Remote Measurement of Interrupt-Coalescence Latency of Internet Hosts

Khondaker M. Salehin
Illinois State University, USA
Email: kmsaleh@ilstu.edu

Vinitmadhukar Sahasrabudhe
P3 North America, USA
Email: sahasrabudhevinit@gmail.com

Roberto Rojas-Cessa
New Jersey Institute of Technology, USA
Email: rojas@njit.edu.

***Abstract*—Interrupt coalescence (IC) in the network interface card (NIC) of an Internet host is a hardware artifact to reduce the processing load of the host, but it also adds an additional delay in the processing time of a packet. As the transmission speeds of Internet hosts are increasing at a faster pace than the processing speeds of the hosts, NICs running at 100 Mb/s and above are adopting IC. This added delay from IC incurs a significant error to the measurement of various network parameters performed by Internet hosts. Knowledge of the IC-based delay helps to increase the accuracy of the measurement of several network parameters, including delay and bandwidth, and supports delay-sensitive applications, such as IP geolocation and load balancing. In this paper, we propose a scheme to measure the IC period of a remote Internet hosts using pairs of probing packets. The scheme does not require infrastructural support along the path or specialized instrumentation at the host under test. We evaluated the scheme on a testbed and an open-access campus network, on links of up to 1000 Mb/s. The scheme measures the IC period with 90% accuracy, quickly, and with a small probing load.**

***Index Terms*—Network measurement, interrupt coalescence, PPT, k-means clustering, end-to-end delay, IP geolocation.**

I. Introduction

Hosts are connected to the Internet for data communications through network interface cards (NICs), which transmit and receive packets. Once a NIC receives a packet, it notifies the event to its host by invoking system calls and interrupts involving the NIC driver, main memory, central processing unit (CPU), and operating system [1], [2]. The host then acknowledges the receiving process and generates a time stamp for the received packet [3].

Hosts used for high-speed data communications must process a packet at the same speed as the transmission of their NICs. However, it is difficult to comply with this required processing time because line rates are increasing at a faster pace than the processing speeds of hosts [4], [5]. Invoking frequent interrupts may induce heavy CPU loads that could result in performance degradation [6].

To reduce CPU load on a host, NICs are equipped with a hardware artifact called interrupt coalescence (IC) [7]–[10]. IC makes a NIC invoke a single interrupt after the arrival of multiple packets over a period of time. IC periods are found to be on the order of hundreds of microseconds [11]. However, this reduction of CPU load comes at the cost of adding a delay in the per-packet processing delay at the hosts.

ISBN 978-3-901882-94-4

The presence of IC in NICs may affect the measurement of various network parameters, such as delay and bandwidth, and the performance of delay-sensitive applications, such as IP geolocation and traffic load balancing. Therefore, it is essential to measure the IC period of Internet hosts. Although IC is widely deployed in high-speed NICs, no scheme is yet available to measure it, to the best of our knowledge.

To fill in this void, we propose a scheme to actively measure the IC period in the NIC of a remote host in this paper. The scheme is practical and easy to deploy because it does not require infrastructural support along the path of measurement or specialized instrumentation at the host under test. It uses pairs of packets, where the variations in the measured gaps infer the duration of the IC period. We determine the variations in the gaps using a k-means clustering algorithm [12].

In the remainder of the paper, Section II describes the operation of IC and the corresponding NIC timers. Section III outlines the applications of IC measurement. Section IV introduces the proposed scheme for measuring the IC period of a remote host. Section V presents experimental evaluations of the proposed scheme in a testbed and on an open-access campus network. Section VI presents our conclusions.

II. Background

The objective of IC in a NIC is to improve the efficiency of interrupt processing at a host by processing packets in periodic intervals (i.e., batches) at the cost of having an additional delay in processing packets. The duration of the IC period is set to minimize the load of the CPU generated by the frequent arrival of packets. As a result, packets are processed in batches after they arrive in the NICs [11].

A small IC period minimizes the per-packet delay without adding a heavy CPU load on the host for infrequent packet arrivals. On the other hand, a large IC period minimizes the CPU load by generating a single interrupt for multiple packets that arrive over a short interval under high transmission speeds. NIC drivers (e.g., Intel e1000e) implement three different timers to adjust the IC period for transmitting and receiving packets: i) the packet timer, ii) the absolute timer, and iii) the interrupt-throttle timer [11].

The packet timer resets (i.e., re-initializes) the IC period each time a packet arrives in the NIC buffer before the expiration of an existing IC period. This timer defines the minimum delay a NIC adds before generating an interrupt for

a packet [11]. Figure 1(a) presents an illustration of a packet timer as three packets, P_1, P_2, and P_3, arrive in the NIC. Here, the IC period is reset each time at the arrival of P_1, P_2, and P_3; an interrupt is finally generated after receiving P_3 because no other packet arrives before the expiration of the ongoing IC period. The packet timer is efficient when the transmission speed is low because under a high transmission speed, frequent resetting of the IC period may cause buffer overflow in NICs if the inter-arrival time of packets is smaller than the IC period [11].

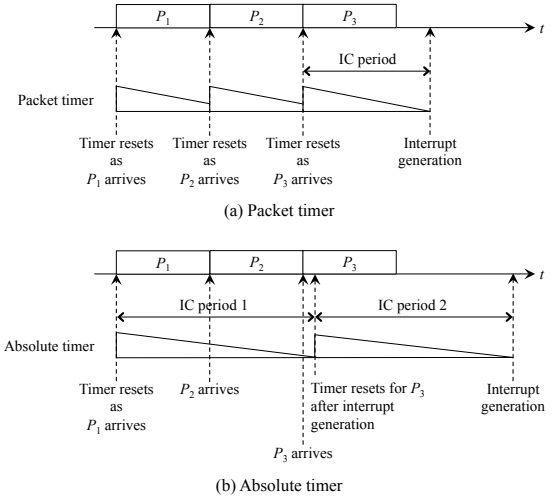


Fig. 1: IC periods using the (a) packet and (b) absolute timers.

