A Machine Learning Approach to Estimating Queuing Delay on a Router over a Single-Hop Path

Travis Ricker, Khondaker Salehin, Yi Wang, and Alex Chen
California State University, Dominguez Hills, USA
tricker1@toromail.csudh.edu, {ksalehin, ywang, achen}@csudh.edu

Eiji Oki
Kyoto University, Japan
oki@i.kyoto-u.ac.jp

Roberto Rojas-Cessa
NJIT, USA
rojas@njit.edu

***Abstract*—Queuing delay is a dynamic network parameter that plays an important role in defining the performance of Internet applications over an end-to-end path. However, measurement of queuing delay is challenging because it requires a large infrastructural support from the path under test. In this paper, we propose an active scheme to measure queuing delay on a router using a probe-gap model. The scheme uses a popular data-clustering algorithm to process its data samples; therefore, its measurement efficacy is not dependent on the issues related to infrastructural access, certain variations (e.g., compression) in the probe gaps, and the number of clusters in the data processing. Here, we present a detailed evaluation of the scheme against the current state-of-the-art on a single-hop path through ns-3 simulation. Our results show that the proposed scheme is robust, consistent, quick, and highly accurate under different traffic conditions.**

***Index Terms*—Network measurement, queuing delay, clustering algorithm, wired networks, ns-3 simulator.**

I. Introduction

Queuing delay refers to the waiting time of data packets inside the intermediate nodes (i.e., routers) over an end-to-end path [1]. It is a dynamic network parameter that is dependent on the instantaneous states (e.g., rate and number) of the competing traffic flows on the routers. Even though high transmission speeds of the network links are expected to minimize queuing delays on routers, non-trivial (i.e., non-zero) queuing delays on Internet paths are still commonplace [2], [3]. Therefore, accurate measurement of queuing delay is essential for ensuring optimal performance of various applications in the conventional Internet (e.g., IP geolocation) and Tactile Internet (e.g., telesurgery) environments [4], [5].

Multiple schemes are available for measuring queuing delay through passive or active measurement [5], [6]. Passive schemes are popular among Internet Service Providers that use ongoing data traffic on the path under test for measuring queuing delay. However, their strict requirements for infrastructural support (e.g., specialized equipment [7], [8] or specific network architecture [9]) and administrative access (e.g., direct access to the router under test [10]) make them less deployable on the Internet. On the contrary, active schemes are not as restrictive for wider deployments on the Internet because they use commodity equipment (e.g., workstations) and synthetic probing packets for measuring the parameter. However, the limiting issue of these schemes is either inconsistent accuracy due to their specialized probing packets (i.e., Internet Control Message Protocol, ICMP, packets [11], [12]) or strict topological configurations (e.g., tree-structured network topology [13]) for the path under test.

To overcome the limitations of the existing schemes, an active scheme, called COMPRESS, was recently proposed to perform accurate measurement without any support from the networks [4], [5]. COMPRESS uses User Datagram Protocol (UDP) packets in its packet-pair structure, consisting of a small heading packet (P_h) and a large trailing packet (P_t). It exclusively uses compression in the gap between P_h and P_t to estimate queuing delay on the router under test. For example, Figure 1 shows the transmission of multiple packet pairs through an intermediate router ($router$) after each pair is sent, with no separation between P_h and P_t, from the input link (L_i) to the output link (L_o). L_o illustrates three possible scenarios, i.e., compression (co), no change (nc), and decompression (de), in the packet pair, as identified by the corresponding intra-probe gaps: G_{co}, G_{nc}, and G_{de}. These scenarios occur due to the sizes of P_h and P_t along with the interference from cross-traffic packets (Cs) inside $router$. COMPRESS ignores G_{de} and calculates G_{nc} as a prior knowledge from the transmission times of P_h and P_t on L_o and L_i, respectively [5], [6]. It then determines queuing delay on $router$ from the differential between G_{nc} and G_{co}. In short, COMPRESS relies on a restrictive set of its probing data that has the potential to compromise the measurement accuracy.

In this paper, we propose an enhanced scheme based on a packet-pair model for actively measuring queuing delay on an Internet router. This scheme has two important properties: i) It uses an unsupervised learning algorithm that allows robust and self-sufficient measurement of queuing delay without any prior information from the infrastructure. ii) It uses both compression and decompression in pairs of packets. We perform a comparative evaluation of the proposed scheme over a single-hop path through computer simulation. Our simulation results show that the proposed scheme is robust enough to consistently measure queuing delay with a high accuracy under different traffic conditions.

