Jitterbug: A new framework for jitter-based congestion inference

Esteban Carisimo[1], Ricky K. P. Mok[2], David D. Clark[3], and kc claffy[2]

[1] Northwestern University {esteban.carisimo}@northwestern.edu
[2] CAIDA, UC San Diego {cskpmok, kc}@caida.org
[3] MIT {ddc}@csail.mit.edu

Abstract. We investigate a novel approach to the use of jitter to infer network congestion using data collected by probes in access networks. We discovered a set of features in jitter and *jitter dispersion* —a jitter-derived time series we define in this paper— time series that are characteristic of periods of congestion. We leverage these concepts to create a jitter-based congestion inference framework that we call *Jitterbug*. We apply Jitterbug's capabilities to a wide range of traffic scenarios and discover that Jitterbug can correctly identify both recurrent and one-off congestion events. We validate Jitterbug inferences against state-of-the-art autocorrelation-based inferences of recurrent congestion. We find that the two approaches have strong congruity in their inferences, but Jitterbug holds promise for detecting one-off as well as recurrent congestion. We identify several future directions for this research including leveraging ML/AI techniques to optimize performance and accuracy of this approach in operational settings.

1 Introduction

The general notion of network congestion – demand exceeds capacity for network (link capacity or router buffer) resources – is widespread on the Internet, and an inherent property of traditional TCP dynamics. A TCP connection endpoint induces congestion to infer its appropriate sending rate, increasing this rate until it fails to receive acknowledgement of receipt of a packet by the other endpoint, i.e., infers congestion based on packet loss [19]. More recent attempts to improve TCP's congestion control algorithms rely on increased latency rather than packet loss as a signal of congestion [22,7,8,35].

Outside of protocol dynamics, latency and loss are still the fundamental metrics used to detect episodes of network congestion, or more generally path anomalies that degrade performance [13,14,30,17,12,15]. Although researchers have developed autocorrelation techniques to infer persistent recurrent patterns congestion [12], the challenge of detecting one-off episodes of congestion in traffic data remains an open problem after 30 years of Internet evolution. One-off episodes of congestion have many causes, including traffic management transitions, router operating system overheads, network configuration errors, flash crowds (e.g., software releases), and DDoS attacks. Inferring congestion from these phenomenological events is still an open challenge for the research network community.

We propose a new framework – Jitterbug – to use jitter and other metrics derived from round-trip-time (RTT) measurements to infer congestion. RTT measurements alone are often insufficient to infer congestion episodes, but we found that jitter-related metrics can distinguish congestion from other path anomalies, e.g., route changes. Specifically, we identify a correlation between periods of elevated latency (minimum RTT) and changes in the profile of jitter signatures – *jitter dispersion* – during congestion episodes. Relying on this concept, we develop a new framework that allows us to extend interdomain congestion inferences from recurrent patterns to one-off congestion events, i.e., discern recurrent from one-time congestion events. Using data collected between 2017 and 2020, this novel approach obtains similar results to state-of-the-art autocorrelation-based methods [12], but overcomes the limitation of the autocorrelation methods that can only detect recurrent periodic patterns of congestion. We find that Jitterbug introduces a promising approach to detect one-off congestion events. Our contributions are:

1. We identified a set of features in jitter and jitter dispersion time series, including a change of regime or transitory increase of the jitter dispersion, that characterize periods of congestion.
2. We used these features to develop and implement Jitterbug, a new jitter-based congestion inference method that combines pre-existing approaches to change point detection with information embedded in jitter signals.
3. We applied the Jitterbug framework to a wide range of challenging traffic scenarios, and explain its inferences.
4. We compare Jitterbug congestion inferences to the state-of-the-art autocorrelation-based methods [12], finding strong consistency in autocorrelation-applicable scenarios, i.e., for recurrent periodic congestion.
5. We release the source of code of Jitterbug[4].

The rest of the paper is structured as follows. We provide context by describing the latency model (§2.1) and jitter signatures in multiple real-world examples (§2.2). Leveraging these concepts, §3 describes Jitterbug and its components in detail. §4 describes the dataset we use to *(i)* investigate Jitterbug congestion inferences in different scenarios (§5), and *(ii)* cross-validate Jitterbug congestion inferences against other methods (§6). §7 summarizes lessons we learned during our study. §8 provides an extensive list of related work and §9 discusses open challenges in congestion inference. Finally, §10 offers concluding thoughts.

2 Background on RTT and Jitter Signatures

To provide context, we describe the latency model (§2.1) and four typical signatures we extract from RTTs and jitter (§2.2).

[4] Jitterbug repository: https://github.com/estcarisimo/jitterbug

2.1 Latency model

Round-trip time (RTT) in end-to-end measurements comprises both deterministic and random components. Eqn. (1) depicts the components of RTT between source (u) and destination (v) for a packet traversing a total of H hops in the round-trip path [21].

$$RTT(u,v) = d_{icmp} + \sum_{i=0}^{H} (d_s(i) + d_{prop}(i) + d_q(i) + d_{proc}(i)), \tag{1}$$

where d_{icmp} is the processing delay of ICMP messages in routers. d_s, d_{prop}, and d_{proc} represent delay induced by serialization, propagation, and packet processing, respectively. These deterministic components do not depend on traffic volume or link utilization. In contrast, d_{icmp} and d_q are random variables and contribute RTT variance, because their values depend on router CPU utilization and queue size of network interfaces when packets arrive. Prior work [23,12] has shown that RTT correlates with bottleneck link utilization, indicating that the queuing delay is the dominant factor in delay variation. Delay jitter, also referred to as jitter or IP packet delay variation [10], is the absolute difference between the current RTT value and the reference value of the previous time episode (i.e., $J_T = RTT(u,v)_T - RTT(u,v)_{T-1}$), where T is the current time episode. In this work we develop and evaluate a framework for using simple RTT and jitter-based metrics to classify path anomalies.

2.2 Analyzing RTT and jitter signatures in congested links

We use four real-world examples to illustrate the challenges and opportunities of using RTT and jitter to detect and identify path anomalies (Fig. 1). We focus on three properties of RTT and jitter to characterize the nature of path anomalies: *periodicity, amplitude, variability.*

Periodicity captures events that recur at a fixed frequency and duration, such as diurnal variations.

Amplitude measures the degree of changes in RTTs from the baseline. During network congestion events, probe packets are more likely to experience queuing delay. The elevation of RTTs reflects the queue size in the bottleneck link.

Variability refers to the stability of RTTs during the elevated periods, which allows us to discern congestion from other path anomalies such as a route change.

Fig. 1 shows four examples of two-week RTT and jitter time series measured from four vantage points in the U.S. to four router interfaces on the far-side[5] of interdomain links. Two examples (Fig. 1a and 1b) show periodic inflation in RTTs (blue/orange curves), indicating recurring congestion events. However,

[5] We referred as *near* and *far* sides to consecutive IP pairs in a traceroute path following the convention defined by Luckie *et al.* [23].

the jitter amplitude (green curve) in Fig. 1b, is much lower than that of Fig. 1a, consistent with a smaller queue size in the bottleneck link. Previous use of autocorrelation methods have shown that such persistent diurnal elevations in RTT at the far-side of an interdomain are evidence of interdomain congestion [12]. In contrast, the two cases in Fig. 1c and 1d are one-off events. The interesting difference is that in Fig. 1d the jitter increases as the RTT baseline jumps from 20ms to 40ms. In contrast, in Fig. 1c the jitter remains stable throughout. We suspect that this latter scenario was a route change event rather than congestion.

Although many different approaches to RTT change point detection could partition these time series into intervals, an approach solely based on RTTs would fail to distinguish congestion from other path anomalies such as route changes. The RTT signal is simply too noisy. This example shows that evaluating changes in jitter can enable us to differentiate these scenarios and thus we should consider jitter as a metric for characterizing path anomalies.

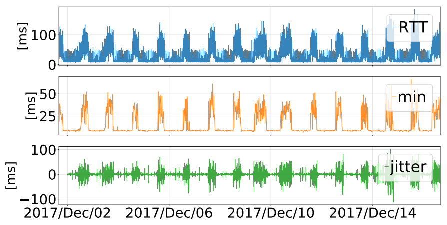
We next introduce our framework to support systematic analysis and classification of type of path anomalies with three properties that we extract from RTT and jitter time series data.

