ is an empirical approximation of the following mixture distribution:

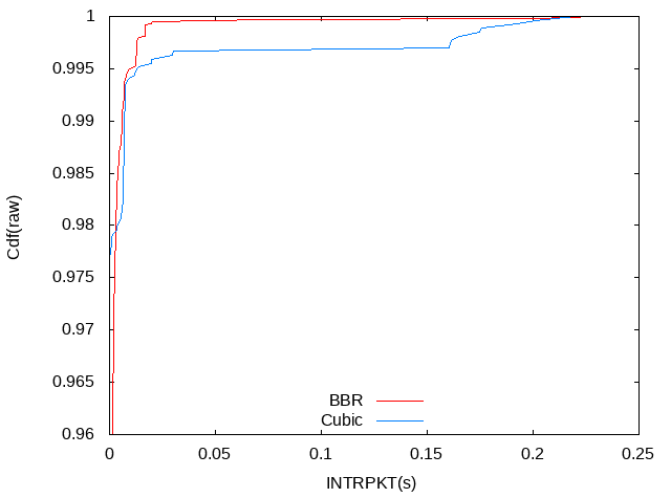
$$\phi(x) = \frac{T_{OFF}}{T_{ON}+T_{OFF}} g(x) + \frac{T_{OFF}}{T_{ON}+T_{OFF}} h(x) \tag{4}$$

It can be observed that the weight $\frac{T_{OFF}}{T_{ON}+T_{OFF}}$ of $g(x)$ in Eq. 4 is not negligible (contrary to the weight $\frac{1}{N+1}$ in Eq. 1). By replacing a uniform sampling strategy across packets with a uniform sampling strategy over time, more weight is assigned to the inter-packet intervals that correspond to silences between two bursts. This property is significant as it allows for better differentiation between the INTRPKT distributions of CUBIC and BBR.

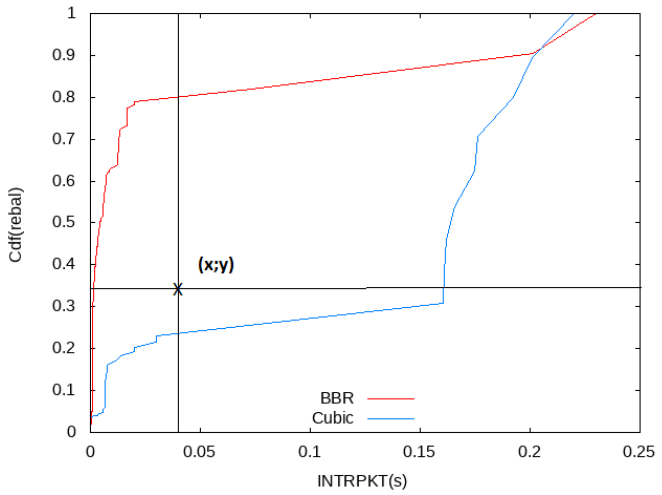
E. Choice of the optimal decision point

To capture distribution characteristics in a resolution-independent manner, we transition from PDFs to CDFs. Figures 10a and 10b respectively display the raw and re-balanced CDFs of INTRPKTs during the SS state for CUBIC and BBR. It can be observed that in the rebalanced case, which mimics uniform sampling over time, the two distributions exhibit much better separation compared to the raw case, which corresponds to the original uniform sampling over packets. The use of rebalanced CDFs enhances the distinction between the INTRPKT distributions of CUBIC and BBR, providing a more effective means of discrimination.

Let us assume that our goal is to identify if a TCP connection is BBR traffic or not. This problem can be considered as a test between two hypotheses: $H_0 : \{CUBIC\}$ and $H_1 : \{BBR\}$. We are going to propose a method that takes a decision based on the INTRPKT values $x_1, x_2, \ldots, x_T$ measured during the SS state. As there are more long INTRPKTs in CUBIC, then we decide the connection is using CUBIC if the proportion of values $(x_i)_{i=1,T}$ greater than a threshold x is larger than y. On the contrary, if the proportion of values $(x_i)_{i=1,T}$ greater than x is smaller than y, then we decide BBR. In simpler terms, we aim to find a point (x, y) in Figure 10b that effectively separates the red and blue curves. If the curve $(x, \tilde{f}(x))$ (Eq. 3) lies below this point, we classify it as CUBIC, and if it lies above the point, we classify it as BBR. Considering that there are two types of risks in a hypothesis test, namely false alarm, and non-detection, we propose fixing the value of $y = \theta(x)$ for a specific x. This choice ensures that both risks have an equal probability. In Figure 10b, this corresponds to positioning the point (x, y) approximately in the middle between the blue and red curves, assuming an equal number of CUBIC and BBR connections. The optimal x is then selected to minimize the total probability of error, which



(a) CDF of INTRPKT for a CUBIC and BBR capture using the raw distribution (uniform sampling over packets)



(b) CDF of INTRPKT for the same captures using the re-balanced distribution (uniform sampling over time). A typical decision point $(x; y)$ is shown.