II. Proposed Scheme

We propose an active scheme, called **Co**mpression and **De**compression (CoDe), based on an unsupervised learning algorithm for measuring queuing delay on a router. CoDe uses both the compression and decompression phenomena in pairs

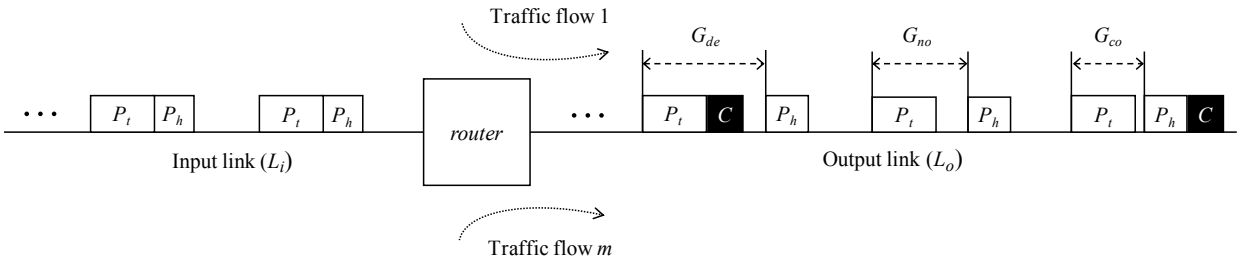


Fig. 1: Packet pairs, each with a small heading packet (P_h) and a large trailing packet (P_t), experience compression, no change, and decompression in the intra-probe gaps (i.e., G_{co}, G_{nc}, and G_{de}, respectively, where $G_{co} < G_{nc} < G_{de}$) in the presence of cross-traffic flows.

of UDP packets after their transmissions through the router under test.

We outline the working principle of CoDe using Figure 2, which shows a trimodal distribution of intra-probe gaps for multiple packet pairs following their transmissions through a router with cross traffic. Here, the local peaks refer to the individual modalities of co, nc, and de in the packet pairs due to different cross-traffic interferences, as discussed in Section I. Given the statistical means of these local peaks: M_{co}, M_{nc}, and M_{de}, queuing delay on the router can be estimated from the magnitude of compression (i.e., M_{nc} - M_{co}) and the magnitude of decompression (i.e., M_{de} - M_{nc}).

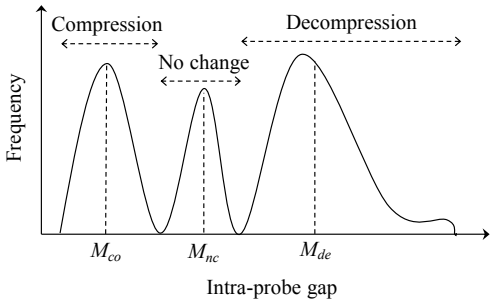


Fig. 2: A trimodal distribution of intra-probe gaps for a train of packet pairs in the presence of cross-traffic flows on a router that represents compression, no change, and decompression with three local means M_{co}, M_{nc}, and M_{de}, respectively.

However, the primary challenge in the above mechanism is the identification of modalities in the dataset (i.e., measured intra-probe gaps) that represent co, nc, and de during queuing-delay measurement. We mitigate this challenge in CoDe by utilizing the k-means clustering algorithm [14]. Here, we use a fixed cluster size, i.e., k = 3, to incorporate the required modalities in the dataset. Therefore, there is no ambiguity in optimally processing the dataset using the k-means clustering algorithm for producing a consistent measurement.

We chose the k-means clustering algorithm in the proposed scheme because it is a popular unsupervised learning algorithm that is efficient, easy to implement, and guarantees convergence [15], [16]. Our choice of algorithm was also motivated by the prior futile use of a supervised learning algorithm, i.e., linear regression, in an ICMP packet-based scheme, called Pathchar [11], which produces inconsistent results in its measurement [18]. Moreover, advanced machine learning approaches, e.g, deep learning algorithms, are not suitable for measuring queuing delay because of two fundamental reasons. Firstly, our dataset is one-dimensional and small; therefore, its complexity is simple. Secondly, deep learning algorithms only provide enhanced performance with a very large dataset in complex problems [16], [17]. In a few words, k-means clustering algorithm came out as the best candidate for processing the dataset of the proposed scheme.