The absolute timer determines the maximum delay before generating an interrupt after a packet arrives in the NIC buffer [11]. This timer does not reset the IC period at the arrival of every packet during an ongoing IC period. Figure 1(b) shows that IC period 1 is initiated at the arrival of P_1 and that the absolute timer does not reset before the expiration of IC period 1 though P_2 and P_3 start to arrive. IC period 2 is initiated after an interrupt is generated following the expiration of IC period 1, which acknowledges the first two packets, as P_3 arrives. The absolute timer is used under high transmission speeds [11].

The interrupt-throttle timer determines the IC period in a different manner from those of the above two. Instead of providing a single and discrete IC period, it defines the maximum number of interrupts that can be generated per unit time to determine an IC period in reference to the arrival rate of packets. Here, the IC period is reset periodically independently of packet inter arrivals as the timer does not initiate a new IC period before exhausting all (available) interrupts per unit time [11].

IC is configured as either a static or dynamic period. A static IC period is configured using the packet and/or absolute timer. In a dynamic IC configuration, the interrupt-throttle timer is used to determine an upper and lower ranges of the IC period, based on the arrival rate of packets at the NIC [11].

III. Application of IC Measurement

Knowledge of the IC period of a host is needed to achieve the correct measurement of different network parameters and applications. Here, we discuss some examples.

A. Delay Measurement

Delay over an end-to-end path is an important network parameter that is usually measured using Ping [13], King [14], and Ting [15]. However, delay measurement using existing tools are inaccurate because these tools do not consider the time stamping delay, i.e., the packet processing time (PPT), and IC period of Internet hosts [3], [10], [16]. The PPT of a host is the elapsed time as a packet travels from the data-link layer to the application layer of the TCP/IP protocol stack [17]. According to RFC 7679 [18], the measured *OWD* (*OWD′*) by Ping over a single-hop path between two end hosts src and dst is

$$OWD' = PPT_{src} + IC_{src} + OWD + PPT_{dst} + IC_{dst} \quad (1)$$

If the PPTs at the end hosts are PPT_{src} = PPT_{dst} = 2 μs and the IC periods are IC_{src} = 0 and IC_{dst} = 100 μs, the measured delay *OWD′* = 181 μs when the actual delay *OWD* = 77 μs given that the transmission time of P (t_t) = 12 μs, the average queueing delay (t_q) = 40 μs [19], and the propagation time (t_p) = 25 μs, considering a 5-km optical (Gigabits) link between src and dst. Here, the measurement error = $|\frac{OWD'-OWD}{OWD}|$ = 135%. This error in the delay measurement can only be alleviated if knowledge of PPT and IC period are available in advance.

In addition, the utility of IC measurement in delay estimation has far-reaching impacts on various delay-sensitive applications of the Internet. For example, precise delay measurements are essential for producing high accuracy in IP-geolocation schemes [20]–[22], and financial trading [23], [24]. In case of IP geolocation, a 1-μs error in the measured delay varies the estimated geographic distance by 200 m between two end hosts connected over optical links, where data travel $\frac{2}{3}$ the speed of light in vacuum [25]. Therefore, IC measurement is important because IC periods are on the order of hundreds of microseconds.

B. Bandwidth Measurement

The knowledge of the IC period of a host can be used to accurately measure bandwidth, e.g., link capacity and available bandwidth, of Internet links [7], [8]. Consider a 1000-Mb/s link between src and dst where the capacity of the link is under measurement using a pair of probing packets, each with a length of 1500 bytes. The intra-probe gap between the packets at dst is must be proportional to the link capacity for accurate measurement; the expected intra-probe gap is

$$E[G] = \frac{\text{Packet size}}{\text{Link capacity}} = \frac{1500 \text{ bytes}}{1000 \text{ Mb/s}} = 12 \ \mu\text{s} \quad (2)$$

However, ensuring the proportional gap in (2) is infeasible because dst has an IC period, $IC_{dst} = 100$ μs, which is larger

than $E[G]$. The presence of IC eliminates the expected gap because the packets are buffered in the same IC period and this gap loss generates errors in link-capacity measurement [7], [8]. With prior knowledge of the IC period, it would be possible to determine an optimal packet size for producing high accuracy in packet-pair-based measurement schemes [26]–[30].

Additionally, the schemes that use trains of probing packets, e.g., Pathrate [31], IGI [32], pathChirp [33], and cartouche [34] may utilize the knowledge of the IC period to improve their measurement efficiency. Probing-train based schemes uses an arbitrary train length, which contributes to the large probing traffic in the network. Such practice in bandwidth measurement can be avoided by determining an optimal (i.e., minimum) train size without compromising accuracy if the IC period is known a priori.

C. Load Balancing of Traffic

Load balancing of traffic allows efficient usage of Internet resources by splitting end-to-end traffic flows among multiple routes at different granularities, e.g., per packet and per destination-address. However, load balancing schemes usually suffer from packet reordering at the end hosts. To eliminate packet reordering, a study proposes a traffic-splitting scheme using the burstiness of Internet traffic that can be generated by the IC periods of the end hosts [35]. NIC IC usually generates traffic bursts in 10s of packets in length in the Internet [9]. The study assumes the burstiness of traffic in their experimental evaluation without actually characterizing it. The performance of the proposed scheme can be improved by characterizing the level of burstiness of traffic flows using the knowledge of IC.

IV. Proposed Scheme

We are aware of only two prior work that were motivated to detect the presence of IC in NICs [8], [10] at the time of writing this paper, but there is no scheme that can measure IC period to the best of our knowledge. In this paper, we propose an active scheme to measure the IC period of a remote host over an end-to-end path. The scheme uses a packet-pair structure, called compound probe [3], [36], [37], and estimates the IC period by measuring the variations of the intra-probe gap of the probing structure at the host under test. The compound probe consists of user datagram protocol (UDP) packets, and the intra-probe gap is measured by a software process on the host. Hence, the proposed scheme does not require infrastructural support from the network or instrumentation at the host.