Figure 10: Comparison between CDFs of a CUBIC and BBR packet capture using two sampling techniques - the blue curve represent CUBIC traffic and the red curve represents BBR traffic.

encompasses both false alarm and non-detection probabilities. This approach helps minimize the probability of misclassifying a connection.

VI. Method Evaluation

In this section, we evaluate our method and model for classifying connection captures, using two datasets: one of 221 packet captures for training, and the other of 583 packet captures for testing. We use the single decision point selected from our training dataset on the dataset dedicated for testing.

A. Packet traces characterization

The packet traces were captured on one of our servers, which was accessed by multiple active probes via the public Internet. The measurements were performed by conducting several downloads from our server with BBR and CUBIC CCA algorithms. Our server is based in Europe and our probes are on another continent.

For the sake of representativity, we positioned our probes across two different countries. In addition, our downloads were carried out during peak and off-peak hours, seeking different levels of congestion. Table I shows the variations in the exit time of the SS state across the different downloads (packet captures), showing different resource conditions for the different TCP connections. Indeed, as the CCA stays in the SS state as long as it estimates that the bottleneck capacity is not yet reached, the variation of the exit time of the SS state reveals variations in the available network resources. The average RTT of our captures varies between 100ms and 400ms, while the majority fall between 200ms and 300ms as shown in Table II. For the considered probe destinations, an RTT value less than 250 ms usually indicates a good QoE, on the other hand, an RTT value higher than 250 ms normally means a congested network.

SSET (sec)	<0.4	[0.4;1[	[1;1.6[	[1.6;2.2[	[2.2;2.8[	>2.8
Downloads	1%	14%	22%	30%	32%	1%

Table I: Distribution of exit time of the SS state (SSET) for the considered dataset. Variation in the exit time of the SS state can be an indicator of variation in the available network resources.

RTT value (ms)	<200	[200;250[	[250;300[	>300
Downloads	1%	53%	45%	1%

Table II: Distribution of RTT values for the considered dataset. For the considered destinations less than 250 ms corresponds to good connections and higher than 250 ms to a congested network.

B. Difficulties in selecting representative packet traces

Our study primarily focused on the analysis of TCP packet traces exhibiting good and poor connections. To identify TCP packet traces with bad connections, we employed a comparative approach by comparing the throughput obtained during file downloads using TCP BBR or TCP CUBIC with the throughput achieved using a UDP connection. UDP, being a connectionless protocol that prioritizes speed and efficiency over reliability, provides a reference point for comparison. In an ideal condition, the throughput of TCP and UDP connections should be similar. Therefore, any significant difference in throughput between the TCP and UDP connections is considered an indication of an anomaly within the TCP packet trace. This approach allows us to detect instances where the TCP connection deviates from the expected behavior, potentially pointing to underlying issues affecting the connection quality.

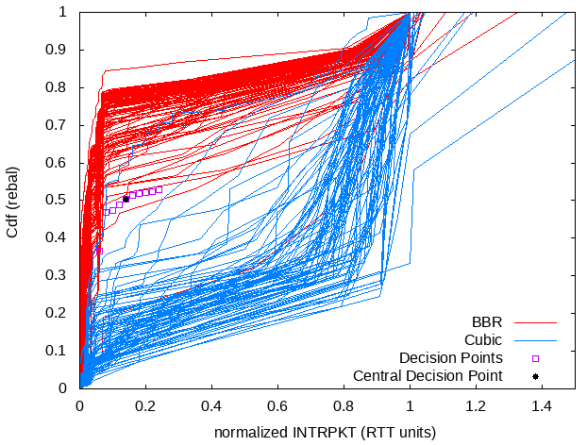


Figure 11: Choice of the decision point using 221 packet captures in which 88 CUBIC and 133 BBR captures - the blue curves represent BBR and the red curves represent CUBIC.