An important characteristic of CoDe is its robustness and self-sufficiency as an active scheme. For example, COMPRESS requires prior information about the link capacities over a path to determine the reference value (i.e., no change gap) on the router under test for estimating queuing delay [5], [6]. The prior information is obtained through link-capacity measurement which, however, is prone to high errors on the Internet paths [22]. This infers that erroneous capacities are a source of performance bottleneck for COMPRESS towards estimating queuing delay accurately. Because CoDe identifies the no-change gaps from the k-means clusters, its performance is strictly reliant on its dataset consisting of intra-probe gaps, not on additional information.

We present the detailed steps of CoDe for measuring queuing delay over a single-hop path, consisting of a router ($router$) connecting a source (src) and destination (dst) nodes, below:

1. Send a train of n packet pairs, each consisting of a small P_h and a large P_t without any separation in between, from src to dst. Details about sizing P_h and P_h over the single-hop path are available in [6].
2. Timestamp P_h and P_t of each packet pair at dst to measure n intra-probe gaps.
3. Process the measured gaps using the k-means clustering algorithm to partition them into three different clusters.
4. Label the clusters as co, nc, and de for compression, no change, and decompression, respectively, in the packet pairs such that the cluster centroids (M_{co}, M_{nc}, and M_{de}) have the following relationship:

$$M_{co} < M_{nc} < M_{de} \quad (1)$$

5. Truncate every intra-probe gap (i.e., member gap) in each cluster that has a frequency less than $0.2s_i$, where s_i is the (population) size of its holding cluster, i.e., s_{co}, s_{nc}, or s_{de}. Then update the centroid of each cluster using its remaining member gaps.
6. Determine queuing delay on $router$ from the centroids:
   a. Calculate queuing delay (w_{co}) from compression:
   $$w_{co} = M_{nc} - M_{co} \tag{2}$$
   b. Estimate queuing delay (w_{de}) from decompression:
   $$w_{de} = M_{de} - M_{nc} \tag{3}$$
   c. Estimate overall queuing delay (w):
   $$w = \sum_{i=\{co,de\}} w_i \bar{s}_i, \tag{4}$$
   where $\bar{s}_i$ is the weighted cluster size, e.g., $\bar{s}_{co} = \frac{s_{co}}{s_{co}+s_{de}}$.
   d. Estimate the variability (σ) in w:
   $$\sigma = \sqrt{\frac{1}{n'-1}\sum_{j=1}^{n'}(x_j - w)^2}, \tag{5}$$
   where $x_j \in \{co, de\}$ and $n' = s_{co} + s_{de}$.
7. Report a range of queuing delay estimated over the single-hop path as $w \pm \sigma$.

III. Simulation Results

We implemented CoDe in ns-3 [19] to evaluate its performance over an end-to-end path, as shown in Figure 3. We identify this path as single-hop because it consists of only one router [6]. We incorporated the standard k-means algorithm (i.e., Lloyd's algorithm [20]) in the scheme. Our implementation of CoDe also utilized the k-means++ algorithm [21] to initialize the cluster seeds for optimal data processing.

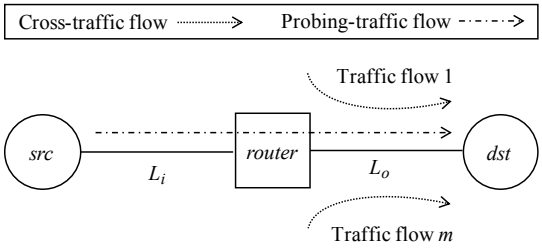


Fig. 3: Simulation setup with multiple cross-traffic flows over a single-hop path, where a router under test ($router$) connects src and dst using input (L_i) and output (L_o) links, respectively.

Figure 3 shows the intermediate router ($router$) over the single-hop path carries $m = 4$ cross-traffic flows between src and dst nodes. These flows are made up of constant-bit-rate (CBR) traffic that create packet queuing on the router from collision probabilities. For example, these flows carry approximately 457 (400), 229 (267), 152 (160), and 152 (160) Mb/s of cross traffic when the single-hop path has 1-Gb/s links. In case of 100-Mb/s links on the path, these flow rates are 47, 24, 16 and 12 Mb/s.