3 Jitterbug: Jitter-based congestion inference

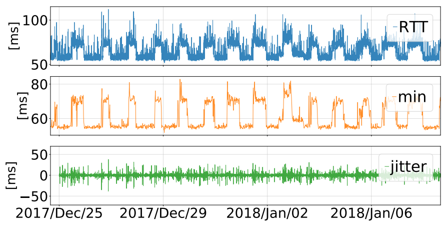
Fig. 2 shows the building blocks of our framework, which combines change point detection algorithms (§3.2) with simultaneous analysis of minimum RTT and jitter time series obtained from latency measurements. The change point detection algorithm splits RTT timeseries into *candidate* time intervals that might suffer from congestion. The next step of the framework is to analyze the jitter in each time interval to classify candidate intervals as congestion events or other path anomalies. We infer congestion based on the three elements we observed in §2.2: changes in baseline RTT, increase of jitter amplitude, and increase of jitter dispersion during a phase transition. We developed two different statistical methods for this analysis–*(i) KS-test method*, and *(ii) jitter dispersion method (JD)*. The first combines detection of changes on RTT latency baseline with the Kolmogorov-Smirnov (KS) test to detect changes in the jitter time series. The *jitter dispersion method (JD)* detects a jitter dispersion increase that correlates with a baseline RTT increases as a signal of congestion. The common goal of both methods is to objectively capture the signatures in the jitter signals. This section describes the role of each element of the Jitterbug framework in detail. We designed Jitterbug to support different RTT data sources, and have applied it to measurements collected by Ark CAIDA and RIPE Atlas. The current implementation uses a 5-minute and 15-minute granularity for RTT measurements and the aggregated minimum RTT time-series, respectively.

3.1 Signal filtering

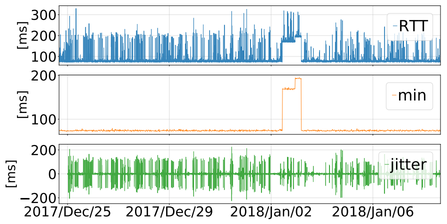
Jitterbug congestion inferences use three signals: *(i) min* RTT time series, *(ii)* jitter, and *(iii)* jitter dispersion. As we saw in §2.2, raw RTTs can be too noisy



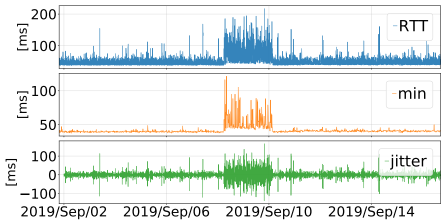
(a) Recurring congestion event. RTTs and jitter increase during congestion episodes.



(b) Recurring congestion event. Jitter does not significantly increase during elevated RTT periods, likely due to small buffer size in the bottleneck router.



(c) One-off non-congestion event on Jan 2. Jitter remained stable in face of inflated RTT.



(d) One-off congestion event. Increased varibility in jitter indicates the occurance of congestion.

Fig. 1. Typical examples of network events. The raw timeseries (top figures) is the raw RTT data with 5-minute resolution. We aggregate the raw data into 15-minute buckets with the minimum function to filter noise (middle figures). We compute jitter using the 15-minute aggregated data to quantify variability in RTTs (bottom figures).

to yield meaningful signatures. We first aggregate the raw RTT data by selecting the minimum value in each 15-minute time interval (*min* time series). The signal filtering module then computes the jitter using both the *raw RTT* and *min* time series to produce *jitter* and *j-min* time series, respectively.

We use two additional filters to better capture the variability in *j-min*. First, we apply the *Moving IQR filter* to the *j-min* time series, which computes the inter-quartile range (IQR) of a sliding window of 150 minutes (10 jitter samples). We define as jitter dispersion to the operation of computing the moving IQR to a jitter signal.

We then compute the 5-sample moving average of the resultant time series as the *jitter dispersion* time series to mitigate the impact of short-term latency spikes. Fig. 3 shows the correlation between the *min* RTT time series and the *jitter dispersion* of previous examples (Fig. 1). Correlation between the two time series in Fig. 3c) is low. We believe that the shift of baseline RTT corresponds to a route change that increased the propagation delay, which is a deterministic component that induces low variance to RTTs.

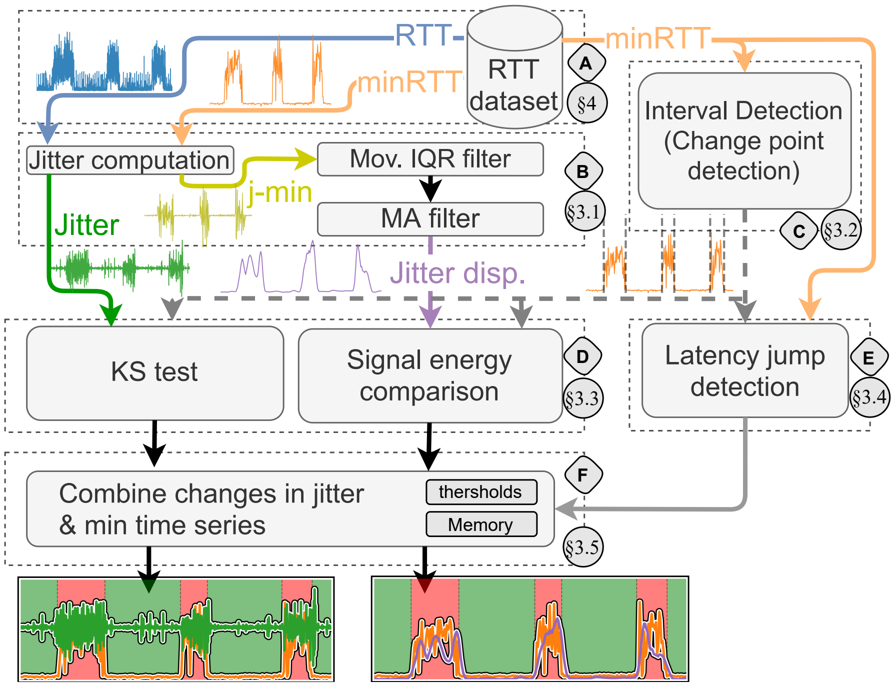
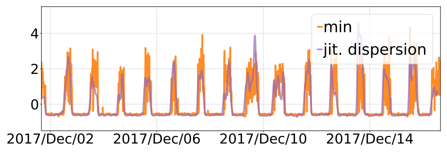


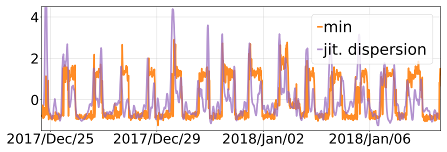
Fig. 2. The Jitterbug framework comprises: (A) data acquisition (B) signal filtering (C) detection of intervals of elevated latency (D) detection of changes of state of jitter and jitter dispersion signals (E) detection of increments of the min time series (F) correlation of changes in jitter state with increments of changes of state in jitter signals.

3.2 Detection of period of elevated latency

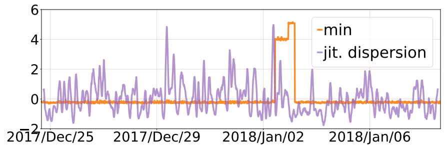
Identifying time intervals with elevated RTTs. is a fundamental step of the congestion inference process since the subsequent modules examine these periods to determine if latency elevations were caused by increases of traffic loads. Our framework can accommodate any change point detection algorithm that can segment time intervals based on changes in RTTs. As proof of concept, we use two state-of-the-art change point detection algorithms—Bayesian Change Point (BCP) Detection or Hidden Markov Models (HMM)—to process the *min* time series. We have not yet had the opportunity to test these methods on a large variety of data sources, so we provide both alternatives to let Jitterbug users select which is more effective for their data source. We believe these two algorithms can complement each other in circumstance where one fails to cover all change points in a signal. In §5.8, we test both algorithms with challenging latency signatures and show how all periods of elevated latency are captured by at least one of the methods.



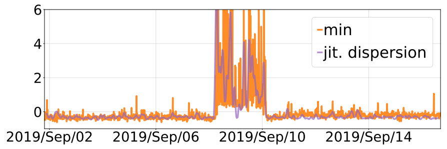
(a) Recurring congestion event. There is a correlation between periods of elevated latency and the growth of jitter dispersion.