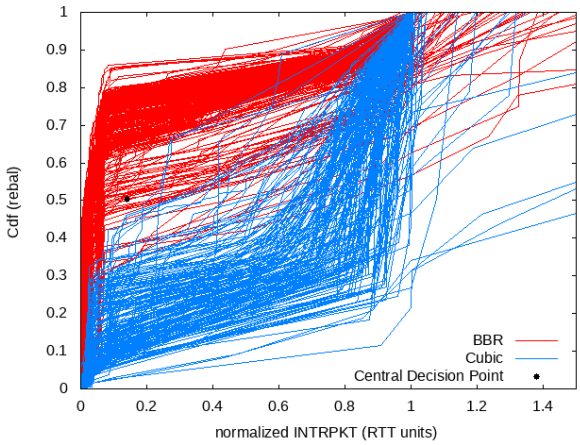


Figure 12: Model evaluation on an independent test set using 583 packet captures in which 389 BBR and 194 CUBIC - we have obtained a 4.1% total error rate.

Upon collecting the necessary packet traces, we conducted a detailed analysis of each trace to assure the collection of various packet traces. The manual classification was employed to identify and categorize each CUBIC or BBR download into good or bad connections. By examining various performance metrics, including throughput, we were able to gain insights into the nature and characteristics of the anomalies present in the TCP packet traces. This methodology enabled us to effectively collect a representative dataset. We note that the gathering and analysis of the 804 downloads used for training and evaluation were done over a 4-month period due to the substantial time required for labeling and analyzing each trace, which can be a time-consuming and resource-intensive process.

C. Choice of the decision point using a training dataset

To determine the optimal decision point, we conducted training on a dataset comprising 221 packet captures, including 133 BBR captures and 88 CUBIC captures. By utilizing the equal-error-rate minimization approach described earlier, applied to the resampled CDF curves of these packet captures as shown in Figure 11, we identified the decision point to be at $(x = 0.14, y = 0.503)$. In essence, the optimal decision criterion involves comparing the median of the resampled INTRPKT distribution with $0.14 * RTT$.

By employing the aforementioned decision point, the minimum total error rate attained on the training dataset amounts to 2.6%. Given the degenerate nature of the optimum (multiple points yielding the same error rate), we selected the central point to ensure a wider margin. This selection provides more robust and reliable discrimination between the CUBIC and BBR. The confusion matrix is displayed in Table III. 1 over 88 CUBIC captures would be identified as BBR and 2 over 133 BBR captures would be identified as CUBIC with the best decision point (x, y).

	Predicted CUBIC	Predicted BBR	Total
Actual CUBIC	87	1	88
Actual BBR	2	131	133

Table III: Confusion matrix of the training data.

D. Testing the decision point to identify BBR CCA

To assess our approach, we gathered a total of 583 packet captures, consisting of 389 BBR and 194 CUBIC captures. The resampled CDF curves of all 583 captures are displayed in Figure 12. By utilizing the decision point obtained from the training dataset, our model is capable of identifying the TCP variants with an overall error rate of only 4.1%. None of the 194 CUBIC captures are misclassified, while 16 out of the 389 BBR captures are mistakenly classified as CUBIC as displayed on the confusion matrix of Table IV. This slight bias indicates the need for a larger and more diverse training set, which we plan to incorporate in future work.

	Predicted CUBIC	Predicted BBR	Total
Actual CUBIC	194	0	194
Actual BBR	16	374	389

Table IV: Confusion matrix of the evaluated data.

VII. Model advantages and future challenges

We propose a "BBR vs. CUBIC" classifier for TCP connections, which is both extremely cheap in training and runtime computational resources (as the model comprises only two dimensionless scalars and feature extraction is trivial), and quite promisingly accurate as per our preliminary evaluation.

Despite our confidence in the approach, we recognize that various obstacles could jeopardize it if newer versions of CUBIC and BBR were to be widely adopted by the industry. The first that comes to mind is BBRv2 [23], the most

recent version of BBR, which incorporates an improved loss detection mechanism to better react to changes in network conditions. However, this improvement does not impact our method since the pacing rate during the SS state remains unchanged. Of more concern could be newer versions of CUBIC resorting to some level of pacing, which might blur the rather clear contrast with BBR during the SS phase. However, at the time of writing only a small part of providers turn to this option, partly due to its absence in the default stack tuning of popular operating systems. Should this state of affairs evolve, one might consider addressing this 3-class task with two decision points, with somewhat lowered accuracy.