Figure 2 shows a compound probe that consists of a large heading packet (P_h) and a small trailing packet (P_t), and they are being received by the NIC of a destination host. Here, both P_h and P_t arrive in the NIC back-to-back (i.e., without any separation in between) and are buffered at the input queue inside a single IC period. After the expiration of the IC period, two consecutive time stamps, TS_h and TS_t, are generated for P_h and P_t, respectively, with a processing delay, or the time-stamp generation time, on the receiving host to finish the receiving process. Because the intra-probe gap of the compound probe (i.e., the time difference between the last bits of P_h and P_t) is determined by using the time stamps, the measured intra-probe gap corresponds to the processing delay or PPT of the destination host rather than the actual transmission time of P_t [10], as Figure 2 shows. The disproportionate and small intra-probe gap in the compound probe suggests the presence of IC in the NIC [10].

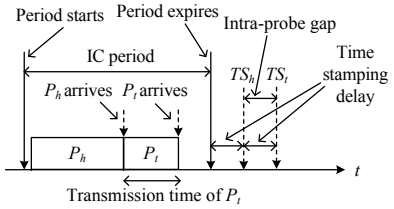


Fig. 2: Buffering of a compound probe during an IC period.

Figure 3 illustrates another scenario where P_h and P_t arrive in two consecutive IC periods, IC periods 1 and 2, as the packets are received in the NIC. Here, TS_h and TS_t are generated after the expiration of IC periods 1 and 2, respectively. In this figure, both the initiation of IC period 2 and the time stamping of P_t are shown as two separate but parallel events. Because TS_h and TS_t are interleaved by IC period 2, the measured intra-probe gap of the compound probe is equal to the IC period and larger than the intra-probe gap measured in Figure 2.

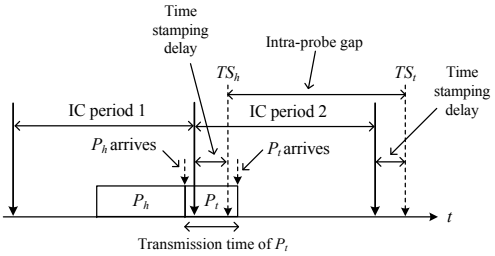


Fig. 3: The measured intra-probe gap of a compound probe when P_h and P_t arrive in two consecutive IC periods.

Considering these two possible scenarios, the proposed scheme at a sender host sends compound probes with a constant P_t size and an increasing P_h sizes to the remote host under test so that the packets arrive in consecutive IC periods of its NIC. The scheme adjusts the P_h size in an iterative manner during the course of measurement. P_h and P_t fall into consecutive IC periods only if the transmission time of P_h is smaller, but the combined transmission time of P_h and P_t is larger than the IC period.

Figure 4 shows the variations of the measured intra-probe gaps when the propose scheme sends compound probes with increasing P_h sizes. Here, the small gaps (identified as IC-detection phase) correspond to the scenario depicted in Figure 2 and the large gaps (identified as transition phase) correspond to the scenario depicted in Figure 3. The proposed scheme first identifies the IC-detection phase from the smaller intra-probe

gaps. It then detects the transition phase in reference to the smaller gaps for measuring the IC period at the remote host. In the IC-detection phase, the scheme estimates the PPT of the remote host, as discussed above.

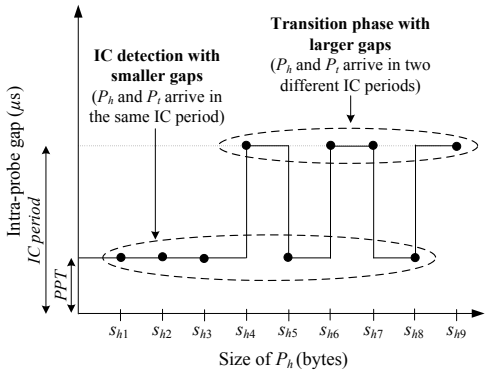


Fig. 4: Variations of the intra-probe gap for compound probes with different P_h sizes.

Figure 5 shows an n-hop path between src and dst, the sender host and the remote host under test, respectively. The detailed steps for measuring the IC period at dst from src are described below:

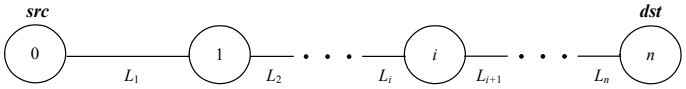


Fig. 5: An n-hop path between src and dst.

1) Send a train of i compound probes from src to dst using a P_h with a size of s_h, e.g., s_h = the maximum transmission unit (MTU) of the path, and a P_t with a size of $s_t < s_h$, such that the packet-size ratio of the compound probes, $\alpha = \frac{s_h}{s_t}$, is large enough to ensure back-to-back arrivals of the probing packets. The sizing of s_t in the compound probe for back-to-back arrival of P_h and P_t at dst over an n-hop path is determined as
$$(\frac{s_h}{l_n} - \frac{s_h}{\alpha l_{n-1}}) + (\frac{s_h}{l_{n-1}} - \frac{s_h}{\alpha l_{n-2}}) + ... + (\frac{s_h}{l_{z+1}} - \frac{s_h}{\alpha l_z}) = 0 \quad (3)$$
Here, l_z and l_{z+1} are the capacities of the input and output links at a node z over the n-hop path, where P_h and P_t experience the largest dispersion of the path [37].
2) Measure the intra-probe gap, $G(s_t)$, of each compound-probe in the train received at dst:
$$G = \{G_j(s_t) \,|\, j \leq i\}, \quad (4)$$
where G is the required dataset of intra-probe gaps for estimating the IC period at dst.
3) Determine the distribution of the data set, G, using the k-means clustering algorithm such that the resultant cluster set is
$$C' = \{C_k \,|\, k > 1\}, \quad (5)$$
where $C_k \subset G$ and $C_k \cap C_{k+1} = \emptyset$.
4) Update the cluster set C' after discarding the cluster elements that have a population size smaller than a portion p of the size of G as
$$C = \{C_k \,|\, (|C_k| \geq \frac{p|G|}{100})\} \quad (6)$$
Here, C is the updated cluster set where $|C| \leq |C'|$.
5) Consider v_k to be the centroid (i.e., average intra-probe gap) of C_k. With $V = \{v_k \,|\, k \leq |C'|\}$ being the corresponding set of all centroids of the cluster set C, determine the cluster with the smallest centroid, v_s, in C to estimate the PPT at dst as
$$PPT = v_s = min\{V\} \quad (7)$$
Here, v_s corresponds to the case illustrated in Figure 2.
6) Calculate the difference between the smallest centroid value and each of the remaining centroid values in V:
$$\Delta = \{\Delta_{s,k} = v_k - v_s \,|\, (v_s \neq v_k) \wedge (k \leq |V| - 1)\} \quad (8)$$
7) Discard all data elements such that $\Delta_{s,k} < \Gamma$ to check if P_h and P_t arrive in consecutive IC periods, as illustrated in Figure 3, and update Δ as
$$\Delta = \{\Delta_{s,k} \,|\, (\Delta_{s,k} \geq \Gamma) \wedge (k \leq |V| - 1)\}, \quad (9)$$
where Γ is the minimum threshold defined by the minimum IC period of the NIC at dst. For example, the minimum IC period in Linux and Windows is approximately 50 μs [11].
8) If $\Delta = \emptyset$, none of the centroid values exceed the minimum threshold values, as Figure 6 shows. This condition also means that the compound probes using $P_h = s_h$ did not detect a transition phase in the measured intra-probe gaps, as illustrated in Figure 4. Repeat Steps 1 to 7 using $P_h = s_h + \tau$ and the same $P_t = s_t$ in the compound probe. Here, τ, the step size for s_h, can be determined from the clock resolution and the transmission speed of the NIC at dst [3], [37].
9) Else, at least one or more centroid values exceed the minimum threshold value, as Figure 7 shows; therefore, determine the IC period at dst as
$$IC = \{v_k \,|\, \Delta_{s,k} = min\{\Delta\}\} \quad (10)$$