(b) Recurring congestion event. Jitter dispersion transitorily increases at the beginning of periods of elevated latency



(c) One-off non-congestion event on Jan 2, with no correlation between signals.



(d) One-off congestion event, with a positive correlation between the *min* and jitter dispersion time series.

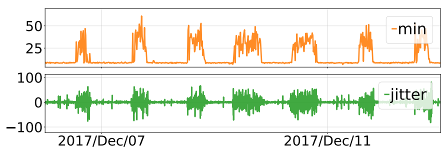
Fig. 3. *min RTT* (orange) and jitter dispersion (purple) time series. We normalized the values using standard score for this visualization; normalization is not necessary in actual computation. In Fig. 3a and 3d, these two signals are strongly correlated during period of congestion. In Fig. 3b, jitter dispersion has a transitory increase at the beginning of the period of elevated latency. Fig. 3c (no apparent congestion) shows no correlation between these signals, which is consistent with a route change that increased RTT.

– **Bayesian Change Point (BCP)**: We chose an offline BCP algorithm[6] proposed by Xuan *et al.* [36]. We experimented with other popular change point detection algorithms (e.g., Change finder [11] and ATDK LevelShift [6]) and found that BCP was the most effective at detecting boundaries of intervals with RTT latency measurements in our data.
– **Hidden Markov Models (HMM)**: We selected an implementation designed to identify different discrete states in RTT latency time series, by combining Hidden Markov Models (HMM) with Hierarchical Dirichlet Process (HDP) [27]. HMM also yields boundaries for each state (or level) in the time series, and in our case, consecutive RTT latency samples typically belong to the same state for long periods of times.

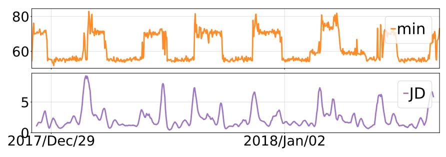
3.3 Examination of jitter signals

Jitterbug uses two approaches to examine changes in jitter and jitter dispersion time series during periods of elevated latency (Module (D) in Fig. 2): *(i) KS-test method* (using the jitter time series), and *(ii) Jitter dispersion method* (using the

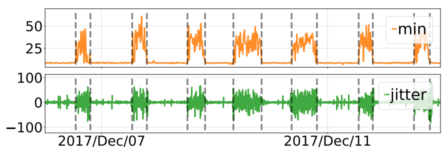
[6] Implementation of Xuan *et al.* change point detection algorithm: https://github.com/hildensia/bayesian_changepoint_detection.



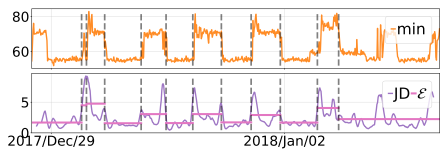
(a) KS-test: STEP 0



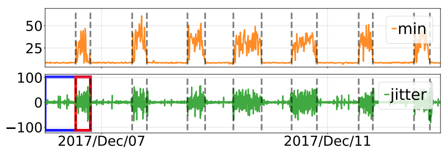
(b) JD: STEP 0



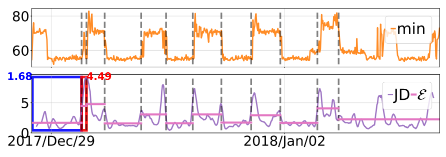
(c) KS-test: STEP 1 (BCP)



(d) JD: STEP 1 (BCP)



(e) KS-test: STEP 2 (Compare intervals)



(f) JD: STEP 2 (Compare intervals)

Fig. 4. Steps of KS-test (left) and jitter dispersion (right) congestion inference methods. In both methods, Jitterbug uses the *min* time series to identify the beginning and end of periods of elevated latency (Fig. 4c and Fig. 4d). Using these boundaries, both methods look for changes in jitter signals in adjacent intervals. To detect these changes, KS-test computes the Kolmogorov-Smirnov test on adjacent jitter samples (Fig. 4e); the jitter dispersion method (Fig. 4f) compares the mean value of the jitter dispersion signal ($\mathcal{E}$).

jitter dispersion time series). Both methods rely on boundaries previously identified by the Interval Detection Module (Module (C) in Fig. 2). Fig. 4 describes the input time series of each method, how they use change points detected by Interval Detection Module and how they detect changes in jitter signals.

KS-test method This method examines changes in the jitter time series (Fig. 4a). Using the change points extracted from the minimum time series by the Interval Detection Module (Fig. 4c), Jitterbug detects a change of regime in jitter time series during periods of elevated latency. Our hypothesis is that a trace switched into a different congestion state if there is a change point in the minimum time series and, at the same time, the jitter changes to a different regime. To identify such a regime, Jitterbug applies the Kolmogorov-Smirnov (KS) test to jitter samples in partitions before and after the change point (Fig. 4e). In case the jitter samples in the partition before the change point have a different distribution from the following partition, the KS test will reject the null hypothesis ($\alpha = 0.05$) meaning both samples were not generated by the same random process. To verify that the result of the KS test is not an artifact due to the change point detection method, we apply the KS test to two random samples in the same interval. For this validation test, we expect the KS test does not reject the null hypothesis, which means there is no evidence to conclude that samples within the same partition belong to different jitter regimes. We repeat this process for all pairs of adjacent partitions.

Jitter dispersion method The input to this method is the jitter dispersion time series that we pre-computed in §3.1 (Fig. 4b). Similar to the KS-test method, this method uses change points extracted from the minimum time series by the Interval Detection Module (Fig. 4c), as boundaries between periods of elevated latency (Fig. 4d). We assume when the elevation of latency is caused by congestion, then the jitter dispersion increases, either transitorily at the beginning (phase transition) or throughout the period (§3.1).

In both cases, during a period of congestion the average jitter dispersion is larger than that of congestion-free periods. If the mean value of the jitter dispersion between consecutive periods (Fig. 4f) increases, we consider this period as congested. We repeat this inference process for all pairs of adjacent partitions.

3.4 Latency Jump Detection

Jitterbug assumes that *a period of congestion is a period of elevated latency* that manifests the growth of routers' buffers occupancy. In the Latency Jump Detection module (Module (E) in Fig. 2) Jitterbug uses the *min* time series and the intervals identified by the Interval Detection Module (Module (C) in Fig. 2) to detect latency increments. This modules flags a candidate *period of congestion* if it detects in that period an increment of the mean value of the *min* time series compared to its predecesor.

3.5 Combine changes in jitter and minimum time series

Jitterbug classifies a *period of congestion* (Module F in Fig. 2) if adjacent intervals meet two conditions: *(i)* an increase in the RTT latency baseline *(ii)* an increase in the jitter amplitude (transitory or generalized). Jitterbug combines the results obtained by the Latency Jump Detection module with KS-test and Jitter dispersion methods. Jitterbug assumes that a period of elevated latency was generated by an increase in routers buffer occupancy if the KS-test or jitter dispersion method detected changes in the jitter signals during that interval.

To increase the accuracy of these Jitterbug inferences in challenging scenarios, and to allow users to calibrate inferences with their tolerance values, Jitterbug includes two *additional features*: *(i) congestion inference thresholds*, and *(ii) memory.*

- **Congestion inference thresholds**. To increase confidence of Jitterbug congestion inferences, we include *congestion inference thresholds* when we

Table 1. Description of the near- and far-side ASes in the evaluation dataset. We use measurements collected from 13 Ark monitors hosted in 6 U.S. ISP to 18 far-side ASes (7 Content Providers and 11 Access/Transit networks) and 49 far-IP addresses. This data collection comprises 1.7M raw RTT samples collected between 2017 and 2020. for 1290 unique combinations of <day, VP, far IP>.

near-side ASes		far-side ASes	
# VPs	ISPs	#ASes (# addr.)	far ASname
13	COMCAST, Verizon, AT&T, CenturyLink, Charter, Cox	18 (49)	COMCAST (AS7922), Netflix (AS2906), NTT (AS2914), Level3 (AS3356), PCCW (AS3491), KT (AS4766), Telstra (AS4637), TATA (AS6453), China Telecom (AS4134), Zayo (AS6461), Cloudflare (AS13335), Charter (AS7843), XO (AS2828), Edgecast (AS15133), Google (AS15169), Amazon (AS16509), Akamai (AS20940), Facebook (AS32934)

compare the mean value of the minimum RTT time series of consecutive intervals. We also include a Jitter dispersion threshold (JD threshold) in the jitter dispersion method when we compare changes in the mean value of the signal in adjacent intervals. The values we use for this research are 0.25ms and 0.5ms thresholds for jitter dispersion and baseline, respectively, as we found in the evaluation dataset (see § 4) that min and jitter dispersion fluctuations tend to be below these values during periods of no suspected congestion. These parameters allow us to reduce false positives and false negatives in Jitterbug congestion inferences.