VIII. Conclusion

In this work, we presented a method to automatically differentiate between BBR and CUBIC traffic. Our method focuses on the sending rate of each CCA during the slow-start phase, specifically by analyzing the inter-arrival times of packets. We used the empirical CDF of inter-arrival times, under a pertinent resampling allowing us to give significant weight to underrepresented events (long inter-arrival times, corresponding to off periods). These CDFs are then compared to a decision point calculated during the training phase to minimize the total equal error rate. Our method was trained on 221 packet traces, including 133 BBR and 88 CUBIC captures, and evaluated on 583 traces, including 389 BBR and 194 CUBIC captures collected over 4 months under various network conditions. Our method achieved a 4.1% total error rate with no false negatives among the 389 BBR captures.

On another aspect, the CCA classifier could take part in the investigation of how bursty the produced TCP traffic is, which depends on the fraction of TCP sources that pace. As paced TCP traffic shows better performance than no-paced TCP, operators are interested in detecting the burstiness of TCP traffic to improve their infrastructures in order to offer better quality of experience to their customers. In future work, we will focus on applying our method to other TCP CCA and testing it under a mix of BBR and no-BBR traffic happening simultaneously. We also consider extending our recognition and detection of the paced TCP traffic beyond the slow-start state to cover the congestion avoidance state.

References

[1] M. Welzl, D. Papadimitriou, B. Briscoe, M. Scharf, and M. Welzl, "Open Research Issues in Internet Congestion Control," RFC 6077, Feb. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6077

[2] W. Eddy, "Transmission Control Protocol (TCP)," RFC 9293, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9293

[3] S. Floyd, "Congestion Control Principles," RFC 2914, Sep. 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2914

[4] E. Blanton, D. V. Paxson, and M. Allman, "TCP Congestion Control," RFC 5681, Sep. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[5] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, p. 64–74, jul 2008. [Online]. Available: https://doi.org/10.1145/1400097.1400105

[6] M. Welzl and D. Ros, "A Survey of Lower-than-Best-Effort Transport Protocols," RFC 6297, Jun. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6297

[7] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, p. 58–66, jan 2017. [Online]. Available: https://doi.org/10.1145/3009824

[8] D. Lee, B. E. Carpenter, and N. Brownlee, "Observations of udp to tcp ratio and port numbers," in *2010 Fifth International Conference on Internet Monitoring and Protection*, 2010, pp. 99–104.

[9] R. Nelson, D. Lawson, and P. Lorier, "Analysis of long duration traces," *Computer Communication Review*, vol. 35, pp. 45–52, 01 2004.

[10] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great internet tcp congestion control census," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, pp. 1–24, 12 2019.

[11] C. Yi, J. Arpit, S. Kriti, B. Aruna, and G. Anshul, "When to use and when not to use bbr: An empirical analysis and evaluation study," *IMC '19: Proceedings of the Internet Measurement Conference*, pp. 130–136, 10 2019.

[12] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of tcp pacing," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 3, 2000, pp. 1157–1165 vol.3.

[13] S. Chavan, "Should paced tcp reno replace cubic in linux?" in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, 2016, pp. 1–8.

[14] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, p. 281–292, aug 2004. [Online]. Available: https://doi.org/10.1145/1030194.1015499

[15] J. Padhye and S. Floyd, "On inferring tcp behavior," *ACM SIGCOMM Computer Communication*, 2001.

[16] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, "Tcp congestion avoidance algorithm identification," in *2011 31st International Conference on Distributed Computing Systems*, 2011, pp. 310–321.

[17] I. K. Jan Rüth and O. Hohlfeld, "Tcp's initial window—deployment in the wild and its impact on performance," *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*, June 2019.

[18] P. Balasubramanian, Y. Huang, and M. Olson, "HyStart++: Modified Slow Start for TCP," Internet Engineering Task Force, Internet-Draft draft-ietf-tcpm-hystartplusplus-13, Jan. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tcpm-hystartplusplus/13/

[19] W. Stevens, "Tcp slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *RFC 2001*, January 1997.

[20] Z. Tlaiss, I. Hamchaoui, I. Amigo, A. Ferrieux, and S. Vaton, "Troubleshooting enhancement with automated slow-start detection," in *2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2023, pp. 129–136.

[21] E. Casey. (2010) Handbook of digital forensics and investigation. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/tcpdump

[22] I. S. I. U. of Southern California, "Transmission control protocol," *RFC 793*, Sep 1981.

[23] Y.-J. Song, G.-H. Kim, I. Mahmud, W.-K. Seo, and Y.-Z. Cho, "Understanding of bbrv2: Evaluation and comparison with bbrv1 congestion control algorithm," *IEEE Access*, vol. PP, pp. 1–1, 02 2021.