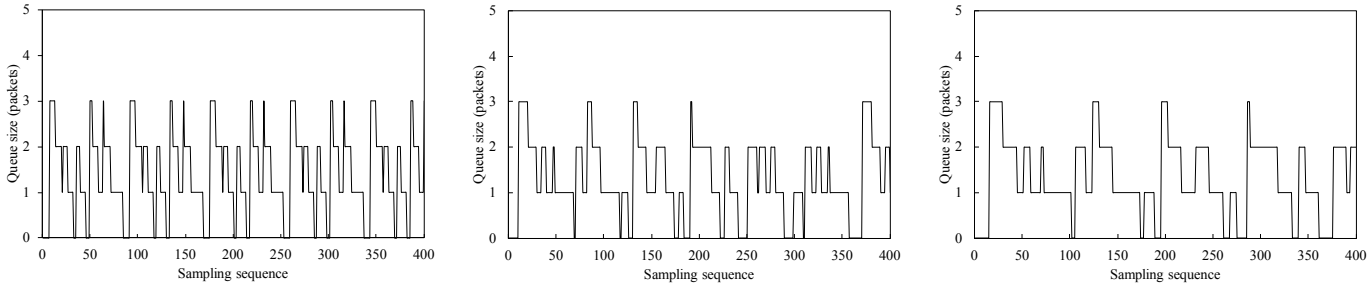
In Figure 4, we present sample distributions of queuing (i.e., the first 400 samples of the output-queue size for every 0.5 µs) that we have induced on $router$ using (a) 400-, (b) 600- and (c) 900-byte packets in the cross-traffic flows considering a 1-Gb/s speed over the single-hop path. In each graph of this figure, the x-axis is the sampling sequence of the output queue and the y-axis is the size of the queue in cross-traffic packets. On average, these distributions generate (4 ± 3), (7 ± 3), and (9 ± 5) µs of queuing delays for the respective packet sizes. In the generated queuing delays, the first digit corresponds to the average value whereas the second digit corresponds to the standard deviation, i.e., variability of the parameter.

We performed a comparative evaluation of CoDe against COMPRESS using P_h = 64 bytes and P_t = {800, 1000, 1400} bytes in reference to the generated queuing delays under 1 Gb/s. We chose these packet sizes in P_h and P_t for both schemes because i) they create large enough gaps between P_h and P_t for estimating the actual queuing delays using COMPRESS and ii) they provide an unbiased evaluation. Note that we did not evaluate CoDe against other active schemes because of their specialized requirements, e.g., ICMP packets as probes, which cannot be emulated through computer simulation for a useful performance evaluation [6].

Figure 5 presents a summary of the comparative evaluation under 1 Gb/s. Here, the x-axis refers to the size of P_t, and the y-axis refers to the rounded queuing delay in microseconds. The (grey) solid bars with whiskers are the actual (i.e., generated) queuing delays, whereas the (blue) dotted and (orange) tiled bars with whiskers are the queuing delays estimated by COMPRESS and CoDe, respectively. The estimated queuing delays correspond to a statistical summary of five measurement runs by both schemes in our simulation. We used 500 packet pairs, each separated by 10–50 ms, in every measurement run to avoid self-interference among our probing packets.

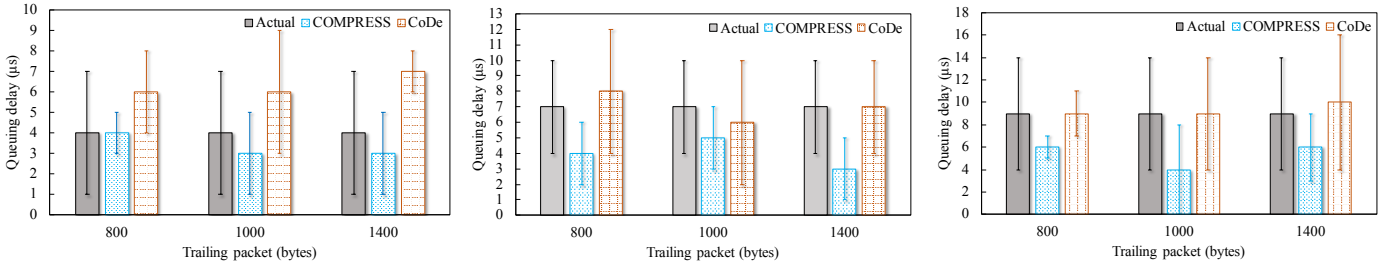
Overall, Figure 5 shows that the ranges of the measured values, identified by the corresponding whiskers for both COMPRESS and CoDe, heavily overlap with the actual queuing delays for all P_t sizes. Considering the fact that queuing delay is a dynamic parameter, this phenomenon infers that both schemes have a high measurement accuracy. The same phenomenon has also been used for characterizing measurement accuracy in [5], [6].