V. Experimental Evaluations

We evaluated the proposed scheme on both a controlled testbed and an uncontrolled intranet environments. The scheme was implemented in a Linux (Ubuntu) operating system that consists of a client process at src and a server process at dst over an end-to-end path, as Figure 5 shows. In our experiments, we used two different workstations: Dell Optiplex 790 (DO790) and Dell Studio XPS 435 MT (DS435). They are both equipped with integrated Intel NICs, whose IC period is configurable in Linux environment [11]. The detailed specifications of the workstations are presented in Table I.

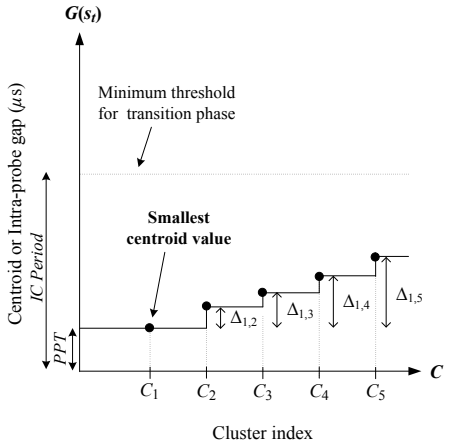


Fig. 6: An example where no cluster centroids exceed the minimum threshold because $\Delta_{1,k} < \Gamma$, where $2 < k \leq 5$ considering $|C| = 5$.

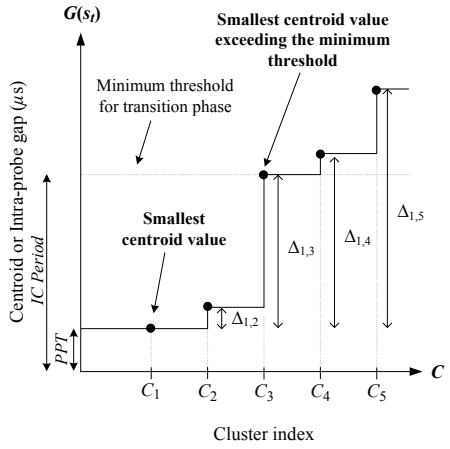


Fig. 7: An example where three cluster centroids exceed the minimum threshold because $\Delta_{1,k} \geq \Gamma$, where $3 < k \leq 5$ considering $|C| = 5$.

TABLE I: Workstation Specifications

Name	DO790	DS435
Processor (speed)	Intel Core i3 (3.30 GHz)	Intel Core i7 (2.67 GHz)
RAM (speed)	8148 MB (1333 MHz)	8725 MB (1066 MHz)
NIC (driver)	Intel 82579LM (e1000e)	Intel 82567LF-2 (e1000e)
Max. transmission speed	1000 Mb/s	1000 Mb/s
Linux distribution	Ubuntu	Ubuntu
Linux kernel	2.6.18	3.11.0

A. End-to-End Paths and IC Configurations

We measured the IC periods of DO790 and DS435 under two different transmission speeds, 100 and 1000 Mb/s, at dst over single- and multiple-hop paths. For example, both a single-hop path (src and dst directly connected by a 100- or 1000-Mb/s link) and a 4-hop path (src and dst connected by 100-Mb/s, 100-Mb/s, 155-Mb/s, and 100-Mb/s or 1000-Mb/s links) were used in the testbed experiments. The 4-hop path consisted of two Cisco 7200 routers and one Cisco 3600 router between the end hosts. The testbed experiments were performed without having cross-traffic loads along the paths; these experiments provided a benign network environment during IC measurement to determine the maximum achievable performance of the proposed scheme.

In the intranet experiments, a multiple-hop path, consisting of at least 4 hops, between the Electrical and Computer Engineering (ECE) and Faculty Memorial Hall (FMH) buildings at New Jersey Institute of Technology (NJIT) was used. The infrastructure (e.g., link capacity, routing policy, and queue length of intermediate routers) and traffic (e.g., traffic load and number of flows) conditions over this path were unknown while running the experiments. This set of experiments evaluated the maximum achievable performance of the proposed scheme, determined by the testbed experiments, in an uncontrolled environment. We evaluated the scheme in NJIT's open-access network for a month during regular working hours.

