Automated slow-start detection for anomaly root cause analysis and BBR identification

Ziad Tlaiss, Alexandre Ferrieux, Isabel Amigo, Isabelle Hamchaoui, Sandrine Vaton

Automated Slow-Start Detection for Anomaly Root Cause analysis and BBR Identification

Ziad Tlaiss[1], Alexandre Ferrieux[2], Isabel Amigo[1], Isabelle Hamchaoui[2], Sandrine Vaton[1]

[1]IMT Atlantique, Lab-STICC laboratory, Brest, France.
[2]Orange Labs Networks, Lannion, France.

Contributing authors: ziad.tlaiss@imt-atlantique.fr; alexandre.ferrieux@orange.com; isabel.amigo@imt-atlantique.fr; isabelle.hamchaoui@orange.com; sandrine.vaton@imt-atlantique.fr;

Abstract

Networks troubleshooting usually requires packet level traffic capturing and analysing. Indeed, the observation of emission patterns sheds some light on the kind of degradation experienced by a connection. In the case of reliable transport traffic where congestion control is performed, such as TCP and QUIC traffic, these patterns are the fruit of decisions made by the Congestion Control Algorithm (CCA), according to its own perception of network conditions. The CCA estimates the bottleneck's capacity via an exponential probing, during the so-called "Slow-Start" (SS) state. The bottleneck is considered as reached upon reception of congestion signs, typically lost packets or abnormal packet delays depending on the version of CCA used. The SS state duration is thus a key indicator for the diagnosis of faults; this indicator is estimated empirically by human experts today, which is time-consuming and a cumbersome task with large error margins. This paper proposes a method to automatically identify the Slow-Start state from actively and passively obtained bidirectional packet traces. It relies on an innovative timeless representation of the observed packets series. We implemented our method in our active and passive probes and tested it with CUBIC and BBR under different network conditions. We then picked a few real-life examples to illustrate the value of our representation for easy discrimination between typical faults and for identifying BBR among CCAs variants.

Keywords: troubleshooting, active measurement, passive measurement, Congestion Control algorithm, Slow-Start state, packet inter-arrival times, BBR

1 Introduction

Quality of Experience (QoE) remains one of the most crucial competitive advantages for an Internet Service Provider as it directly impacts its brand image. However, room for improvement remains consistent for network operators as many networks are still impeded by crippling issues bitterly reported by customers which affect operators reputation. Improving customers experience is a necessary yet delicate task, as it relies on continuous and pervasive monitoring of the network and, as soon as a degradation is detected, a quick identification and fixing of the root cause is demanded, that is troubleshooting.

Unfortunately many problems cannot be handled with mainstream monitoring tools: For example, excessive latency cannot be detected via flow-level tools such as Netflow [1], nor by equipment's counters [2] or traffic sampling methods [3]. Even if end-to-end active measurements can identify slow connections, they are of little help for locating latency bottlenecks on the path. Collecting and analysing exhaustive packet-level traces captured on network mid-points are still operators' best choice for anomaly root cause identification.

For decades, most operators' end-to-end diagnosis methods have been based on the observation of transport protocol (e.g. TCP, Transmission Control Protocol) packet headers. Hence, with a mere one-point traffic capture, passive probes can easily catch latencies and packet losses on both upstream (from sender to probe) and downstream (from probe to destination) segments, see [4] and [5]. Faulty nodes can further be located by moving the capture point (beam splitter, port mirroring, etc.). These captures together with the related analysis are typically performed by human experts upon network monitoring alarms or customers claims. This method remains the cornerstone of troubleshooting for many operators, however, it is jeopardized by the explosive growth of QUIC where transport headers are encrypted, making them unreadable for a midpoint observer. Although QUIC has an option that allows monitoring latencies [6], it is likely that it will not be implemented by client software (e.g. browsers, mobile applications) in the future. QUIC observability via passive probes will thus probably be lacking for a long time.

It is true that active probes, by generating their own traffic, are not impeded by encryption and can detect QoS degradation, however, contrarily to passive probes, their representativeness can be questioned for two reasons: First, test traffic is not real clients' traffic. Second, active probes typically see only a subset of the network. This last point can be balanced through a massive probes deployment, but as current troubleshooting is mainly based on human diagnosis, automation is certainly a key element for dealing with the data deluge collected via such a dense fleet of active probes. Beyond this scalability issue, the whole troubleshooting process should be revisited in light of active measurement specificity.

In this context, observing the source behavior appears as a promising strategy: Source emission patterns derive from decisions of the Congestion Control Algorithm (CCA) embedded in TCP/QUIC stacks, and reflect network conditions rather accurately. The CCA is in charge of regulating the source emission to obtain a good throughput without flooding the network. It calms down as soon as it detects early signs of congestion, such as packet loss or delay. Tracking its behavior thus reveals

crucial hints for fault qualification and location. Loss-based algorithms like CUBIC can lead to reduced throughput for users [7]. This can be addressed by investing in Data Center Networks (DCN) to ensure sufficient bandwidth and speed, and minimize packet loss rates. On the other hand, recent CCAs as BBR (Bottleneck Bandwidth and Round-trip propagation time) offers improved performance, lower latency, and better resource utilization, reducing the need for additional resources to support the same number of users or applications [8]. For network operators, understanding the CCAs in use on their network is crucial for identifying bottlenecks, optimizing network performance, and making informed decisions on network design, configuration, and management. Identifying the type of CCA in use, particularly BBR, can help optimize network infrastructure and reduce investment in network infrastructure.

Slow-Start (SS) is an important state of CCA as it aims at giving a first estimation of the path capacity at the beginning of the connection life, or on resumption after a significant pause. Should it fail, then the connection will experience poor quality. Most importantly, the way it fails (with or without loss, etc.) is an excellent indication of the degradation root cause. In this work, we introduce an effective method to automatically detect the exit from SS state - or equivalent naming. For this purpose, we introduce a novel representation, that is the bytes-in-flight versus sequence number. We use then this representation in order to identify the last packet in the SS state of the CCA, by using a relation between the sequence number values and the bytes-in-flight values that are true only during the SS phase. Due to the similarity of QUIC and TCP CCAs, this method applies to both. However, as it requires accessing transport headers information, it can be applied to QUIC traffic **only** with active measurements while it could be used on TCP traffic with both active and passive measurements. We also introduce how our SS detection method could be used as a powerful tool to easily discriminate between network typical fault types and to identify BBR CCA among TCP traffic.

The remainder of this paper is organized as follows. In Section 2, we demonstrate the significance of CCA behavior for troubleshooting. In Section 3, we describe the troubleshooting process from data collection to data analysis for root cause identification. Section 4 presents our method to detect SS state exit. An evaluation of our method is presented in section 5. An application of the method to network troubleshooting is presented in section 6. A model based on analysing packet arrival time during the SS to identify TCP BBR is introduced in section 7. Section 8 surveys the related work. Finally, a conclusion is given in Section 9.

We note that this paper is an extended version of a paper that we submitted to ICIN 2023 conference [9].

2 Congestion Control Algorithm and troubleshooting

2.1 Finite State Machine transitions series

The CCA role is to prevent congestion collapse while taking into account fairness and improving connection's performance [10]. Roughly, the CCA embedded in TCP/QUIC

stacks aims at reaching, in a fair manner, the highest throughput safely tolerable by the network. Under its control, the emitted traffic falls back as soon as it detects signs of congestion, that is, lost packets or growing delays. This behaviour can be observed with an exhaustive capture of the flow's packets, but also, via the sequence of transitions of the CCA Finite State Machine (FSM). State transitions are typically triggered by degradation events such as detection of congestion signals. The FSM transitions series contains factually rich semantic information providing crucial elements for troubleshooting.