Besides producing a high accuracy, CoDe shows more consistency than COMPRESS in Figure 5. The performance of COMPRESS is characterized by a varied degree of underestimation in its measurements for all queuing delays under test. Due to the fact that COMPRESS strictly uses compression in the packet pairs, underestimation in its results is expected. On the contrary, the results of CoDe are closely aligned with the actual values for P_t = {1000, 1400} bytes when we compare the bar graphs and their whiskers in Figures 5(b) and 5(c). This suggests that the performance of CoDe is not dominated by underestimation.



(a) 400-byte packets (b) 600-byte packets (c) 900-byte packets

Fig. 4: Distributions of output-queue size on the router when it is carrying cross-traffic flows with (a) 400-, (b) 600-, and (c) 900-byte packets generating queuing delays of (4 ± 3), (7 ± 3), and (9 ± 5) µs, respectively.



(a) Queuing delay: (4 ± 3) µs (b) Queuing delay: (7 ± 3) µs (c) Queuing delay: (9 ± 5) µs

Fig. 5: Comparative performance between CoDe and COMPRESS over the single-hop path, consisting of 1 Gb/s links, with (a) (4 ± 3), (b) (7 ± 3), and (c) (9 ± 5) µs of queuing delays.

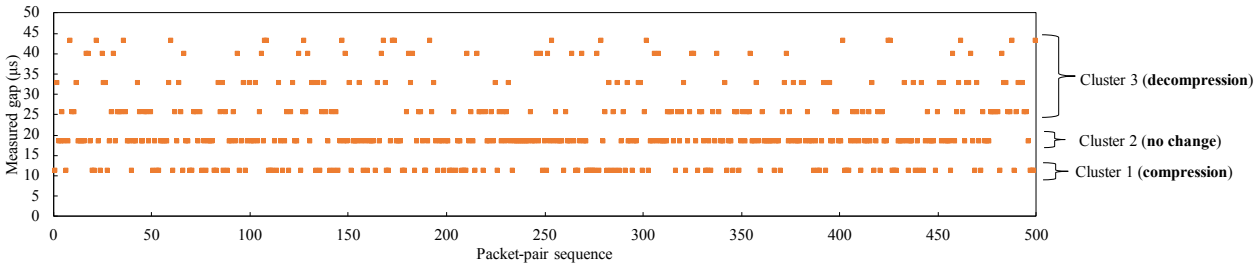


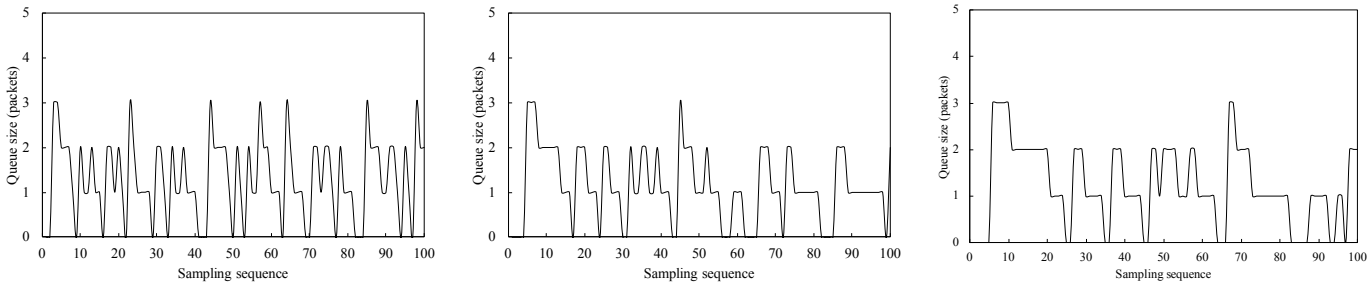
Fig. 6: Measured gaps for estimating (9 ± 5) µs of queuing delay using CoDe with P_t = 1400 bytes in the packet pairs.