In both sets of experiments, we measured the IC periods of DO790 and DS435 under two different IC settings: static and dynamic. In the static setting, the workstation at dst was manually configured with three different fixed IC periods: 125, 166, and 200 μs. These periods were chosen to verify the sensitivity of the proposed scheme, and they were all within the min-max range of the static-IC settings [11]. In the case of the dynamic setting (also known as Mode 3 or Interrupt Throttle Rate, ITR3), the workstation at dst was configured with an IC period that ranges between 50 and 250 μs, which is the default ITR setting in Linux [11].

B. Probing Parameters and Clustering Algorithm

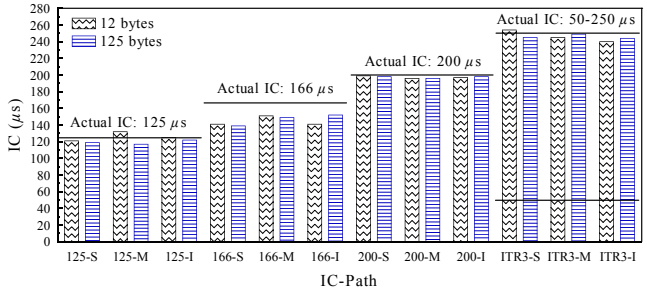
The proposed scheme measures IC of each workstation in all experiments using 50 and 200 compound probes per train. We used these train sizes to verify the performance of the scheme under two different probing loads. Compound probes in each train are separated by 11 to 99 ms to avoid generating probing traffic with a constant rate in the end-to-end path and at dst. In the compound probes, we initially used s_h = 1,492 bytes and s_t = 150 bytes under 100 Mb/s or s_t = 1,250 bytes or less under 1000 Mb/s. The selected s_ts ensure a large α in the compound probe for each speed and considering the limited clock resolution in Linux [38]. The theoretical intra-probe gaps in the compound probes (i.e., $\frac{s_t}{l_n}$ [3]) were 12 and 10 μs for the respective speeds.

In each experiment, we started the measurement process with the initial s_h and s_t and then continued to send probing trains with increasing s_hs until five successful measurements of the IC period were achieved to calculate an average estimate. The client process increases s_h with either 12- or 125-byte step size following the completion of sending a probing train. These two step sizes were chosen considering the NIC speeds, 100 and 1000 Mb/s, respectively, and the limited clock resolution (usually 1 μs) in Linux system [38]. In case the scheme failed to achieve five successful measurements when s_h = 64,000 bytes (i.e., the maximum allowable packet size), the client process ended the measurement process, and calculated an average IC estimate from the available successful measurements.

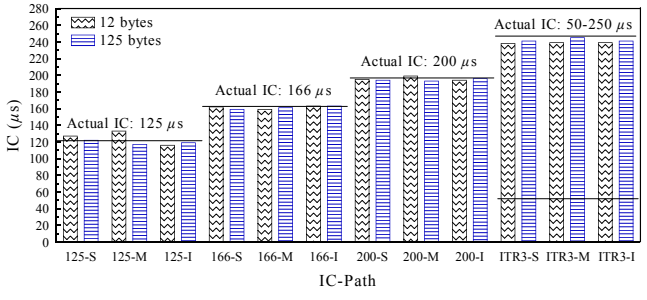
We used the VLFeat open-source library [39] to implement the k-means algorithm and incorporated it in the client process at src. In the k-means algorithm, the clusters were filtered in Step 3 of the proposed scheme using a population size p = 5, which we chose empirically from our measurement experiences, because it provides a high accuracy. For data filtering, the scheme ran the k-means clustering algorithm on each dataset for 100 iterations using a random seed value, as implemented in the VLFeat library, before producing an initial set of clusters C'. The scheme also used a minimum threshold value Γ = 50 μs for detecting a transition phase in the measured intra-probe gaps.

C. Measurement Results under 100 Mb/s

Figure 8 shows the summary of IC periods measured on (a) DO790 and (b) DS435 under 100 Mb/s using a train size of 200 compound probes and two different step sizes τ = {12, 125} bytes. In the remainder of the paper, we present measurement results of only 200-probe trains because this train size produces better performance than 50-probe trains. In Figure 8, the x-axis corresponds to different IC-Path configurations, where IC = {125, 166, 200, ITR3} and Path = {testbed single-hop path (S), testbed multiple-hop path (M), intranet multiple-hop path (I)}. The y-axis corresponds to the measured IC period by the proposed scheme. In addition, the horizontal lines in the figure shows the actual IC periods in microseconds for static ICs in the experiments. In case of dynamic IC (i.e., ITR3), the two parallel horizontal lines correspond to the upper and lower range of the configured IC period.



(a) DO790

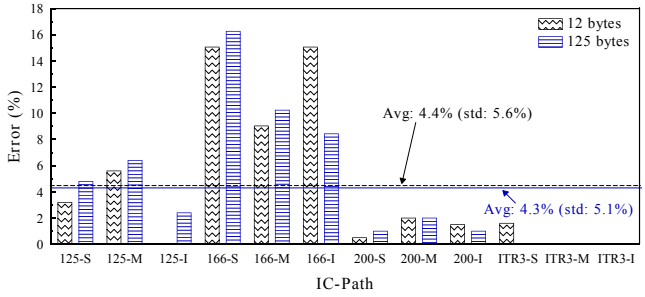


(b) DS435

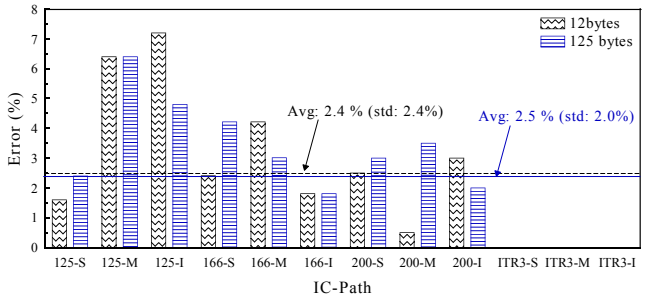
Fig. 8: Measured IC periods under 100 Mb/s on the (a) DO790 and (b) DS435 workstations for different IC configurations over the testbed and intranet paths.