– **Memory**. To reduce errors in congestion inferences as a result of false positives in the change point detection process, we include the concept of *memory*. In some cases, change point detection algorithms identify path anomalies within periods of congestion (e.g., route change during a congestion episode) or a false positive. Under these circumstances, our congestion detection methodology would not detect any change, either transitory or permanent in the jitter, and it would label the next interval as a *period of no congestion*. However, the congestion status has not changed between these adjacent intervals. To overcome this limitation, we include a rule called *memory* that assumes that *a period of congestion has not finished if in the following interval the mean value of the minimum RTT does not decrease.* For example, for two given adjacent intervals I_1 and I_2, we will label I_2 as *a period of congestion* if we also labeled I_1 as *a period of congestion* and $mean(minRTT(I_2)) \geq mean(minRTT(I_1))$.

4 Dataset

We focus on congestion at interdomain links which requires identification of IP addresses of intedomain routers' interfaces. MANIC [2] uses *bdrmap* [24] to infer the IP addresses of all interdomain links visible from the Autonomous System hosting a CAIDA Ark [1] vantage point (VP). *bdrmap* returns pairs of near- and far-side IP addresses of an interdomain link, and a set of prefixes reachable through a path containing those near- and far-side IP addresses. We use the data API of the MANIC platform [2] to obtain longitudinal RTT measurements from Ark's VPs to the far-side interface of interdomain links using the Time-Series Latency Probing (TSLP) method [23]. Each VP runs TSLP measurements every 5 minutes using ICMP TTL-limited packet probes to all near- and far-side pairs to collect RTT samples between the VP and IP addresses on the near and far side of interdomain links. Furthermore, the MANIC platform labels interdomain links that might have congestion events using an autocorrelation-based method [12], which is effective in locating recurring congestion events that significantly inflate the RTTs. We will use these inferences as cross-validation (§6.2).

We demonstrate our methodologies by inferring congestion from 13 VPs in 6 U.S. ISPs to 18 far ASes and 49 far-IP addresses, as it is shown in Table 1. This dataset covers a total of 1290 unique combinations of <day, VP, far IP> and contains 1.7M raw RTT samples collected between 2017 and 2020.

5 Results

We present our results of Jitterbug congestion inferences in the scenarios we introduced in §2.2, which map to the taxonomy in Fig. 5. Specifically, we show Jitterbug congestion inferences for periodic signals of large (§5.1) and small (§5.2) amplitude as well as for one-off periods of elevated latency (§5.3 and 5.4). We further investigate Jitterbug congestion inference in hybrid scenarios with one-off events in the middle of repetitive periods of elevated latency (§5.5). We also study the impact of *memory* (§5.6) and the *JD threshold* (§5.7) in the accuracy of Jitterbug congestion inferences. Finally, we investigate how errors in detecting change points impact in Jitterbug congestion inference (§5.8).

5.1 Scenario 1: recurrent period of elevated latency with large amplitude signals

Both methodologies labeled every recurrent period of elevated latency as a period of congestion (Fig. 6). We suppose that the accuracy of the congestion inferences is partially due to the small of contribution of other random factors since we observe small variability in the baseline during periods of non-elevated latency. The profile of the minimum time series indicates a small contribution of other random components, which create slight fluctuations during periods of non-elevated latency. In addition, the size of this router buffer amplifies the range of the raw, min and jitter time series (in some cases over 100ms) which simplifies the task of identifying periods of high jitter fluctuations.

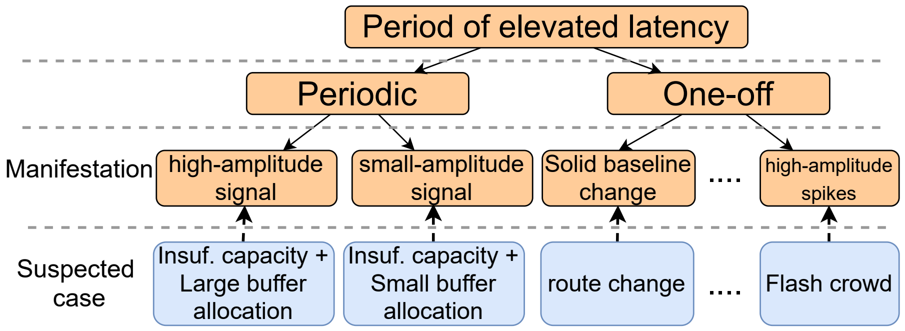


Fig. 5. Hierarchical classification of characteristics of elevated latency. We classify periods of elevated latency as either *periodic* (left branch) or *one-off* (right branch). Recurrent latency with a consistent period (periodic) suggests an underprovisioned link. A one-off episode of elevated latency can have many causes, e.g., bufferbloat, flash crowd, misconfiguration, route change.

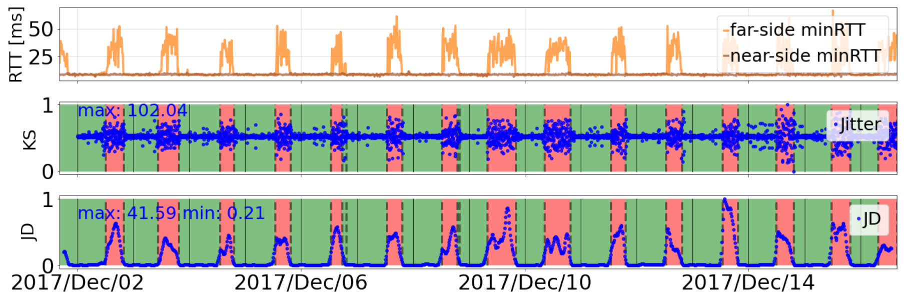


Fig. 6. KS-test (middle plot) and Jitter dispersion (lower plot) congestion inferences for a periodic high-amplitude signal. In this case, both methods label every recurrent period of elevated latency as periods of congestion. Red-filled intervals indicate periods of congestion.

5.2 Scenario 2: recurrent period of elevated latency with small amplitude signals

Fig. 7 shows that only the jitter dispersion method labels periods of elevated latency as periods of congestion. We believe that the stability in the jitter time series at periods of elevated latency impedes the KS-test method's inferences. This jitter stability may be due to small buffers (differences between peak and valley values is 30ms) or traffic engineering on the far side network, which in this case is a large Content Provider. On the other hand, the high amplitude of the phase transitions in the jitter dispersion time series allows the JD method to detect differences in the mean value of this signal during periods of elevated latency. We note that the change point detection module is not capable of detecting period of elevated latency between January 1, 2018 and January 3, 2018.

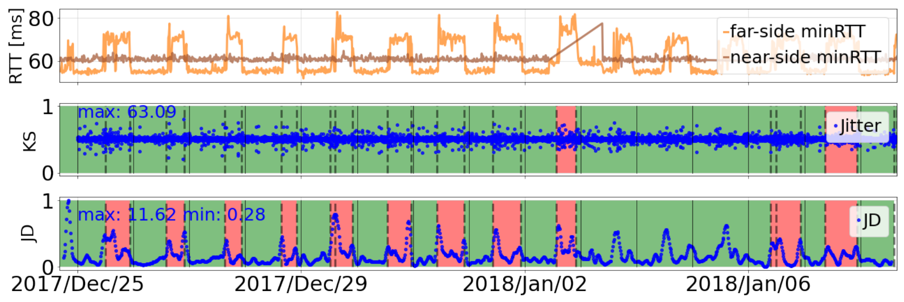


Fig. 7. KS-test and Jitter dispersion congestion inferences for a periodic small-amplitude signal. Only the jitter dispersion method infers congestion from this recurrent pattern, which we speculate relates to small buffers that keep jitter itself relatively stable. Remarkably, the change point detection algorithm was not able to capture some periods of elevated latency. Red-filled intervals indicate periods of congestion.

(The slightly smoother transition during this period trace could have hindered the accuracy of the change point detection algorithm.)