2.2 Invariants in CCA behaviour

Amongst the CCAs, the most commonly encountered are CUBIC and BBR. According to [11], they alone contribute to the vast majority of today's traffic, in particular, BBR traffic probably represents more than half of all internet traffic. Other CCAs can be observed, but in significantly lower proportion. However, the CCA landscape remains diverse as many flavours coexist for a given CCA, depending of the underlying implementation (e.g. Linux version on server side).

Even if these CCAs differ in major ways, they share a few common mechanisms, particularly at the beginning of the connection life. Should it be called "Slow Start" or whatever equivalent (e.g. Hystart or Hystart++, see [12]), the CCA behaviour in the first state is the same: an exponential rate growth until reaching the bottleneck capacity. An example of this exponential growth is shown in Figure 1 and Figure 2 for a CUBIC and BBR capture.

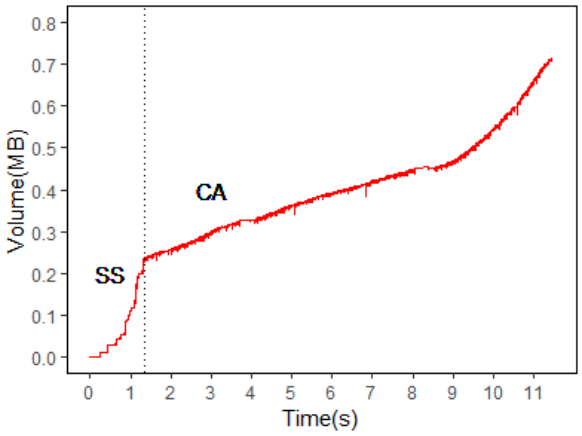

Fig. 1 BIF (Bytes In Flight) over time evolution for a CUBIC capture showing SS & CA states, we can notice the exponential growth of the SS phase versus the linear growth during the CA phase

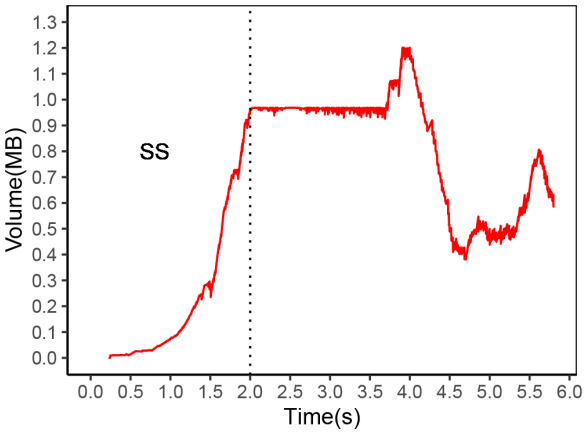

Fig. 2 BIF over time evolution for a BBR capture showing SS state, we can notice the exponential growth of the SS phase versus the linear growth after

2.3 The (not so) Slow Start

Generally speaking, the CCA controls the amount of data being injected into the network. For this purpose, the sender typically maintains a variable called Congestion Window (cwnd) that determines the amount of data that can be transmitted before receiving an acknowledgement from the destination [13]. This cwnd is a variable internal to the sender stack and then is unknown from any mid-point observer, or from the destination. However, it can be inferred from observation of the Bytes-in-Flight (BIF), i.e. the number of bytes sent but not yet acknowledged, as the BIF is, by definition, strictly bounded by the cwnd value at any given time.

During the SS, the CCA exponentially increases its Congestion Window (or similarly, its emission rate) to quickly reach the bottleneck capacity. The very first packets are emitted with respect to a so-called Initial Congestion Window, typically 10 packets [14]. Then, the sender keeps doubling the congestion window value every round trip time until a congestion signal is detected or a threshold (ssthresh) is reached [15]. This congestion signal is typically a loss for CUBIC or an excessive delay for BBR [16]. Figure 1 shows the cwnd exponential growth during the SS state ending at around 1.4 sec before entering the Congestion Avoidance (CA) state.

2.4 Slow-Start exit time

When analysing a packet trace, the SS exit time is a key element for troubleshooting experts. If many state transitions suggest specific root causes, SS exit time has a particular significance in detecting faults type and identify the used CCA variant. Indeed, in SS, the source estimates the value of the path's capacity by exponentially increasing its rate (binary search) until a congestion signal is received, then it exits the SS state to enter a new phase with a much lower rate growth.

Should the SS overestimate the bottleneck, then the source will exceed the bottleneck capacity and thus experience multiple packet losses, from which recovery is

painful. On the contrary, if it underestimates the bottleneck capacity and triggers an early SS exit, the source will under-use the available bandwidth and possibly experience a poor throughput. While bottleneck overestimation is quite unusual, underestimation is a very common cause for low performance. It simply reveals that the SS has mistakenly detected a congestion signal. This is typically the case in presence of transmission loss, or excessive jitter related to radio mobile access - even underloaded. In both cases, limiting the rate will lead to a pitiful customer experience, without any benefit regarding a non-existent congestion. For example, as shown in Figure 1, the CCA exits SS after around 1.4 sec with a cwnd value around 0.2 MB, however, we can notice that the true bottleneck capacity is around 0.7 MB as the cwnd value keeps slowly increasing until reaching it. As the trigger signal is not obvious, this early exit should be investigated.

3 Troubleshooting: from data extraction to data analysis

In this section we present the metrics used for network troubleshooting, together with associated basic data processing.

3.1 Data collection & Data processing

Capturing Packet traces can be made via many tools, such as Wireshark [17] and tcpdump [18]. These tools capture transport layer packet headers together with their arrival times as shown in Figure 3.

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

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

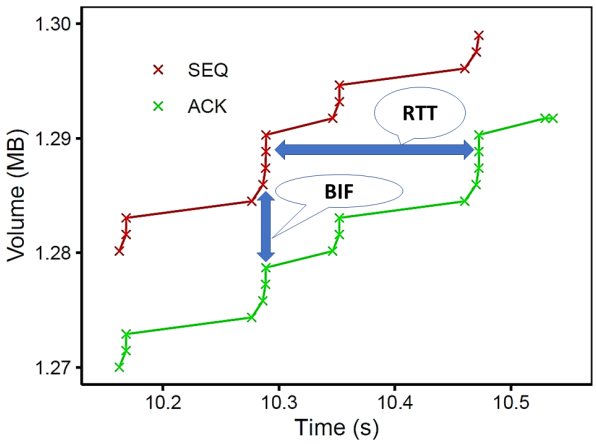

Fig. 4 BIF and RTT calculation method

These captures should be processed, so as to derive significant timestamped indicators; The most significant ones are:

- Sequence number (SEQ): identifies the first byte in a segment [19]. For a better match up with the acknowledgements, we denote SEQ the last byte of the transmitted segment plus one, i.e. $SEQ = sequence\,number + length$.
- Acknowledgment (ACK): it informs the source about the sequence number of the next expected segment.
- Receiving window (RWIN) : it identifies the number of bytes that the receiver can accept.
- Bytes in flight (BIF): it represents the number of 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 and Figure 3.
- Round-Trip-Time (RTT): it represents the delay between the emission of a packet and the reception of the corresponding acknowledgment. It can be calculated using the SEQ and ACK arrival times as shown in Figure 4.
- Packet inter-arrival times (INTRPKT): it represents the elapsed time between the arrival of two consecutive packets.

3.2 Data analysis

This final step typically consists in visually analyzing the temporal evolution of these indicators, e.g. with a tool as tcptrace [20]. For each trace, a human expert should visually detect the state transitions and QoS degradations affecting the connection a.

To automate the analysis, we designed a method and developed a tool to automatically detect the SS exit time on collected traces. Combined with other indicators (presence of loss, etc.), this is a significant step toward full automation.