According to Figure 8(a), the measured IC periods on DO790 are very close to the actual IC periods for all experimental configurations. For example, the proposed scheme consistently measures IC periods of approximately 120, 140, and 200 μs when the actual static IC periods are 125, 166, and 200 μs, respectively, over all path configurations. In the case of dynamic IC, the measured IC period is approximately 245 μs, which is within the range of the IC period of ITR3, i.e., between 50 and 250 μs, and close to the upper range of the IC period, on both testbed and intranet. Figure 8(b) shows that the measured static IC periods on DS435 are about 120, 160, and 197 μs, respectively, whereas the dynamic IC period is about 240 μs, which is, again, close to the upper range of the ITR3 setting. Overall, Figure 8 shows that the proposed scheme is consistent in measuring ICs on both workstations under different experimental conditions.

We summarize the consistency of the IC measurements in Figure 9 using the measurement errors on (a) DO790 and (b) DS435. Error is defined as $\left|\frac{\text{IC Period} - \text{Measured IC Period}}{\text{IC Period}}\right| \times 100\%$, which is plotted along the y-axis in Figure 9. However, for dynamic IC, the error is determined by considering how far off the measured IC period is from the expected range.



(a) DO790



(b) DS435

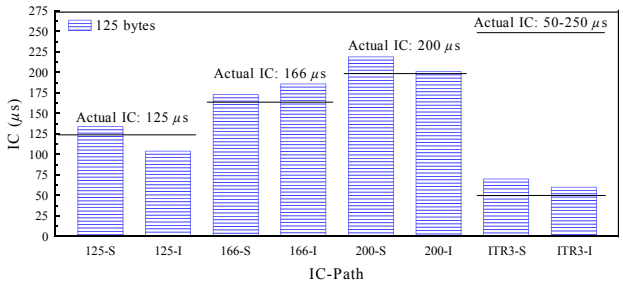
Fig. 9: Measurement errors under 100 Mb/s on the (a) DO790 and (b) DS435 workstations for different IC configurations over the testbed and intranet paths.

Figure 9 shows that the measurement errors are consistently small on both workstations to validate the high accuracy of the proposed scheme irrespective of the experimental configurations. For example, the average measurement errors on DO790 with τ = {12, 125} bytes are 4.3 and 4.4%, respectively, as indicated by the dashed- and solid-horizontal lines in Figure 9(a). Here, the standard deviations are 5.6 and 5.1%,

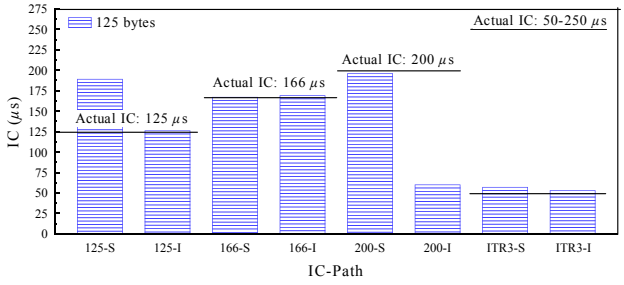
respectively. These large standard deviations are due to the relatively high measurement errors (approximately 12% on average) for IC period = 166 μs. On DS435, the average errors with $\tau = \{12, 125\}$ bytes are 2.4 and 2.5% with standard deviations of 2.4 and 2%, respectively, as Figure 9(b) shows. Although the standard deviations in the above measurements are large, the accuracy of the proposed scheme is high as the average errors on DO790 and DS435 are below 5%.

D. Measurement Results under 1000 Mb/s

Figure 10 summarizes the IC measurements on (a) DO790 and (b) DS435 under 1000 Mb/s. In this set of experiments, we measured both static and dynamic IC periods, over the single-hop testbed and multiple-hop intranet paths using a single step size, i.e., τ = 125 bytes, since the transmission time of 125 bytes under 1000 Mb/s is 1 μs. Here, we used s_t = 1,250 bytes over the single-hop path and $s_t = \{200, 250\}$ bytes for DO790 and DS435, respectively, over the multiple-hop path. The large s_t = 1,250 bytes over the single-hop path provided a large intra-probe gap (i.e., 10 μs) similar to that used in the experiments under 100 μs, and the smaller s_ts were the lower bounds of s_t that ensured zero-dispersion gap in the compound probe over the multiple-hop path.



(a) DO790

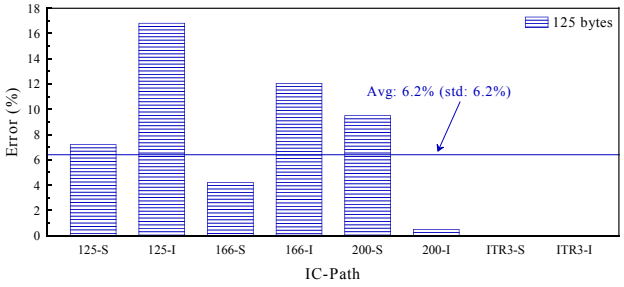


(b) DS435

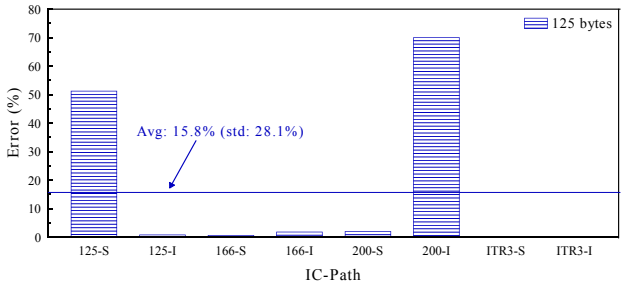
Fig. 10: Measured IC periods under 1000 Mb/s on the (a) DO790 and (b) DS435 workstations for different IC configurations over the testbed and internet paths.

Figure 10(a) shows that the measured IC periods on DO790 are approximately 117, 182, and 210 μs for static ICs of 125, 166, and 200 μs, respectively. For dynamic IC, the measured values are approximately 67 μs, which is within the ITR3 range and close to the lower range of the IC setting. Figure 10(b) shows the static IC periods on DS435 are measured about 160, 166, and 200 (excluding over 200-M) μs, respectively. For dynamic IC, the values are about 57 μs, which is within the expected range and close to the lower range of the IC setting.