5.3 Scenario 3: one-off period of elevated latency with no congestion

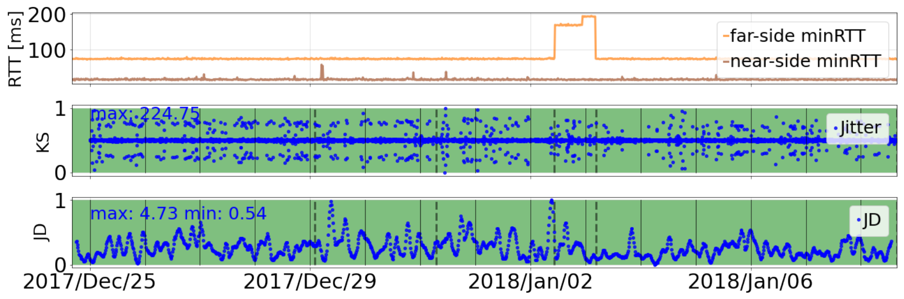


Fig. 8. KS-test and Jitter dispersion congestion inferences for a one-off event suspected as a route change. Inferences for this case indicate no congestion. Red-filled intervals indicate periods of congestion.

Fig. 8 shows an example in which neither method infers congestion. In this case, we do not observe any change in either the jitter time series or the jitter dispersion either before or after the period of elevated latency. We suppose that this period corresponds to a route change based on the stability of the jitter time series and the clean profile of the min time series during the transition. Since there is no simultaneous increase in near-side RTT (orange curve in Fig. 8

top panel), we believe that a route changed in the reverse path from the far-side router.

5.4 Scenario 4: one-off period of elevated latency with congestion

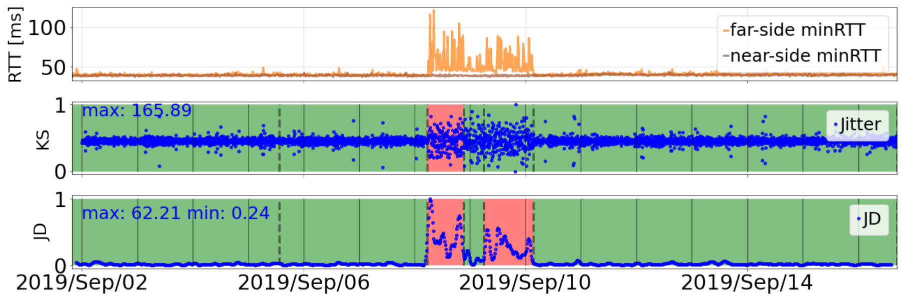
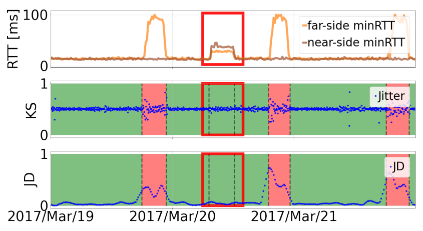


Fig. 9. KS-test and Jitter dispersion congestion inferences for a one-off congestion event. In this case, both methods infer congestion during periods of elevated latency. Red-filled intervals indicate periods of congestion.

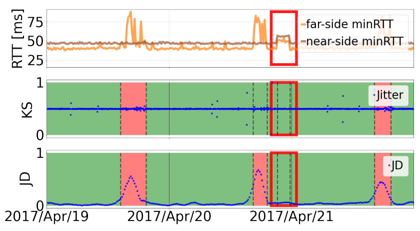
(Fig. 9) Congestion inferences from both methods partially agree on classifying this one-time episode of high amplitude latency spikes as a period of congestion. Detection of multiple change points, and the fact that the period in between has slightly smaller mean value in the min time series, generate that the period of congestion inferred is smaller than the actual period of elevated latency.

5.5 Scenario 5: one-off event during recurrent periods of elevated latency

The biggest challenge for latency-based congestion detection is to distinguish congestion-induced elevated latency from other path anomalies, such as a route change. Fig. 10 shows two examples of KS-test and jitter dispersion congestion inferences when route changes occur in the middle of recurrent periods of elevated latency. In these cases, we confirm that the events occurring on March 20, 2017 at 12pm (Fig. 10a) and on April 20, 2017 before midnight (Fig. 10b) are route changes in the internal network of the ISP since the near- (orange) and far-side (blue) *min* time series detect an elevation simultaneously. As we expected for a route change, these events do not show any change in jitter signals. Our method used the jitter dispersion metric to correctly rule out a candidate congestion period as a route change (rather than congestion), due to low jitter dispersion which we know is not strongly correlated with congestion dynamics. This example illustrates the importance of jitter dynamics in detection of network congestion events.



(a)



(b)

Fig. 10. Two examples of suspected route changes in the middle of recurrent periods of elevated latency. Neither method inferred any congestion. Red-filled intervals indicate periods of congestion.

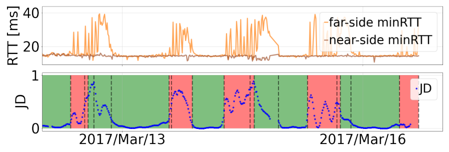
5.6 Scenario 6: Change point detection over-detects change points

We use an additional set of examples to investigate how the *memory* feature compensates for weaknesses in change point detection algorithms, specifically when algorithms are over-sensitive and create too many intervals.

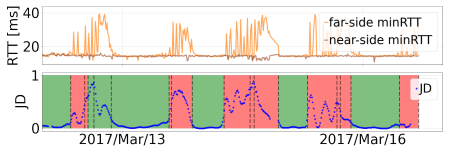
Fig. 11 shows examples of how *memory* improves the accuracy of congestion inferences in different circumstances. Fig. 11a and 11b shows how *memory* increases the accuracy of congestion inferences in the presence of over-partitioned periods of elevated latency. While this feature increases the number of intervals labeled as periods of congestion in the presence of multiple change points, it is not able to fix all of them. Fig. 11c and 11d show how *memory* extends the inferred period of congestion where there is a legitimate change point during this period. These figures show a persistent increase in the minimum RTT baseline, which we suspect was due to a route change during a period of congestion. We assume that the lack of RTT measurements below that baseline corresponds to speed-of-light constraints induced by the more circuitous path used during the period of congestion.

5.7 Scenario 7: Adjusting JD threshold to minimize false positives

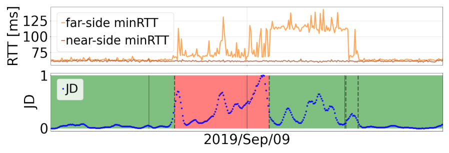
Fig. 12 shows examples of how one can adjust the JD threshold to minimize false positives in congestion inferences. Fig. 12a and 12b compare congestion inferences using JD thresholds of 0.25 ms and 0.5 ms, respectively. In this example, the jitter dispersion ranges from 0.26 to 92.64 ms, showing a flat curve for most of the period and a one-off event that generates a large spike. Due to the flatness of the curve we selected two thresholds close to the baseline jitter dispersion values (D+0.25 and D+0.5 ms), and inferred a period of congestion if jitter dispersion exceeded these thresholds, which in this case means the jitter dispersion doubled or tripled. We found that our first threshold (0.25) was too sensitive, since a small perturbation in jitter dispersion, in addition to a false positive inference from the change point algorithm, generated a false positive congestion inference.



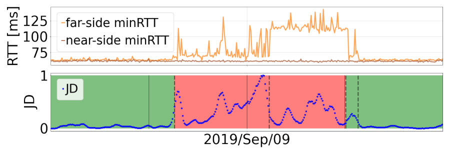
(a) Memoryless congestion inference



(b) memory congestion inference



(c) Memoryless congestion inference



(d) Memory congestion inference

Fig. 11. Examples of how the *memory* feature improves accuracy of congestion inferences in the presence of of over-partitioned intervals and other path anomalies. Fig. 11a and 11b display how *memory* maximizes congestion inferences in scenarios where change point detection algorithms overfit detection, breaking the time series into too many intervals. Fig. 11c and 11d show another example of how *memory* can inform congestion inference when a route change occurs within a period of congestion. Red-filled intervals indicate periods of congestion.