4 Automatic Slow Start exit detection

In this section we introduce a new representation together with a method to automatically detect SS exit.[1]

4.1 Visual CCA states identification

As explained in section 2, the FSM state series, and particularly the SS exit time gives crucial insight for troubleshooting experts. To get hold of these state series, the first idea that comes to mind is direct introspection in the sender stack. Unfortunately, this introspection requires cooperation from the sender's server, which is rather impractical, as many servers belong to third-party internet content providers, often reluctant to open their infrastructures. As a consequence, sticking to measurements from active and passive probes is still operators' best choice to build these state series. Recall that passive probes can only handle TCP traffic, as active probes may monitor both TCP or QUIC flows.

In this context, troubleshooting experts are used to performing visual analysis of the BIF against time to detect the end of the exponential growth, namely the SS exit time. This method is highly time consuming and inaccurate. We can see in Figure 5 an exponential increase of the BIF against time until $t = 1.2$ sec, a telltale sign of the SS phase. Root cause analysis is then completed thanks to the SEQ against time graph (Figure 6) showing bursts of packets retransmission at this very same time, a typical effect of congestion loss [21]. The SS exit is then a legitimate reaction of the CCA to reaching the actual bottleneck.

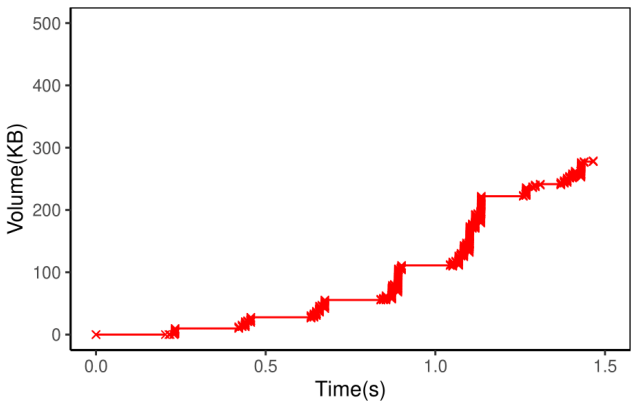

Fig. 5 BIF against time during SS state, focus on the exponential growth

[1] Tool available online at https://193.252.113.227/cgi-bin/ats.cgi

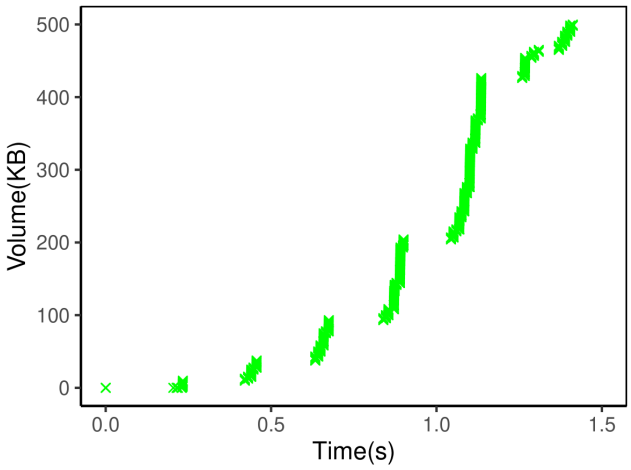

Fig. 6 SEQ against time during SS state - packet burst doubled after each RTT

4.2 Challenges towards automation: Noise and Non-stationarity

While manual analysis can easily handle noise and irregularities in the data, this is non trivial for an automated procedure. For example, some smoothing may be necessary in order to recognize the exponential growth in Figures 5 and 6. More generally, the graphs might need to be segmented into a number of homogeneous regimes before any kind of pattern recognition can be applied.

The aforementioned time series are typically non-stationary, and not even piecewise stationary. This makes it impossible to define homogeneous areas, which invalidates most usual mathematical methods. An example about this non-stationarity can be observed in Figure 7 and Figure 8. Figure 7 shows the evolution of BIF over time, while Figure 8 represents the arrival times of the captured packets. The latter focuses on the so-called "on/off pattern" of packets arrival times which reflects the basic congestion window mechanism, waiting for ACKs before sending a new burst of packets. However, while this on/off pattern can be detected at the beginning of the connection (from 0 sec to 0.8 sec), it blurs over time, due to TCP's (intentional) tendency towards "ACK clocking" [13].

All previously mentioned challenges invalidate most methods like using regression and Markov Modulated Poisson Process (MMPP) that we have considered and tried when working on the automated detection of the SS state. A more promising approach yielding better initial results was "exponential regression", i.e. fitting the BIF-against-time with an exponential. However, it turns out that in case of *very* early SS exits (within 2 or 3 RTT), the exponential part is dwarfed by the subsequent evolution, making it impossible to detect the exponential part reliably. This is unfortunate, as we use the SS state detection to troubleshoot networks, where the most frequent cases of bad performance are correlated with a premature exit from SS. As it turns out,

this fundamental problem is resolved using the new representation that we introduce in the next section.

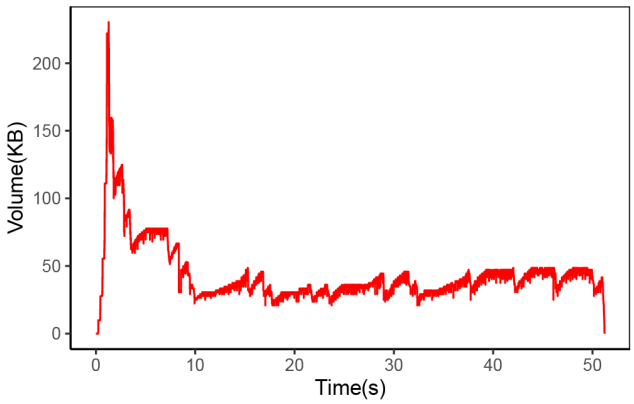

Fig. 7 BIF against time - non stationarity and noise on BIF values

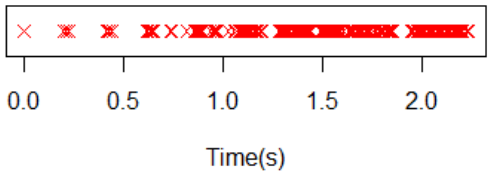

Fig. 8 Packets arrival times - blurring of on/off pattern over time

4.3 Timeless packet series representation

In the light of our troubleshooting experience, it turns out that the main hurdle to automation lies in the on/off patterns of the source emissions. A natural way to get rid of them without any loss of information is to switch to a *timeless* representation. To this effect, we chose to represent BIF as a function of SEQ as shown in Figure 9. In essence, we replace the time axis with the sequence number progression: this naturally wipes out all burst, silence, or RTT variation effects, while preserving the important correlations between significant indicators, thus focusing on the CCA dynamics.
To the best of our knowledge, such a representation was not described before in the state of the art.

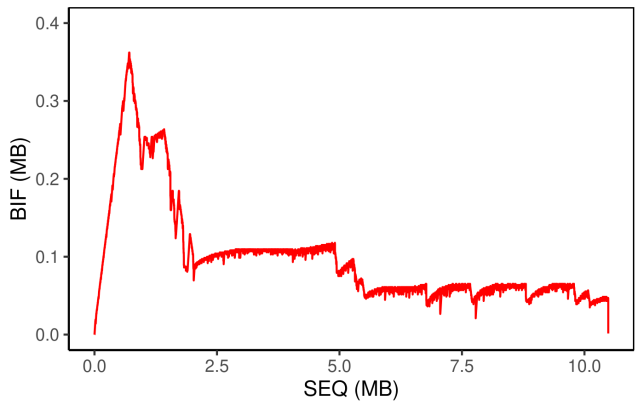

Fig. 9 BIF against SEQ: new representation to detect the SS state exit time

4.4 Slow Start exit time: "slope 1/2" method

A few basic properties of the BIF vs SEQ representation can easily be derived analytically. To begin with, the shape of the graph is readily predictable during 2 phases:

(a) During burst emissions: in the absence of any acknowledgement, during this phase each sent packet increments both SEQ and BIF by an equal value, which is the segment's length.