Figure 11 presents a summary of measurement errors under 1000 Mb/s. Here, the average errors on DO790 and DS435 are 6.2 and 15.8%, respectively, with standard deviations equal to 6.2 and 28.1%, respectively. The relatively large errors with 1000 Mb/s suggests that the IC measurement on 1000 Mb/s is more challenging than with 100 Mb/s.



(a) DO790



(b) DS435

Fig. 11: Measurement errors under 1000 Mb/s on the (a) DO790 and (b) DS435 workstations for different IC configurations over the testbed and intranet paths.

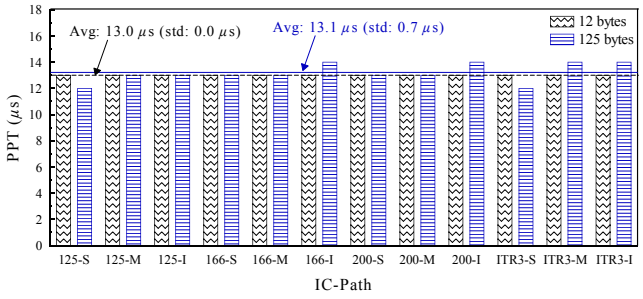
IC presents the most challenging behavior under 1000 Mb/s [40]. Challenges in the measurement of IC period under 1000 Mb/s are attributed to two factors. First, s_t impacts the accuracy of IC measurement under higher transmission speeds because of smaller intra-probe gaps in the compound probe, which is proportional to s_t and the transmission speed [3]. These smaller gaps minimize the probability of having P_h and P_t in two consecutive IC periods. This phenomenon is perceived from the average errors on DS435 for the 125-S and 125-I configurations and on DO790 for the 200-S and 200-I configurations, as Figure 11 shows. Second, the condition of five successful measurements using $p = 5$ impacts the outcome of the proposed scheme. IC measurement using the 5% rule was successful only for a number of configurations due to the minimized probability of having the probing packets in two consecutive IC periods. For example, the measured values and average errors for the ITR3-S and ITR3-I configurations in Figures 10(a) and 11(a), respectively, are generated using $p = 3$. In case of Figures 10(b) and 11(b), the values for 125-S and 200-S are generated using $p = 4$ and $p = 2$, respectively.

We consider that the above challenges can be reduced with larger s_h and s_ts, which would ensure both a larger intra-

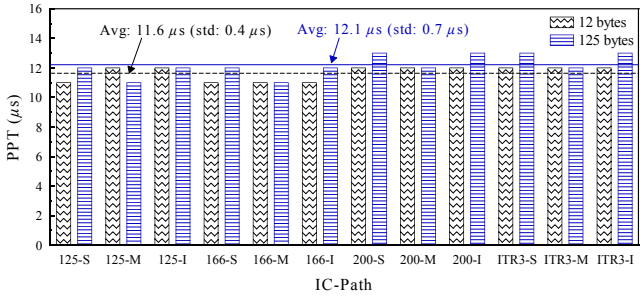
probe gap in the compound probe and a zero dispersion gap over an end-to-end path. Such considerations are feasible if we use larger packet sizes in the compound probe using the IPv6 infrastructure in the Internet.

E. PPTs of the Workstations

The proposed scheme estimates PPT after detecting IC in the NIC as the initial step before measuring the IC period on a workstation. Figure 12 presents the estimated PPTs during IC measurements under 100 Mb/s. As Figure 12(a) shows, the average PPTs estimated on DO790 are approximately 13 μs for $\tau = \{12, 125\}$ bytes. Figure 12(b) shows that the average PPTs of DS435 are approximately 12 μs. The standard deviation in the average PPTs is less than 1 μs.



(a) DO790

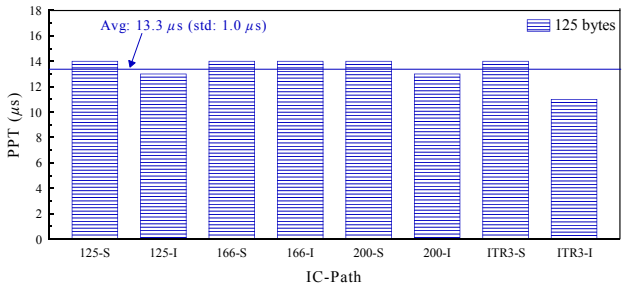


(b) DS435

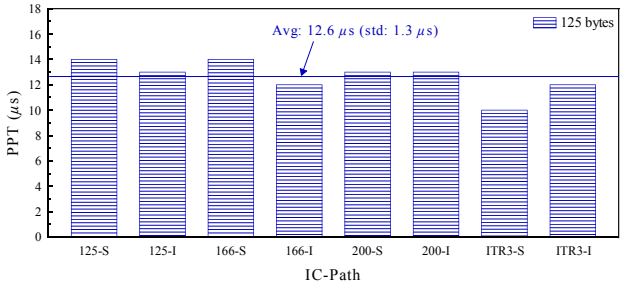
Fig. 12: Estimated PPTs under 100 Mb/s on (a) DO790 and (b) DS435 workstations for different IC configurations over the testbed and intranet paths.

Figure 13 presents the average PPTs under 1000 Mb/s. The solid-horizontal lines in Figures 13(a) and 13(b) show that the average PPTs of DO790 and DS435 are both approximately 12 μs. Here, the standard deviations of the estimated PPTs are up to 1.3 μs. However, such a large variation on a higher transmission speed is coming from the limited clock resolution in Linux system [3].

In a prior work, PPT of both DO790 and DS435 were measured with a dynamic IC (i.e., ITR3), where the values are estimated as 3 μs under 100 Mb/s and 10 μs under 1000 Mb/s, respectively [10]. According to Figures 12 and 13, the estimated PPTs are different than those in the prior work, especially on DO790. We consider the estimated values in this paper are more reliable because the prior work used ICMP probes and utilized only 500 packets (or ten PPT estimates) during the measurements [10].



(a) DO790



(b) DS435

Fig. 13: Estimated PPTs under 1000 Mb/s on the (a) DO790 and (b) DS435 workstations for different IC configurations over the testbed and intranet paths.