We present a sample dataset from our simulation to provide insight into the data processing in Figure 6. This figure illustrates the intra-probe gaps from the first run of measurement by CoDe using P_h = 1400 bytes for estimating (9 ± 5) µs of queuing delay, where the expected no-change gap is 21 µs. This gap is the theoretical intra-probe gap that refers to the difference between the transmission times of P_h and P_t through the input and output links, respectively, without cross traffic on the router [5], [6]. For the k-means clusters outlined in this figure, we present Table I to detail the member gaps, size, and centroid of all clusters that represent compression, no change, and decompression in the gaps. These values correspond to the outcome from 100 iterations of the k-means algorithm on the measured gaps in the above-mentioned measurement. Note that the cumulative size of the three clusters in Table I is not equal to the probing-train size of 500 packets that was used during the measurement. It is because two member gaps (40 and 43 µs) in cluster 3 did not comply with the 20% rule discussed in Step 5 of Section II.

TABLE I: k-MEANS CLUSTERS IN FIGURE 6

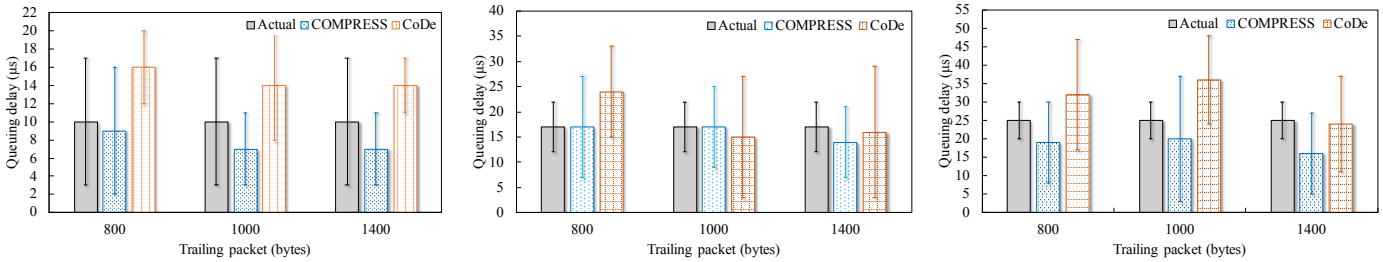
Cluster ID	Cluster label	Member gaps	Cluster size	Cluster centroid
1	compression	11 µs	114	11 µs
2	no change	18 µs	187	18 µs
3	decompression	26 µs, 33 µs	147	30 µs

The queuing delays used so far were challenging because of their high (43% or more) variability. However, the magnitudes of the queuing delays were below 10 µs. Therefore, we ran additional simulations with heavier queuing delays of (10 ± 7), (17 ± 13), and (25 ± 18) µs, each with a very high (70%



(a) 100-byte packets (b) 200-byte packets (c) 300-byte packets

Fig. 7: Distributions of output-queue size on the router when it is carrying cross-traffic flows with (a) 100-, (b) 200-, and (c) 300-byte packets generating heavy queuing delays of (10 ± 7), (17 ± 13), and (25 ± 18) µs, respectively.



(a) Queuing delay: (10 ± 7) µs (b) Queuing delay: (17 ± 13) µs (c) Queuing delay: (25 ± 18) µs

Fig. 8: Comparative performance between CoDe and COMPRESS over the single-hop path, consisting of 100 Mb/s links, with (a) (10 ± 7), (b) (17 ± 13), and (c) (25 ± 18) µs of queuing delays.

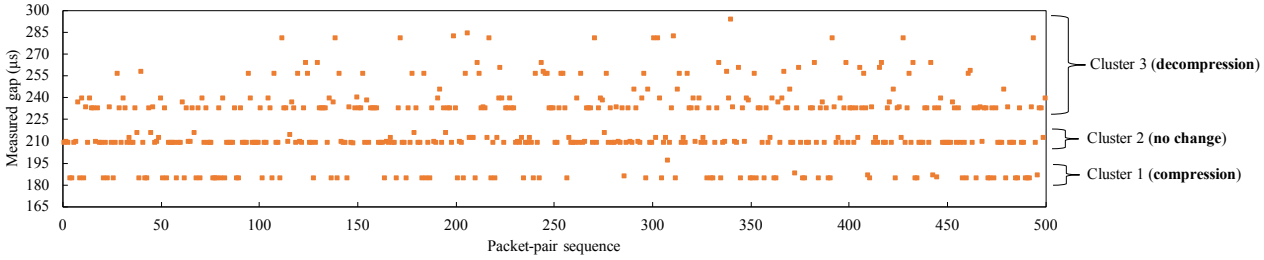


Fig. 9: Measured gaps for estimating (25 ± 18) µs of queuing delay using CoDe with P_t = 1400 bytes in the packet pairs.

or more) variability, on the router. Figure 7 illustrates the distributions of these queuing delays over a sampling period of 100 µs, generated with (a) 100-, (b) 200-, and (c) 300-byte packets in the cross-traffic flows.