(b) During the reception of burst acknowledgements: assuming all packets previously sent were received, the BIF quickly drops back to zero.

As a result of these 2 phases, every round-trip time, the graph is expected to display a triangular shape made of a slope 1 due to phase (a), followed by the vertical drop described in (b), as depicted in Figure 10.

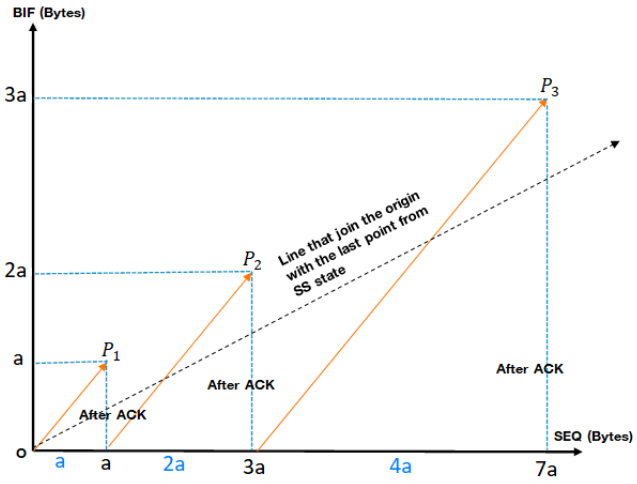

Fig. 10 Theoretical Representation of the BIF vs SEQ evolution

Furthermore, in an ideal SS state, the vertical extent of this triangular shape, which represents the cwnd, is expected to double every round-trip-time. Thus, the graph should display a *fractal* series of triangles, each one being twice the size of the one before. The position of the highest-SEQ point and highest-BIF point in the graph, after n round-trip-times, is thus expected to be:

$$SEQ = \sum_{i=0}^{n-1} a \times 2^i = a \times (2^n - 1)$$

$$BIF = a \times 2^{n-1}$$

The slope of the line from origin to this point is thus

$$BIF/SEQ = \frac{2^{n-1}}{2^n - 1}$$

And hence its limit

$$\lim_{n \to +\infty} BIF/SEQ = 1/2$$

It can further be seen that this asymptote $y = x/2$ is in fact "approached from above", as the top of each triangle satisfies.

$$BIF/SEQ = \frac{2^{n-1}}{2^n - 1} > 1/2$$

However, as soon as the SS state is exited, the exponential growth of the BIF stops, and no further point can stand above the $y = x/2$ line. This yields a very simple and

practical criterion: the SS exit occurs immediately after the last point satisfying

$$BIF \geq \frac{SEQ}{2}$$

It should be stressed that the power of this method lies in its simplicity: no regression neither filtering are needed, a simple linear inequality suffices, once we are in the appropriate representation space.

4.5 Slope 1/2 method details

While the critical state transition event is well characterized by the above criterion, some attention is due to properly interpret the earlier features of the representation. During the SS phase, as mentioned before, local slopes are typically 1, with a series of abrupt drops. As a result, the graph keeps crossing the asymptote, thus, a local decision is not appropriate, as it would readily generate false positives. Fortunately, the *global* criterion of the "last point above the asymptote" is more robust. This is fundamentally linked to the fact that after exiting SS, the CCA essentially takes very careful steps to refrain from going too fast, and by definition will never "catch up" to the exponential regime. The asymptote is never to be crossed again. Figure 11 is an example of our slope 1/2 method application. We can see the BIF vs SEQ curve slightly exceeding the $y = \frac{1}{2}x$ line until it abruptly drifts below, marking the instant when the CCA has exited the SS state.

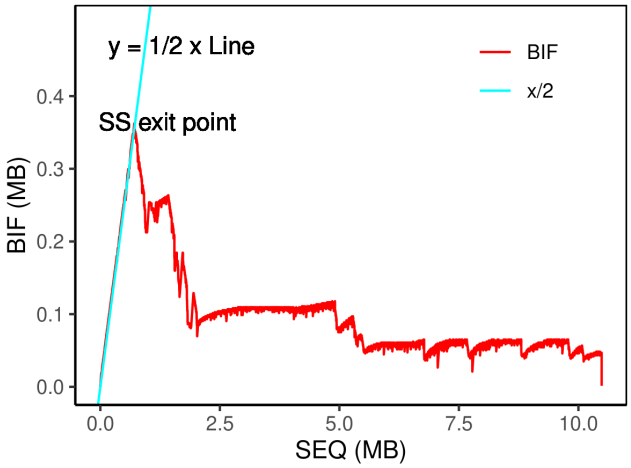

Fig. 11 Automatically detecting SS state exit time with slope 1/2 method

5 Evaluation of the slope 1/2 method

In this section we assess the accuracy of our method by comparing the SS exit time we obtain, against a "ground truth" which we define as the CCA state transition time

recorded in the server stack logs. For this purpose, the assessment is performed on one of our servers, accessed by several active probes, through the public internet. We instrumented the CUBIC and BBR TCP stacks on this server to generate CCA logs; then we performed active measurements with our probes by executing many downloads from our server using BBR and CUBIC CCA, and from these logs, we extracted the "ground truth" SS exit times. We note that in the Linux BBR implementation that we instrumented, the SS (binary search period) is very difficult to identify, especially in presence of massive loss, as it is not a discrete state but a region in parameter space.

For the sake of representativity, we locate our probes amongst 4 Orange affiliates. Depending on the country, the average RTT ranges from 20ms to 200ms with varied loss levels. Moreover, in each of these countries, our downloads were performed with various congestion levels (peak and off-peak hours). Last, we compute the difference between the *slope 1/2* exit time and this "ground truth". This time difference is represented in Figures 12 and 13, in RTT units.

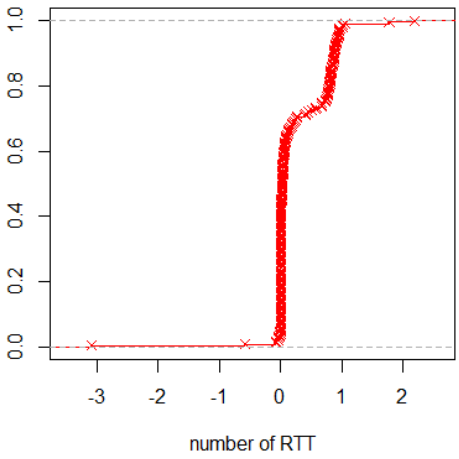

Fig. 12 Cumulative distribution function of the difference in RTT units between the SS exit times from the slope 1/2 method and the one logged by the CUBIC server

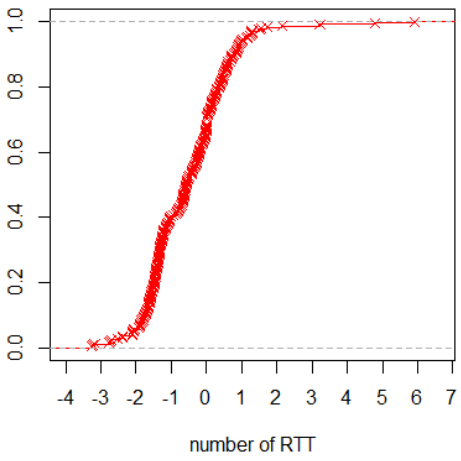

Fig. 13 Cumulative distribution function of the difference in RTT units between the SS exit times from the slope 1/2 method and the one logged by the BBR server

Figure 12 shows the distribution of error in prediction for the CUBIC stack on 219 downloads. We can notice that this error is less than 1 RTT in more than 95% of cases. It is indeed the best accuracy that can be expected, since the typical time granularity of CCA decisions is precisely the RTT.