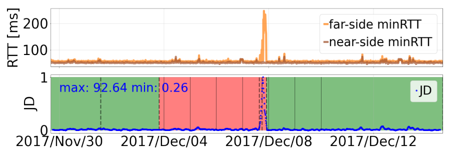
5.8 Scenario 8: False negatives in change point detection

One desired characteristic of a change point detection algorithm is the ability to precisely detect the beginning and ending points (all of them) of *all* periods elevated latency. In practice this is not possible for every time series, and in our case the lack of change points hinder the accuracy of congestion inferences. We use additional examples to investigate the accuracy of the change point detection algorithms we included in Jitterbug.

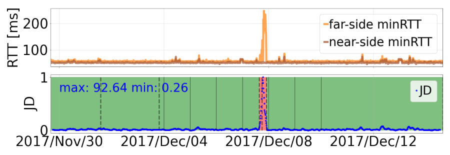
Fig. 13 shows two pairs of examples where the precision of Interval Detection varies depending on the algorithm being applied and the traffic scenario: BCP is more precise that HMM (Fig. 13a and 13b) and HMM is more precise than BCP (Fig. 13c and 13d)). Fig. 13a shows a scenario where HMM misses several consecutive change points, creating a prolonged period that does not precisely capture the periods of congestion in that measurement. For the same scenario, Fig. 13b shows that BCP correctly infers those periods of congestion. Conversely, Fig. 13c shows a scenario in which HMM is more accurate than BCP at detecting change points (Fig. 13d).

6 Comparative evaluation of Jitterbug

The current version of Jitterbug allows users to infer congestion using 4 different configurations by changing: *(i)* the change point detection algorithm (BCP or HMM, see §3.2), or *(ii)* the congestion inference method (KS-test or jitter



(a) JD threshold = 0.25 ms



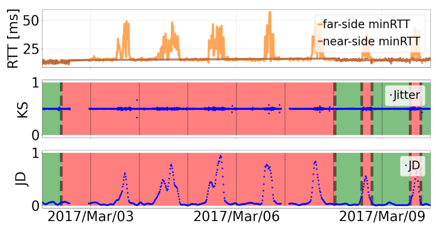
(b) JD threshold = 0.5ms

Fig. 12. Adjusting the JD threshold can mitigate false positive in congestion inferences. Fig. 12a shows that a too-sensitive threshold can yield errors even in the presence of a flat jitter dispersion time series. Fig. 12b shows how small adjustments in this threshold can mitigate false positive congestion inferences. Red-filled intervals indicate periods of congestion.

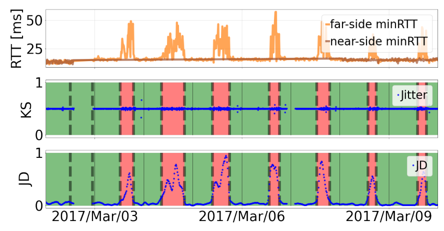
dispersion, see §3.3). In this section we compare Jitterbug inferences for each configuration, first comparing the KS-test and JD methods to each other (§6.1), and cross-validated with the state-of-the-art congestion detection methods [12] (§6.2).

6.1 Comparing inferences of KS-test and JD methods

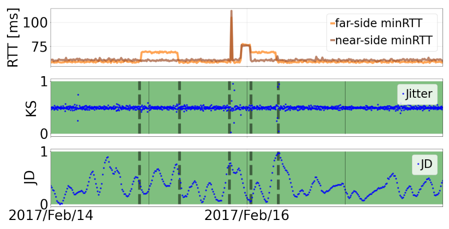
Table 2 compares congestion inferences of KS-test and jitter dispersion methods for the same interval using different change point detection alternatives (BCP on the left hand-side and HMM on the right hand-side). The results show no significant variations related to the change point detection used for the inferences. KS-test and jitter dispersion indicate the same congestion status for most intervals since the fraction of intervals equally labelled is 0.67 (128/192) and 0.64 (129/201) when using BCP and HMM, respectively. The jitter dispersion method tends to label more intervals as *period of congestion* than the KS-test method where the fraction of intervals considered as *periods of congestion* only by jitter dispersion is 0.29 (56/192) and 0.32 (63/201) for BCP and HMM, respectively. The KS-test method labels fewer intervals as *period of congestion* since this method only detects a narrow type of congestion signature in which congestion implies a change in jitter regime. For instance, when random components of latency are more significant than queueing delay, this noise limits the ability of KS-test to detect a change in the jitter regime. In addition, we found that the KS-test is unable to detect congestion generating changes of jitter regimes when a bottleneck router buffer is small. We suspect that small buffers do not allow us to observe jitter fluctuations to classify them as a change of jitter regime. Active traffic engineering strategies could keep jitter within a certain band. Despite that the KS-test method effectively infers congestion for a narrow type of congestion signature, we have included this method for its simplicity to detect congestion in cases with a large signal-to-noise ratio.



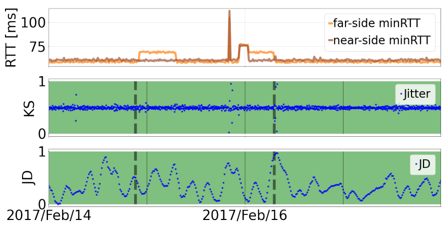
(a) Example A: HMM misses some change points



(b) Example A: BCP detects all change points



(c) Example B: HMM detects more change points than BCP (Fig. 13d)



(d) Example B: BCP misses some change points

Fig. 13. Two pairs of examples showing the limitations of change point detection algorithms to detect *all* change points (vertical dashed lines). Fig. 13a shows an example where HMM is not able to capture some change points in contrast to BCP that detects all of them (Fig. 13b). Fig. 13c shows an example where HMM is a more accurate than BCP at detecting change points (Fig. 13d) Red-filled intervals indicate periods of congestion.

6.2 Comparing inferences with cross-validation data

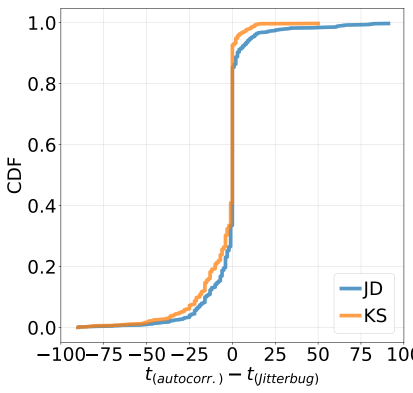
We validate KS-test and jitter dispersion congestion inferences using CAIDA's autocorrelation-based congestion inferences as cross-validation data. In the presence of recurrent congestion, CAIDA's congestion inferences count the number of 15-minute intervals with elevated latency. Using this schema, CAIDA's congestion inferences report the daily congestion severity of a link with a variable that ranges from 0 to 96[7]. We use Jitterbug outputs to generate the same daily estimations.

Fig. 14 shows how close are the daily congestion estimations of Jitterbug and CAIDA's congestion inference data. We also compared estimations with a maximum difference of 10% (in number of congested 15-minute intervals), and the fraction of days that agree to within this 10% margin rises to 76-80% depending on the combination (80% for JD method using BCP). The most prominent discrepancies in this evaluation corresponds to two categories: *(i)* Jitterbug false positive inferences in periods with no congestion, and *(ii)* one-off congestion

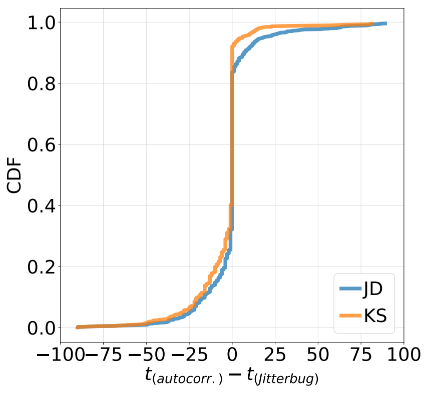
[7] One day has 96 periods of 15 minutes.

Table 2. Fraction (and total number) of (dis)agreements for different methodologies. The bar on top means a scenario with no congestion.

	BCP			HMM		
	$\overline{C_{KS}}$	C_{KS}	**SUM**	$\overline{C_{KS}}$	C_{KS}	**SUM**
$\overline{C_{JD}}$	0.43 (82)	0.04 (8)	0.47 (90)	0.39 (80)	0.04 (9)	0 .43 (89)
C_{JD}	0.29 (56)	0.24 (46)	0.53 (92)	0.31 (63)	0.24 (49)	0.57 (112)
SUM	0.72 (138)	0.28 (54)	**192**	0.70 (143)	0.28 (58)	**201**



(a) BCP



(b) HMM

Fig. 14. Cumulative distribution function of the differences between the estimated daily time of congestion by autocorrelation-based methods and Jitterbug. These methods show remarkable similarity: 52% of days show no difference in inference regardless of change point detection method or congestion-detection method (KS vs JD).

events detected by Jitterbug but not present in CAIDA's congestion inference data since CAIDA's method only attempts to infer recurrent (periodic) congestion episodes.