Figure 8 summarizes the comparative performance of CoDe and COMPRESS under the heavier queuing delays. As in Figure 5, the accuracies of both schemes are high, as demonstrated by the consistent overlapping of the actual and estimated queuing delays. Each graph of Figure 8 also proves the tendency of underestimating queuing delays by COMPRESS. In the case of CoDe, the close alignment between the estimated and actual values is primarily visible in Figures 8(b) and 8(c) for larger P_t sizes, e.g., 1400 bytes. We, however, observe a degree of overestimation by CoDe. This phenomenon is an effect of the high variability in the generated queuing delays. For example, the actual queuing delays in Figure 8 tend to create more frequent decompressions in the packet pairs with a few outlier (i.e., very large) gaps that eventually resulted in overestimations in the measurements.

It may appear that COMPRESS outperforms CoDe concerning the bar-graph values of smaller P_t sizes, i.e., 800 and 1000 bytes, in Figure 8. Note that we manually provided the link capacities of the single-hop path to COMPRESS in our simulations, as was also done for evaluating this scheme in [5], [6]. This, therefore, eliminated the performance degradation in COMPRESS from the link-capacity estimation over the path. For example, capacity estimation for 100-Mb/s and above links is prone to high errors on the Internet [22]. Because CoDe does not require link-capacity information and its performances presented in Figures 5 and 8 are consistently high, all these aspects suggest the robustness of the proposed scheme.

We also show a sample dataset from the first run of CoDe, using P_t = 1400 bytes, for estimating (25 ±18) µs of queuing delay in Figure 9. In addition, Table II presents a detailed

summary (i.e., member gaps, size, and centroid) of the three k-means clusters that we initially outlined in Figure 9. This table shows that all three clusters consist of only one member gap and their corresponding cluster sizes are relatively smaller than those in Table I. This phenomenon is primarily because of the high variability in the measured gaps, as Figure 9 shows. Also, there are multiple member gaps in each cluster that fell short of the required 20% rule during centroid calculation. For example, CoDe discarded four, three, fourteen member gaps from clusters 1, 2, and 3, respectively.

TABLE II: k-MEANS CLUSTERS IN FIGURE 9

Cluster ID	Cluster label	Member gaps	Cluster size	Cluster centroid
1	compression	184 µs	80	184 µs
2	no change	208 µs	146	208 µs
3	decompression	232 µs	100	232 µs

One essential requirement in estimating any network parameter through active means is to induce a small probing load so that the accuracy of measurement is not affected [18], [23]. CoDe fulfills this requirement in reference to the results presented in Figures 5 and 8. For example, both COMPRESS and CoDe generated a maximum of 4.78 Mb/s of probing load for measuring queuing delays using the largest P_t, i.e., 1400 bytes, in the packet pairs. This probing load is insignificant considering the 1-Gb/s and 100-Mb/s links used in the simulation setup. Moreover, CoDe generated all measurement results, presented in the above figures, over a small time period of 200 s. This implies that the proposed scheme can quickly measure queuing delay with a high accuracy using a small probing load.

IV. CONCLUSIONS

Queuing delay is a dynamic network parameter that is challenging to measure without substantial support from the networks. In this paper, we proposed a scheme to measure queuing delay on a router. The scheme uses an efficient unsupervised data-clustering algorithm to analyze the variations (i.e., no change, compression, and decompression) in the pairs of UDP packets for estimating the queuing delay on the router under test. It does not require prior knowledge of link capacities (i.e., infrastructural support) over an end-to-end path or of a specific variation (i.e., compression only) in the packet pairs for successful measurement. Moreover, it uses a fixed number of clusters in the clustering algorithm for effectively measuring queuing delay. As for a detailed evaluation, we simulated the proposed scheme over an end-to-end path, consisting of a single router, under different traffic conditions using the ns-3 simulator. Our results showed that the proposed scheme is robust, consistent, quick, and highly accurate without requiring any infrastructural support over the end-to-end path. For future work, we will explore the measurement of queuing delays on multiple routers over an end-to-end path using the proposed scheme.