The same representation for BBR is shown in Figure 13 for 241 downloads. We can notice that 55 % of cases are bounded between -1 and 1 RTT. We attribute this larger deviation in prediction error more to the approximate ground truth (because of the aforementioned difficulties in the BBR implementation) than to real mis-predictions of our slope 1/2 method. We are currently investigating a better calculation for ground truth.

6 Application to network troubleshooting

The slope 1/2 method, together with our new BIF/SEQ representation, proves to be a powerful tool in network troubleshooting. Indeed, with this representation, typical patterns appear, each of them pointing to a type of degradation. What is more, these graphical patterns could rather easily be identified using trivial criteria, thus leading to easy classification. We describe hereafter some typical faults together with their graphical representation.

6.1 Pattern 1: Constant BIF after SS exit

In this ideal case, the binary search performed during the SS successfully discovers the path bottleneck. Then the $BIF = f(SEQ)$ curve stays constant, as shown in figure 14. This denotes a good and stable QoS.

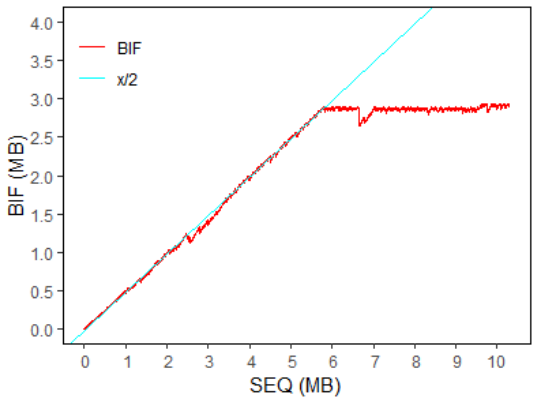

Fig. 14 Constant BIF after SS exit: Good QoS

6.2 Pattern 2: BIF dive after SS exit

Figure 15 exhibits a sudden drop in BIF value after what we identify as the SS exit time; then, the BIF value further stays significantly lower than its peak at SS exit. This pattern is typically associated with QoS degradation due to loss. These losses may originate from cross-traffic competition (i.e. traffic emitted by other sources) or transmission errors.

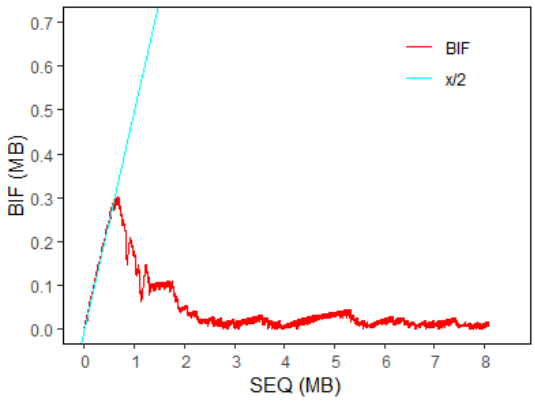

Fig. 15 BIF plunge after SS exit: Loss

6.3 Pattern 3: BIF growth after SS exit

Beyond losses, packet delay variation (also known as jitter) is another typical root cause for early SS exits [12], as they may occur while the bottleneck capacity has not been reached yet. Figure 16 illustrates this case. In contrast with pattern 1, the BIF here continues to grow after SS exit, but at a much slower pace. This allows for an easy discrimination between these two patterns. Pattern 3 also clearly differs from pattern

2, as it exhibits no reduction in BIF, that is, no packet loss. This behaviour is typical of mobile access networks affected with large jitter values.

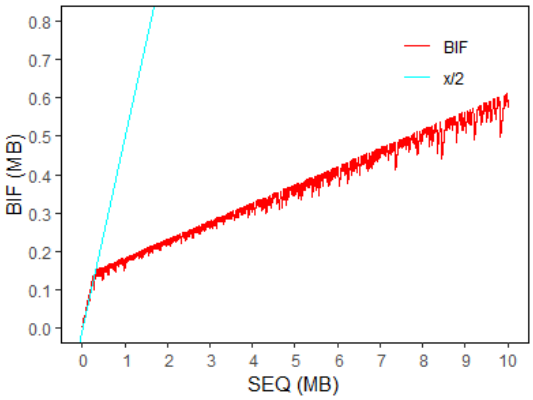

Fig. 16 Slow Bif growth after SS exit: Jitter issue

7 BBR identification using the SS exit time detection

BBR and CUBIC are two popular congestion control algorithms representing together the main part of internet volume [11]. As their behaviour is particularly contrasted in presence of loss or other degradation, it is crucial to differentiate them when investigations are conducted. Indeed, erroneous conclusions can easily be drawn in case of mis-identification of the CCA flavour.

Despite abundant literature on their respective strengths and weaknesses, little has been proposed on solid discrimination tools (see Section 8). In the present section, we propose an identification method based on their different behaviour during the SS state.

7.1 Motivation of the method

By design, of the two end-points of a connection, only the source has an explicit knowledge of the running CCA flavour (CUBIC or BBR). As a consequence, in this paper we work with traces from controlled sources (i.e. our lab servers) to calibrate our recognition algorithm. Extension of this method to real life traffic from third-party servers will be considered in section 7.4.

From the mere observation of traces, it can be seen that at the beginning of the connection, BBR and CUBIC emission patterns are particularly dissimilar (Figure 17): CUBIC is typically bursty whereas BBR exhibits a smooth emission pattern. This results from a fundamental difference between these two CCAs: BBR is rate-based while CUBIC is window-based. In other words, CUBIC sends its full window in a burst and waits for acknowledgements, while BBR paces its emissions according to a rate-based policy.

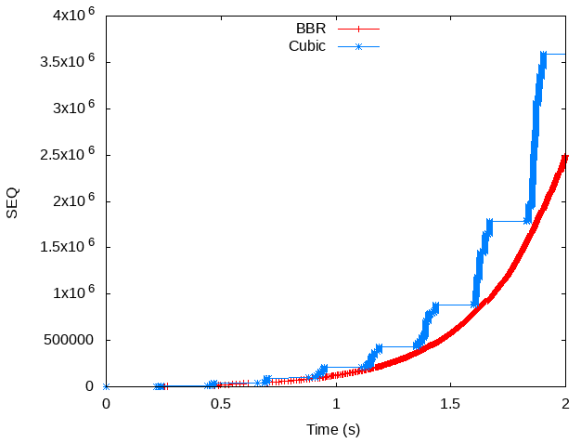

Fig. 17 Time-sequence graph of real CUBIC and BBR connections during SS

However, after SS exit, CUBIC also tends to behave smoothly, as it is subject to the so-called "ack clocking" phenomenon [13], due to bottleneck-induced pacing, as shown in Figure 18. This precisely happens when reaching the bottleneck capacity, typically on SS exit.

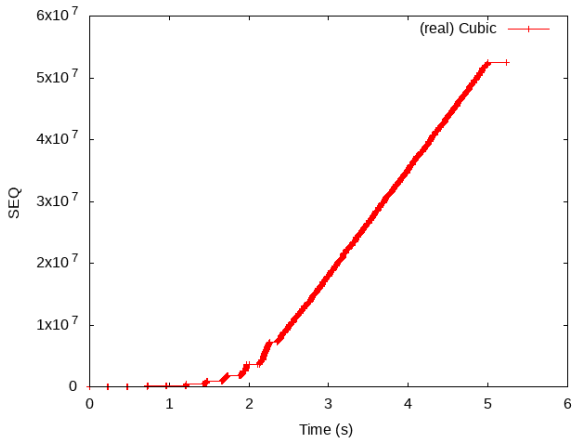

Fig. 18 SEQ against time for a CUBIC capture - we can notice how CUBIC tends to behave smoothly after exiting the SS state.