7 Lessons learned

In this section we enumerate important aspects we have identified for jitter-based congestion inference.

1. **Jitter and jitter dispersion signatures provide meaningful information to identify congestion events as periods of elevated latency**. We found that periods of congestion manifest in RTT latency measurements not only as periods of elevated latency, but also changes in jitter (and jitter-derivated signals) time series.

2. **Jitter signals allowed us to discard periods of elevated latency generated by other path anomalies, e.g., route changes**. Including jitter-based analysis in the detection of congestion events allowed us to differentiate congestion events from other path anomalies. In non-congestion-related events, jitter and jitter dispersion time series tend *not* to change during periods of elevated latency.
3. **Period of elevated latency only to the far-side does not necessarily mean congestion**. We noticed that the simultaneous periods of elevated latency to near- and far-sides suggest a route change in the internal network of the ISP but a period of elevated latency to the far-side only does not necessarily indicate congestion. Although in many cases a period of elevated latency to the far-side only indicates a growth in the buffer occupancy of the interdomain link, this event could also suggest a route change only in the reverse path from the far-side router. We use jitter and jitter dispersion to identify traces with elevations only to the far-side router but not corresponding to congestion events.
4. **Shallower increments of RTT values when a link transitions to a period of congestion tend to affect jitter signatures too**. We observed a negative correlation between the increment of RTT values during periods of congestion and the visibility of changes in jitter signatures. We suppose this decrement in the contrast of RTT latency values between periods of elevated latency and other periods is related to the size of router buffers. We speculate that modern recommendations to keep buffers small [5] will likely affect jitter time series.
5. **The contribution of other random components of RTT latency can reveal congestion dynamics** . Some traces contain random contributions that mask queueing delay fluctuations in the jitter time series during periods of elevated latency. Although this is not a widespread phenomenon, it could compromise Jitterbug's ability to infer congestion, especially with the KS-test method.
6. **Limitations of change point detection methodologies to detect *all* periods of elevated latency**. None of the change point detection algorithms we examined could identify all change points in the *min* time series in our data. There is a wide variety of signal profiles in RTT latency measurements and several types of congestion signatures, including periods of elevated latency with flat, smooth and spiky signatures. We suppose that change point detection algorithms may not be able to capture change points for all types of signatures in this large set of profiles. To be able to identify *all* periods of elevated latency is crucial since the accuracy of Jitterbug congestion inferences mostly relies on detecting these intervals.
7. **Change point detection is expensive**. The BCP and HMM methods required significant time to execute on the 15-day traces we analyzed for this study, typically between 60 and 90 seconds. Optimizing performance of these methods will be critical for operational utility.
8. **The KS-test method only captures a limited type of congestion event signature**. But it is a simple and clean congestion inference approach,

cost-effective for many scenarios beyond those we studied, and can inform further research in this area.

8 Related work

Inferring network congestion with RTT measurements. Previous research efforts focused on interdomain congestion inference leveraging from recurrent periods of elevated latency [12,23]. To generate these inferences, these works relied on a set of CAIDA's (Ark) [1] to run RTT latency measurements to all visible IP-level interconnection links [24]. Time Series Latency Probes (TSLP) [12,23] is the result of these latency measurement campaigns. An autocorrelation method is apply to traces on the TSLP data collection to find multi-day repetition of elevated delays around the same times, i.e., driven by diurnal demand. However, this method to detect congestion requires some level of manual inspection. With a similar approach, Fontugne *et al.* [15] proposed a latency-based methodology to detect congestion in last-mile access networks. They used RIPE Atlas probes to run traceroute measurements campaigns and inferred congestion applying a methodology to detect latency deviations.

Anomaly detection on network paths. RTT time series has been also used to detect a wide range of network events, such as path anomalies [13,14] and route changes [30,17].

Change point detection. Change Point Detection algorithms aim to detect *change point detection* (also known as time series segmentation) as abrupt changes in a sequence of observations (e.g., a time series) to divide a sequence into a finite number of non-overlapping partitions [3]. These algorithms are typically based on mathematical or machine learning models [3,4,11,31,36]. Another study found that some unsupervised anomaly detection tools for change point are notably time consuming [32]. Even though these methods are effective in capturing change points in the time series [9], event classification still requires human inspection.

Mathematical approaches for congestion detection. Another type of studies brought sophisticated mathematical and statistical concepts to investigate congestion events. Mouchet *et al.* [27] proposed to use Hidden Markov Models (HMM) to identify different states in RTT latency time series, however, these states correspond to different latency values and do not report discriminate events caused by different types of events (e.g., route change vs congestion event). More recently, Spang *et al.* [34] proposed to use A/B tests in TCP lab measurements to generate unbiased evaluations of TCP Congestion Control Algorithms (CCA). However, the applicability of this approach relies on the assumption on independent traffic flows, which in practice may be compromised by the synchronization of TCP flow and short-lived TCP transfers. In addition, engineers typically used more pragmatic evaluations to test the impact of their changes.

9 Open Challenges

Other approaches not covered in this paper may be useful to extract information embedded in jitter signals. Early in this project we proposed and tested at least other four different approaches to jitter-based congestion inference. One aimed to capture the jitter variability at the beginning of a period, and another applied the same concept of the KS-test method but using j-min (definition in §3.1) instead. A third alternative used anomaly detection techniques to detect changes in jitter volatility. The fourth alternative used parametric models, including Normal and Levy-Stable distributions, to fit jitter behavior. These alternative approaches are promising and it is worth exploring them as part of future work.

In the future we also expect fluctuations of queueing delay to become more challenging to distinguish in RTT latency measurements as a consequence of smaller router buffers following modern buffer sizing recommendations [5,16]. Jitterbug central assumption is that *a period of congestion is a period of elevated latency*, however, if latency signatures show imperceptible queueing delays, this may comprise the accuracy of change point detection algorithms to detect periods of elevated latency. In addition, the rise of delay-sensitive real-time applications (e.g., videocalls, online gaming, etc.) could also incentivize the reduction of router buffer sizes. We observed (§2.2) a correlation between jitter signatures and buffer sizes and recognize that smaller buffer sizes could impede Jitterbug congestion inferences.

More demanding requirements of jitter-sensitive applications (e.g. live video streaming) could also modify traffic patterns and latency signatures. Today's HTTP-based video delivery relies on playback modulation to mitigate jitter impact on video flow [28,29]. However, in the future, real-time video broadcasting may requiere shorter playback jitters — and consequently dedicated traffic engineering strategies — that could modify the shape of the jitter curve and thus Jitterbug inferences.

Foreseeable changes in the foundational protocols of the TCP/IP stack could modify traffic dynamics and the nature of latency signatures. New latency-based Congestion Control Algorithms could modify latency signatures and buffer occupancy. The rollout of QUIC [22,8,33,18] could spread new features in the network potentially reshaping the nature of traffic dynamics. For example, QUIC proposes to aggregate and multiplex multiple short-lived web data transfers — typically run in parallel per-resource TCP sessions [8]— into a single transport-layer protocol session.

We expect that future work from ML/AI communities develop more cost-effective change point detection tools. The growing necessity of monitoring large-scale time series databases to generate (near) real-time anomaly detection is likely to be the driver of optimization in this space [32]. We expect that in the coming years we are going to count with more rapid and optimized supervised and unsupervised anomaly detection algorithms to detect change points.

10 Conclusions and Future work

In this paper we proposed Jitterbug, a novel framework to infer network congestion combining pre-existing approaches with information embedded in jitter signals. We found that jitter allowed us to expand congestion inference beyond scenarios of recurrent congestion patterns, such as one-time congestion events. We discovered that jitter (and jitter-derived signals) time series is useful to discriminate periods of elevated latency caused by congestion from route changes.