REFERENCES

[1] M. Huang, W. Liu, T. Wang, H. Song, X. Li, and A. Liu, "A Queuing Delay Utilization Scheme for On-Path Service Aggregation in Services-Oriented Computing Networks," in *IEEE Access*, vol. 7, pp. 23816–23833, 2019.

[2] T. Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom, "Measuring Latency Variation in the Internet," in *Proc. of ACM CoNEXT*, 2016, pp. 473–480.

[3] B. Briscoe et al., "Reducing Internet Latency: A Survey of Techniques and Their Merits," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2149–2196, 2016.

[4] K. Salehin, R. Rojas-Cessa, and K. Kwon, "COMPRESS: A Self-Sufficient Scheme for Measuring Queueing Delay on the Internet Routers," in *Proc. of IEEE International Conference on Computing, Networking and Communications*, 2019, pp. 624–629.

[5] K. Salehin, K. Kwon, and R. Rojas-Cessa, "A Simulation Study of the Measurement of Queueing Delay Over End-to-End Paths," in *IEEE Open Journal of the Computer Society*, vol. 1, pp. 1–11, 2020.

[6] K. Salehin and R. Rojas-Cessa, "Scheme for Measuring Queueing Delay of a Router Using Probe-Gap Model: The Single-Hop Case," in *IEEE Communications Letters*, vol. 18, no. 4, pp. 696–699, 2014.

[7] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot, "Measurement and Analysis of Single-Hop Delay on an IP Backbone Network," in *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 908–921, 2003.

[8] L. Angrisani, G. Ventre, L. Peluso, and A. Tedesco, "Measurement of Processing Queueing Delays Introduced by an Open-Source Router in a Single-Hop Network," in *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 4, pp. 1065–1076, 2006.

[9] M. Haiyan, Y. Jinyao, P. Georgopoulos, and B. Plattner, "Towards SDN Based Queueing Delay Estimation," in *China Communications*, vol. 13, no. 3, pp. 27–36, 2016.

[10] B. Choi, S. Moon, Z. Zhang, K. Papagiannaki, and C. Diot, "Analysis of Point-to-Point Packet Delay in an Operational Network," in *Proc. of IEEE International Conference on Computer Communications*, 2004, pp. 1797–1807 vol.3.

[11] V. Jacobson, "Pathchar – A Tool to Infer Characteristics of Internet Paths." [Online]. Available: ftp://ftp.ee.lbl.gov/pathchar/msri- talk.pdf.

[12] K. Anagnostakis, M. Greenwald, and R. Ryger, "Cing: Measuring Network-Internal Delays Using Only Existing Infrastructure," in *Proc. of IEEE International Conference on Computer Communications*, 2003, pp. 2112–2121 vol.3.

[13] Y. Tsang, M. Coates, and R. Nowak, "Network Delay Tomography," in *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2125–2136, 2003.

[14] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proc. of Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[15] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An Efficient k-means Clustering Algorithm: Analysis and Implementation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[16] I. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," in *SN Computer Science*, vol. 2, no. 3, 160, 2021.

[17] M. Karim, et al., "Deep Learning-Based Clustering Approaches for Bioinformatics," in *Briefings in Bioinformatics*, vol. 22, no. 1, pp. 393–415, 2021.

[18] A. Downey, "Using Pathchar to Estimate Internet Link Characteristics," in *Proc. of ACM SIGCOMM*, 1999, pp. 241–250.

[19] ns-3 Network Simulator. Accessed on: Nov. 10, 2021. [Online] Available: https://www.nsnam.org/.

[20] S. Lloyd, "Least Squares Quantization in PCM," in *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982,

[21] D. Arthur and S. Vassilvitski, "k-means++: The Advantages of Careful Seeding," in *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027–1035.

[22] R. Prasad, C. Dovrolis, and B. Mah, "The Effect of Layer-2 Store-and-Forward Devices on Per-Hop Capacity Estimation," in *Proc. of IEEE International Conference on Computer Communications*, 2003, pp. 2090–2100 vol. 3.

[23] K. Salehin, V. Sahasrabudhe, and R. Rojas-Cessa, "Determination of Interrupt-Coalescence Latency of Remote Hosts Through Active Measurement," in *IEEE Access*, vol. 6, pp. 23019–23033, 2018.