In a nutshell, the best period for discrimination appears to be during the SS. As a consequence, we isolate the SS period of each connection and characterize the emission patterns - more precisely, the burstiness - in this period only. To assess it, our approach relies on analyzing the distribution of packet inter-arrival times (INTRPKT), derived from the packet capture.

7.2 Analyzing Packet Inter-Arrival Times: Modelling and Inference

To catch the statistical properties of their INTRPKT distribution, we plot on Figure 19 the probability density functions (pdf) for a BBR (red) vs. CUBIC (blue) connection; they exhibit a significantly different shape: Contrary to BBR, CUBIC distribution is typically heavy-tailed.

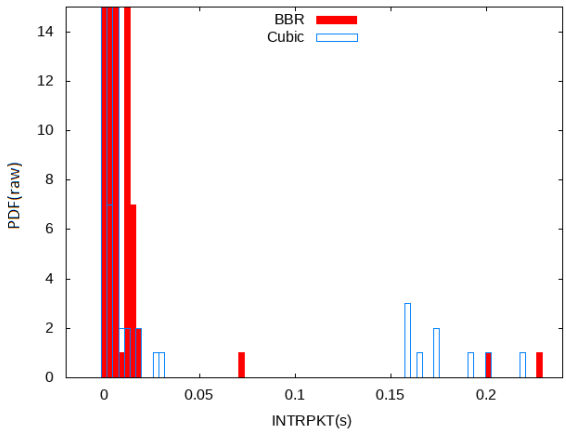

Fig. 19 The PDF of INTRPKT for a CUBIC and BBR capture

7.2.1 CUBIC distribution

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

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

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

7.2.2 BBR distribution

On the other hand, a typical BBR distribution only exhibits a peak on small values. Indeed, occurrences of large INTRPKT are pretty rare, due to smooth emission pattern. The discrimination task then seems to amount to recognizing single-peak (BBR) vs. heavy-tailed (CUBIC) distributions.

7.2.3 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 large number of events in a burst, compared with the small number of pauses between bursts. Since our aim is to discriminate between single-peak and heavy-tailed distributions, we need to compensate this imbalance by magnifying the minority contribution, namely the long-INTRPK component.

Note that, as 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 $\hat{f}_{bal}(x) = \sum_i x_i \mathbb{1}_{x_i}(x)$.

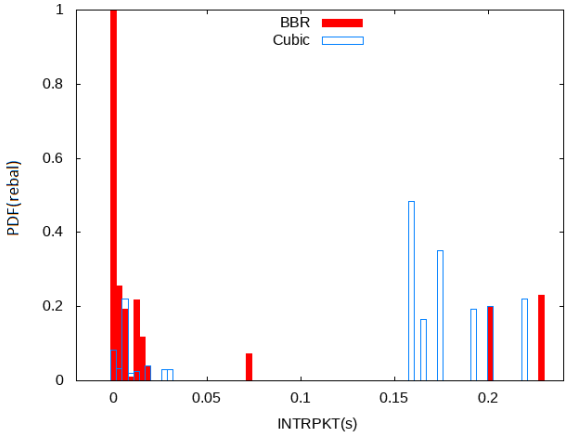

Fig. 20 The PDF of INTRPKT for a CUBIC and BBR capture after expanding the small contributions

It is interesting to notice that the resulting empirical distribution $\hat{f}_{bal}$ approximates the distribution of INTRPKTs that would be observed if sampling was uniform over time (Figure 20) rather than uniform over packets (Figure 19). Choosing each sampling instant uniformly over the observation period corresponds to a Poisson sampling. Because of the PASTA (Poisson Arrivals See Time Averages) property, the probability of a sampling instant falling 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). Replacing a uniform sampling strategy across packets with a uniform sampling strategy over time gives more weight to the inter-packets which correspond to silences between two bursts. This is an interesting property because it makes it possible to better differentiate the distribution of INTRPKT between CUBIC and BBR.

7.3 Choice of the optimal decision point

To better capture distribution features in a resolution-independent manner, we switch from PDFs to CDFs. Figures 21 and 22 respectively show the raw and re-balanced CDFs of INTRPKTs during the SS state of CUBIC and BBR. We see that in the rebalanced case (which mimics an uniform sampling over time), the two distributions are much better separated than in the raw case (corresponding to the original uniform sampling over packets).

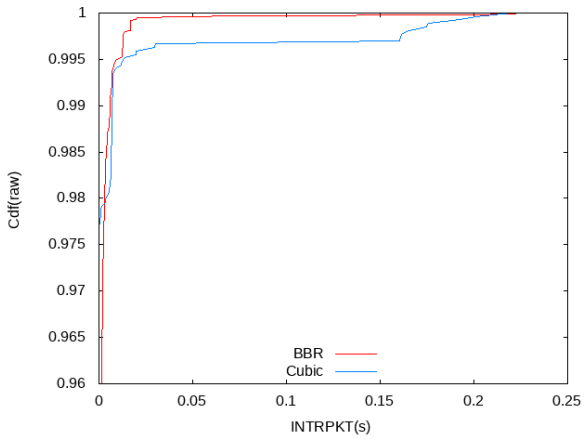

Fig. 21 CDF of INTRPKT for a CUBIC and BBR capture using the raw distribution (uniform sampling over packets)

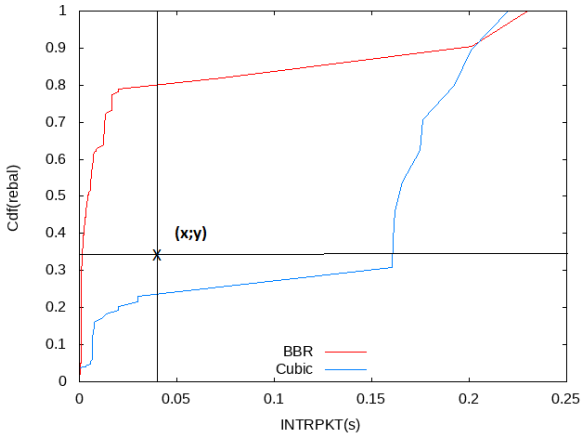

Fig. 22 CDF of INTRPKT for the same captures using the re-balanced distribution (uniform sampling over time). A typical decision point $(x; y)$ is shown.

Let us assume that our objective is to determine whether a connection is carrying BBR traffic or not. To accomplish this, we can frame the problem as an hypothesis test between two options: $H_0 : CUBIC$ and $H_1 : BBR$. During the SS state, we measure INTRPKT values $x_1, x_2, \ldots, x_T$, and propose a method that makes a decision based on these values.

Since CUBIC tends to have more long INTRPKTs, we decide that the connection is using CUBIC if the proportion of values $(x_i)_{i=1,T}$ greater than a threshold x is greater than y. Conversely, if the proportion of values $(x_i)_{i=1,T}$ greater than x is smaller than y, then we conclude that the connection is using BBR.

In other words, we are seeking a point (x,y) that separates the red and blue curves in Figure 22. If the curve $(x, \tilde{f}(x))$ (Eq. 3) is below this point, we decide CUBIC; if it is above the point, we decide BBR. Since there are two types of risk in a hypothesis test between two options, false alarm and non-detection, we propose to fix $y = \theta(x)$ for a particular value of x, so that the two risks have equal probability. This roughly corresponds to positioning the point (x, y) in the middle, between the green and red curves in Figure 22 (assuming equal numbers of CUBIC and BBR connections). We then select the x minimizing the total (false-alarm + non-detection) probability of error, which is the probability of misclassifying a connection.