We have also learned about the various challenges of inferring network congestion with RTT latency measurements. The vastly heterogeneous structure of the network is reflected in diverse latency signatures showing large and short buffer sizes, remarkable presence of randomness unrelated from congestion events, etc. We have also learned about limitations of change point detection algorithms in detecting all beginning and ending points of periods of elevated latency as well as the time required to obtain results from these algorithms.

Applying Jitterbug to the cases in our dataset, we obtained similar results to recent autocorrelation methods [12]. However, in contrast to that method, which is based on the repetitiveness of the signal and uses information of near- and far-side RTT latency measurements, Jitterbug is fully based on far-side RTT latency measurements and does not rely on repetitiveness to discern the congestion status of a period.

We hope that this work will encourage studies focused on network congestion inference, jitter analysis and change point detection algorithms. In the future, we would like to investigate how sampling rates (higher and lower) affect congestion inferences and profiles of RTT latency signatures. For example, studies in financial time series have found that the distribution of assets returns vary depending on the scaling factor (i.e. time elapsed between samples) [20,25], we would like to investigate if this also happens on jitter time series. We are also interested in studying whether we could develop purely jitter-based congestion inference methods. Another topic that we would like to investigate is if inter-packet delay in back-to-back measurements, for example using FAST probing tool [26], could allow us to infer congestion

11 Acknowledgements

We thank the anonymous reviewers for their insightful comments, and Maxime Mouchet for providing an implementation of the HMM algorithm. We would like to thank Fabian Bustamante (Northwestern University) for coming up with the original term *Jitterbug* to name this paper. This work was partly funded by research grants DARPA HR00112020014, NSF OAC-1724853 and NSF CNS-1925729.

References

1. Archipelago measurement infrastructure updates. https://catalog.caida.org/details/media/2011_archipelago, accessed: 2021-9-30

2. Manic. https://catalog.caida.org/details/software/manic, accessed: 2021-10-13
3. Adams, R.P., MacKay, D.J.: Bayesian online changepoint detection. arXiv preprint arXiv:0710.3742 (2007)
4. Aminikhanghahi, S., Cook, D.J.: A survey of methods for time series change point detection. Knowledge and information systems **51**(2), 339–367 (2017)
5. Appenzeller, G., Keslassy, I., McKeown, N.: Sizing router buffers. ACM SIGCOMM Computer Communication Review **34**(4), 281–292 (2004)
6. ARUNO: ADTK Detectors. https://arundo-adtk.readthedocs-hosted.com/en/stable/api/detectors.html (2021)
7. Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V.: Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. Queue **14**(5), 20–53 (2016)
8. Carlucci, G., De Cicco, L., Mascolo, S.: Http over udp: an experimental investigation of quic. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing. pp. 609–614 (2015)
9. Davisson, L., Jakovleski, J., Ngo, N., Pham, C., Sommers, J.: Reassessing the constancy of end-to-end internet latency. In: Proceedings of IFIP TMA (2021)
10. Demichelis, C., Chimento, P.: Rfc3393: IP packet delay variation metric for IP performance metrics (IPPM). https://datatracker.ietf.org/doc/html/rfc3393 (Nov 2002)
11. Desobry, F., Davy, M., Doncarli, C.: An online kernel change detection algorithm. IEEE Transactions on Signal Processing **53**(8), 2961–2974 (2005)
12. Dhamdhere, A., Clark, D.D., Gamero-Garrido, A., Luckie, M., Mok, R.K., Akiwate, G., Gogia, K., Bajpai, V., Snoeren, A.C., Claffy, K.: Inferring persistent interdomain congestion. In: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. pp. 1–15 (2018)
13. Fontugne, R., Mazel, J., Fukuda, K.: An empirical mixture model for large-scale RTT measurements. In: Proceedings of IEEE INFOCOM (2015)
14. Fontugne, R., Pelsser, C., Aben, E., Bush, R.: Pinpointing delay and forwarding anomalies using large-scale traceroute measurements. In: Proceedings of ACM Internet Measurement Conference (2017). https://doi.org/10.1145/3131365.3131384
15. Fontugne, R., Shah, A., Cho, K.: Persistent last-mile congestion: Not so uncommon. In: Proceedings of the ACM Internet Measurement Conference. pp. 420–427 (2020)
16. Gettys, J.: Bufferbloat: Dark buffers in the internet. IEEE Internet Computing **15**(3), 96–96 (2011)
17. Iodice, M., Candela, M., Battista, G.D.: Periodic path changes in RIPE Atlas. IEEE Access **7**, 65518–65526 (2019). https://doi.org/10.1109/access.2019.2917804
18. Iyengar (Ed.), J., Thomson (Ed.), M.: QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000 (Proposed Standard) (May 2021). https://doi.org/10.17487/RFC9000, https://www.rfc-editor.org/rfc/rfc9000.txt
19. Jacobson, V.: Congestion avoidance and control. ACM SIGCOMM computer communication review **18**(4), 314–329 (1988)
20. Jaroszewicz, S., Mariani, M.C., Ferraro, M.: Long correlations and truncated levy walks applied to the study latin-american market indices. Physica A: Statistical Mechanics and its Applications **355**(2-4), 461–474 (2005)
21. Laki, S., Mátray, P., Hága, P., Csabai, I., Vattay, G.: A detailed path-latency model for router geolocation. In: EAI Tridentcom. IEEE (2009). https://doi.org/10.1109/tridentcom.2009.4976258
22. Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., Swett, I., Iyengar, J., et al.: The quic transport protocol: Design and

internet-scale deployment. In: Proceedings of the conference of the ACM special interest group on data communication. pp. 183–196 (2017)

23. Luckie, M., Dhamdhere, A., Clark, D., Huffaker, B., Claffy, K.: Challenges in inferring internet interdomain congestion. In: Proceedings of the 2014 Conference on Internet Measurement Conference. pp. 15–22 (2014)
24. Luckie, M., Dhamdhere, A., Huffaker, B., Clark, D., Claffy, K.: Bdrmap: Inference of borders between ip networks. In: Proceedings of the 2016 Internet Measurement Conference. pp. 381–396 (2016)
25. Mantegna, R.N., Stanley, H.E.: Econophysics: Scaling and its breakdown in finance. Journal of statistical Physics **89**(1), 469–479 (1997)
26. Marder, A., Claffy, K.C., Snoeren, A.C.: Inferring cloud interconnections: Validation, geolocation, and routing behavior. In: International Conference on Passive and Active Network Measurement. pp. 230–246. Springer (2021)
27. Mouchet, M., Vaton, S., Chonavel, T., Aben, E., Den Hertog, J.: Large-scale characterization and segmentation of internet path delays with infinite hmms. IEEE Access **8**, 16771–16784 (2020)
28. Mustafa, I.B., Nadeem, T.: Dynamic traffic shaping technique for http adaptive video streaming using software defined networks. In: 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). pp. 178–180. IEEE (2015)
29. Pu, W., Zou, Z., Chen, C.W.: Video adaptation proxy for wireless dynamic adaptive streaming over http. In: 2012 19th International Packet Video Workshop (PV). pp. 65–70. IEEE (2012)
30. Pucha, H., Zhang, Y., Mao, Z.M., Hu, Y.C.: Understanding network delay changes caused by routing events. ACM SIGMETRICS Performance Evaluation Review **35**(1), 73–84 (jun 2007). https://doi.org/10.1145/1269899.1254891
31. Punskaya, E., Andrieu, C., Doucet, A., Fitzgerald, W.J.: Bayesian curve fitting using mcmc with applications to signal segmentation. IEEE Transactions on signal processing **50**(3), 747–758 (2002)
32. Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., Zhang, Q.: Time-series anomaly detection service at microsoft. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 3009–3017 (2019)
33. Rüth, J., Poese, I., Dietzel, C., Hohlfeld, O.: A first look at quic in the wild. In: International Conference on Passive and Active Network Measurement. pp. 255–268. Springer (2018)
34. Spang, B., Hannan, V., Kunamalla, S., Huang, T.Y., McKeown, N., Johari, R.: Unbiased experiments in congested networks. arXiv preprint arXiv:2110.00118 (2021)
35. Turkovic, B., Kuipers, F.A., Uhlig, S.: Interactions between congestion control algorithms. In: 2019 Network Traffic Measurement and Analysis Conference (TMA). pp. 161–168. IEEE (2019)
36. Xuan, X., Murphy, K.: Modeling changing dependency structure in multivariate time series. In: Proceedings of the 24th international conference on Machine learning. pp. 1055–1062 (2007)