7.4 Model evaluation

We assess our method and model using two datasets, consisting of 221 captures for training and 583 for evaluation.

The packet traces were captured on one of our servers, which was accessed by multiple active probes via the public Internet. The active 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 different countries. In

addition, our downloads were carried out during peak and off-peak hours, seeking different levels of congestion. The average RTT of our captures varies between 100ms and 400ms, while the majority fall between 200ms and 300ms.

7.4.1 Choice of the decision point using a training dataset

In order to identify the optimal decision point, we trained on a dataset of 221 packet captures, consisting of 133 BBR captures and 88 CUBIC captures. By applying the equal-error-rate minimization described above to the CDF curves of these packet captures, depicted in Figure 23, we determined that the decision point should be set at $(x = 0.14, y = 0.503)$: in other words, the optimal decision criterion amounts to comparing the median of the INTERPKT distribution with $0.14 * RTT$.

Using this decision point, the minimum total error rate achieved on the training dataset is 2.6% . As the optimum is degenerate (several points with the same error rate), we selected the central one to get as a wide a margin as possible.

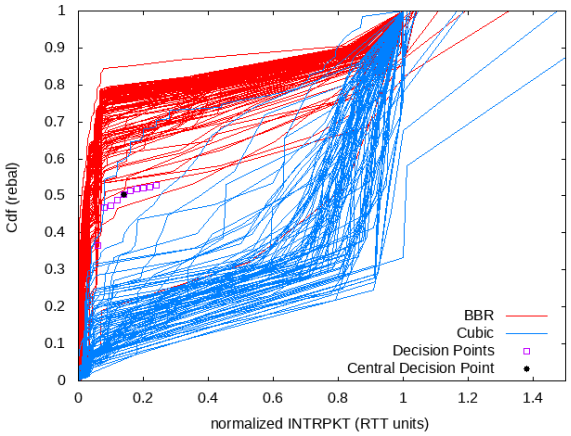

Fig. 23 Choice of the decision point on the training set

7.4.2 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 CDF curves of all 583 captures are presented in Figure 24. By using the decision point we obtained on the training dataset, the model is able to identify TCP variants with an overall error rate of only 4.1%. None of the 194 CUBIC captures is misclassified, and 16 out of the 389 BBR captures are mistakenly classified as CUBIC. This slight bias points to the necessity of a larger and more diverse training set, slated for future work.

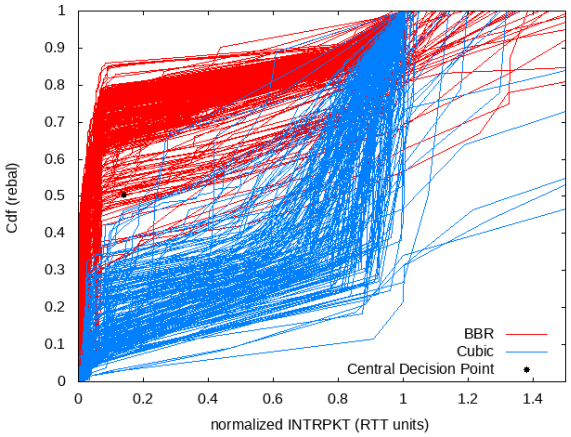

Fig. 24 Model evaluation on an independent test set

7.5 Model advantages and future challenges

By taking advantage of the novel SS-detection method described in the earlier sections of this paper, it is possible to build 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 there are various obstacles that 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, 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.

8 Related Work

We identified two areas of related work to our study: Inferring and identification of the CCA and network troubleshooting tools. In this section, we focus on those directly related to our work in each area.

8.1 Identification of CCA states to infer TCP behavior

Hagos et al. [22] use machine learning approaches to recognize loss-based TCP CCAs and infer the congestion window within a passively collected traffic at mid-point.

Although estimating the cwnd can be useful for network operators to troubleshoot their network, it does not cover non-loss-based CCAs such BBR. Our work differs from theirs as our method focuses on the application of CCA SS state detection in order to detect network root causes anomalies, and could be applied for all types of CCA, loss-based or not.

Padhye et al. [23] 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. [24] 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.

Jaiswal et al. [25] introduce a passive measurement methodology to infer the cwnd and round-trip-time. They build a replica of the CCA state for each TCP connection at the midpoint. This replica updates its estimate of the cwnd based on the observed acknowledgments that could change the CCA state. They use those estimates to recognize 3 of the TCP flavors: Reno, NewReno and Tahoe. Even if [25] are interested in initial cwnd, SS state and congestion avoidance states to identify the CCA types, they do not try to accurately locate state transitions.

Kato et al. [26] use unidirectional packet traces to characterise TCP CCAs. They define a new metric that is seen as being proportional to cwnd size, and apply curve fitting to recognize the CCA. In the continuity of their work Kato et al. [27] identify TCP CCAs using a sequence number vs packet arrival time representation.

Zhang et al. [28] analyse TCP passive packet captures and investigate CCA mechanisms to understand the origins of limitation in the transmission rates of flows by grouping packets into flights using a round-trip-time estimator.

Mishra et al. [11] 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.

In summary, [28], [24] and [11] show interest in the detection of the SS state; however, [28]'s method tracks the SS state in the first flights based on explicit segmentation, which does not work consistently in real life, e.g. when ACKs are not "bursty". On the other hand, [24] and [11] only take losses into consideration in the detection of the SS state. In contrast, in our work, while we do end up using the cause of SS exit to identify the root cause of an anomaly, we start by locating the event regardless of the cause.

On the other hand, none of the previously mentioned works address the identification of BBR in its own by analysing 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 [23] and [11], we don't need to generate multiple traffic and control the cwnd by holding the acknowledgments which can be exhaustive, as our method consists on analysing a simple packet trace so one single download/packet capture is sufficient. We also differ from [24] 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.

8.2 Troubleshooting tools

Guo et al. [29] developed pingmesh, a tool for large scale data center network latency measurement and analysis to track network latency issues. Zhu et al. [30] proposed Everflow, a packet level tracing and analysis tool. While [29] and [30] are 2 troubleshooting solutions, their scope is limited to a specific set of equipment-level performance metrics; this makes sense from a "repairman"'s point of view, to whom exonerating a specific router from guilt is critical, but is not sufficient to address an end-to-end scenario, where the offending connection spans continents and (possibly non-cooperative) actors. In our work, we aim to get a broader view of the issue at hand, by providing a cause-agnostic observable, the SS exit time, as input to further investigations.

9 Conclusion

As many network operators use the SS state duration as a key indicator for the diagnosis of faults, it is crucial to automate its extraction to save human experts time. In this work, we have presented a method to automatically detect the exit from the Slow-Start state, enabled by an innovative timeless representation of the observed packets series. We deployed our method in active and passive probes in 4 countries with varied access networks and traffic conditions, and tested it with both CUBIC and BBR. This evaluation shows the method to be accurate enough for the purpose, i.e. very often within 1 RTT of the oracle.

As a bonus, the representation, together with the SS exit time, proves to be a powerful enabler for classifiers, first of performance issues, and then of TCP CCAs, CUBIC and BBR.

In further academic work, we plan to refine the issue classifier so as to identify more classes and integrate the method in an automated tool. Our purpose is to deploy this classifier in all our probes and validate the solution at scale in a field trial. On another aspect, the CCA classifier could take part into the investigation of "fairness" in the wild, observing high-capacity links carrying multiple unknown TCP variants.

It should be noted that our timeless representation uses the BIF value derived from information in packets headers, which is impossible with passive (mid-point) observation of encrypted transport like QUIC traffic. Still, the method remains valid with active measurements, where encryption keys are available. Moreover, we are currently investigating a promising generalization of the $BIF = f(SEQ)$ representation to an "ACK-agnostic" variant, which is applicable to passive capture of encrypted traffic.

Conflicts of interest

Not applicable.

References

[1] Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC Editor (2004). https://doi.org/10.17487/RFC3954 . https://www.rfc-editor.org/info/rfc3954

[2] Yu, M., Jose, L., Miao, R.: Software defined traffic measurement with opensketch. In: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation. nsdi'13, pp. 29–42. USENIX Association, USA (2013)

[3] Panchen, S., McKee, N., Phaal, P.: InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC Editor (2001). https://doi.org/10.17487/RFC3176 . https://www.rfc-editor.org/info/rfc3176

[4] Veal, B., Li, K., Lowenthal, D.: New methods for passive estimation of tcp round-trip times. In: Dovrolis, C. (ed.) Passive and Active Network Measurement, pp. 121–134. Springer, Berlin, Heidelberg (2005)

[5] Benko, P., Veres, A.: A passive method for estimating end-to-end tcp packet loss. In: Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE, vol. 3, pp. 2609–26133 (2002). https://doi.org/10.1109/GLOCOM.2002.1189102

[6] Iyengar, J., Thomson, M.: QUIC: A UDP-Based Multiplexed and Secure Transport. RFC Editor (2021). https://doi.org/10.17487/RFC9000 . https://www.rfc-editor.org/info/rfc9000

[7] Ha, S., Rhee, I., Xu, L.: Cubic: A new tcp-friendly high-speed tcp variant. SIGOPS Oper. Syst. Rev. **42**(5), 64–74 (2008) https://doi.org/10.1145/1400097.1400105

[8] Cardwell, N., Cheng, Y., Yeganeh, S.H., Swett, I., Jacobson, V.: BBR Congestion Control. Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-02, Internet Engineering Task Force (March 2022). Work in Progress. https://datatracker.ietf.org/doc/draft-cardwell-iccrg-bbr-congestion-control/02/

[9] Tlaiss, Z., Hamchaoui, I., Amigo, I., Ferrieux, A., Vaton, S.: Troubleshooting enhancement with automated slow-start detection. In: 2023 26th Conference on

Innovation in Clouds, Internet and Networks and Workshops (ICIN), pp. 129–136 (2023). https://doi.org/10.1109/ICIN56760.2023.10073485

[10] Floyd, S.: Congestion Control Principles. RFC Editor (2000). https://doi.org/10.17487/RFC2914 . https://www.rfc-editor.org/info/rfc2914

[11] Mishra, A., Sun, X., Jain, A., Pande, S., Joshi, R., Leong, B.: The great internet tcp congestion control census. Proceedings of the ACM on Measurement and Analysis of Computing Systems **3**, 1–24 (2019) https://doi.org/10.1145/3366693

[12] Balasubramanian, P., Huang, Y., Olson, M.: HyStart++: Modified Slow Start for TCP. Internet-Draft draft-ietf-tcpm-hystartplusplus-13, Internet Engineering Task Force (January 2023). Work in Progress. https://datatracker.ietf.org/doc/draft-ietf-tcpm-hystartplusplus/13/

[13] Blanton, E., Paxson, D.V., Allman, M.: TCP Congestion Control. RFC Editor (2009). https://doi.org/10.17487/RFC5681 . https://www.rfc-editor.org/info/rfc5681

[14] Rüth, J., Kunze, I., Hohlfeld, O.: Tcp's initial window—deployment in the wild and its impact on performance. IEEE Transactions on Network and Service Management **16**(2), 389–402 (2019) https://doi.org/10.1109/TNSM.2019.2896335

[15] Stevens, W.R.: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC Editor (1997). https://doi.org/10.17487/RFC2001 . https://www.rfc-editor.org/info/rfc2001

[16] Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V.: Bbr: Congestion-based congestion control. Commun. ACM **60**(2), 58–66 (2017) https://doi.org/10.1145/3009824

[17] Sanders, C., Smith, J.: Applied network security monitoring, p. . Syngress, Boston (2014). https://doi.org/10.1016/B978-0-12-417208-1.09984-0 . https://www.sciencedirect.com/science/article/pii/B9780124172081099840

[18] Casey, E., Altheide, C., Daywalt, C., de Donno, A., Forte, D., Holley, J.O., Johnston, A., van der Knijff, R., Kokocinski, A., Luehr, P.H., Maguire, T., Pittman, R.D., Rose, C.W., Schwerha, J.J., Shaver, D., Smith, J.R.: Handbook of digital forensics and investigation, pp. 1–17. Academic Press, San Diego (2010). https://doi.org/10.1016/B978-0-12-374267-4.00001-X . https://www.sciencedirect.com/science/article/pii/B978012374267400001X

[19] Transmission Control Protocol. RFC Editor (1981). https://doi.org/10.17487/RFC0793 . https://www.rfc-editor.org/info/rfc793

[20] Kary: Understanding the tcptrace time-sequence

graph in wireshark. https://packetbomb.com/understanding-the-tcptrace-time-sequence-graph-in-wireshark/

[21] Tlaiss, Z.: Anomaly root cause diagnosis from active and passive measurement analysis. In: 2021 33th International Teletraffic Congress (ITC-33), pp. 1–3 (2021)

[22] Hagos, D.H., Engelstad, P.E., Yazidi, A., Kure, O.: General tcp state inference model from passive measurements using machine learning techniques. IEEE Access **6**, 28372–28387 (2018) https://doi.org/10.1109/ACCESS.2018.2833107

[23] Padhye, J., Floyd, S.: On inferring tcp behavior. ACM SIGCOMM Computer Communication Review (2001) https://doi.org/10.1145/383059.383083

[24] Yang, P., Luo, W., Xu, L., Deogun, J., Lu, Y.: Tcp congestion avoidance algorithm identification. In: 2011 31st International Conference on Distributed Computing Systems, pp. 310–321 (2011). https://doi.org/10.1109/ICDCS.2011.27

[25] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., Towsley, D.: Inferring tcp connection characteristics through passive measurements, pp. 1582–15923 (2004). https://doi.org/10.1109/INFCOM.2004.1354571

[26] Toshihiko Kato, R.Y. Leelianou Yongxialee, Ohzahata, S.: A study on how to characterize tcp congestion control algorithms from unidirectional packet traces. ICIMP 2016 : The Eleventh International Conference on Internet Monitoring and Protection (2016)

[27] Kato, T., Yan, X., Yamamoto, R., Ohzahata, S.: Identification of tcp congestion control algorithms from unidirectional packet traces, pp. 22–27 (2018). https://doi.org/10.1145/3291842.3291922

[28] Zhang, Y., Breslau, L., Paxson, V., Shenker, S.: On the characteristics and origins of internet flow rates. SIGCOMM Comput. Commun. Rev. **32**(4), 309–322 (2002) https://doi.org/10.1145/964725.633055

[29] Guo, C., Yuan, L., Xiang, D., Dang, Y., Huang, R., Maltz, D., Liu, Z., Wang, V., Pang, B., Chen, H., Lin, Z.-W., Kurien, V.: Pingmesh: A large-scale system for data center network latency measurement and analysis. SIGCOMM Comput. Commun. Rev. **45**(4), 139–152 (2015) https://doi.org/10.1145/2829988.2787496

[30] Zhu, Y., Kang, N., Cao, J., Greenberg, A., Lu, G., Mahajan, R., Maltz, D., Yuan, L., Zhang, M., Zhao, B.Y., Zheng, H.: Packet-level telemetry in large datacenter networks. SIGCOMM Comput. Commun. Rev. **45**(4), 479–491 (2015) https://doi.org/10.1145/2829988.2787483