F. Measurement Time

To explore the efficacy of the proposed scheme, we estimated the average time required for measuring IC on each workstation. For example, the scheme required 72 and 57 s for measuring IC period on DO790 using 50 compound probes when $\tau = \{12, 125\}$ bytes, respectively. On DS435, the average measurement times are approximately 23 and 50 s, respectively. These values suggest that the proposed scheme measures IC period in a short time. They also suggest that the measurement time decreases as the step size increases, i.e., from 12 to 125 bytes, without losing any performance gain, as Figure 9 shows.

We found that the scheme requires a longer time for measuring IC under 1000 Mb/s since the average measurement time on DO790 and DS435 are 180 and 140 s, respectively, using $\tau = 125$ bytes. This behavior is expected considering the challenges discussed in Section V-D.

G. Probing Load

We also calculated the probing loads generated by the proposed scheme during IC measurements since active measurement schemes are considered intrusive for Internet traffic [41]. In the above experiments under 100 Mb/s, the scheme generates probes at the average rates of 0.3 and 0.2 Mb/s on DO790 and DS435 workstations, respectively. Under 1000 Mb/s, these rates are 1.7 and 1.2 Mb/s, respectively. These small probing loads show that the proposed scheme is not intrusive to the data traffic on an end-to-end path because the scheme utilizes less than 1% of the above stated transmission speeds.

VI. Conclusions

IC is a hardware artifact of the NIC of an Internet host that delays the processing of a packet in order to reduce CPU load. High-speed NICs are equipped with an IC period on the order of 100s of microseconds, which can affect different network parameters and applications if neglected. In this paper, we proposed an active scheme to measure IC period in the NIC over an end-to-end path. The proposed scheme is practical and easy to deploy because it does not require infrastructural support along the path of measurement or instrumentation at the host under test. The scheme sends pairs of probing packets at the host to determine the intra-probe gaps in the packet pairs. The measurement method uses a k-means clustering algorithm to determine variations in the measured gaps, which are used to infer the duration of IC period.

We evaluated the proposed scheme to measure different IC periods on two different workstations, in both controlled and uncontrolled environments and both without and with real-life cross traffic, respectively. The experimental results showed that our scheme measured IC periods over an end-to-end path with a high accuracy, e.g., approximately 5% and 11% error under 100 and 1000 Mb/s, respectively, regardless of the used workstation and existing network conditions. The scheme also measures IC period in a short time and generates a small amount of probing load into the network.

References

[1] P. Willman, H. Kim, S. Rixner, and V. Pai, "An efficient programmable 10 Gigabit Ethernet network interface card," in Proc. *IEEE HPCA-11*, CA, USA, 2005, pp. 96–107.

[2] J. Shafer and S. Rixner, "A reconfigurable and programmable gigabit ethernet network interface card," Rice University – Department of Electrical and Computer Engineering, Tech. Rep., December 2006.

[3] K. Salehin, R. Rojas-Cessa, C. Lin, Z. Dong, and T. Kijkanjanarat, "Scheme to measure packet processing time of a remote host through estimation of end-link capacity," *IEEE TC*, vol. 64, no. 1, pp. 205–218, 2015.

[4] N. McKeown. High performance routers – Talk at IEE, London UK. October 18th, 2001. [Online]. Available: http://tinytera.stanford.edu/ nickm/talks/index.html.

[5] S. Savage. IP router design. [Online]. Available: http://cseweb.ucsd.edu/classes/wi05/cse123a/Lec8.pdf.

[6] J. Mogul, D. Western, J. Mogul, and K. Ramakrishnan, "Eliminating receive livelock in an interrupt-driven kernel," *ACM TOCS*, vol. 15, pp. 217–252, 1997.

[7] G. Jin and B. Tierney, "System capability effects on algorithms for network bandwidth measurements," in Proc. *ACM IMC*, USA, 2003, pp. 27–38.

[8] R. Prasad, M. Jain, and C. Dovrolis, "Effects of interrupt coalescence on network measurements," in Proc *PAM*, France, 2004, pp. 247–256.

[9] R. Kapoor, A. Snoeren, G. Voelker, and G. Porter, "Bullet Trains: A study of NIC burst behavior at microsecond timescales", in Proc. *ACM CoNEXT*, USA, 2013, pp. 133–138.

[10] K. Salehin and R. Rojas-Cessa, "Measurement of packet processing time of remote hosts in the presence of interrupt coalescence," in Proc. *IEEE HPSR*, Japan, 2016, pp. 144–149.

[11] Intel. Interrupt moderation using Intel GbE controllers. [Online]. Available: http://www.intel.com/content/www/us/en/ethernetcontrollers/gbe-controllers-interrupt-moderation-appl-note.html.

[12] E. Alpaydin, *Introduction to Machine Learning, ch. 7*. MA, USA: The MIT Press, 2010.

[13] B. Forouzan, *TCP/IP Protocol Suit, ch. 9*. NY, USA: McGraw Hill, 2010.

[14] K. Gummadi, S. Saroiu, and S. Gribble, "King: Estimating latency between arbitrary Internet end hosts," in Proc. *ACM IMW*, 2002, pp. 5–18.

[15] F. Cangialosi, D. Levin, and N. Spring, "Ting:Measuring and exploiting latencies between all tor nodes," in Proc. *ACM IMC*, Japan, 2015, pp. 289–302.

[16] D. Frost and S. Bryant, "Packet loss and delay measurement for MPLS networks," RFC 6374, September 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6374.txt.

[17] K. Salehin, R. Rojas-Cessa, and S. Ziavras, "A method to measure packet processing time of hosts using high-speed transmission lines," *IEEE SJ*, vol. 9, no. 4, pp. 1248–1251, 2014.

[18] G. Almes, S. Kalidindi, M. Zekauskas, and A. Morton, Ed., "A one-way delay metric for IPPM," RFC 7679, January 2016. [Online]. Available: http://www.ietf.org/rfc/rfc7679.txt.

[19] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, and C. Diot, "Measurement and analysis of single-hop delay on an IP backbone network," *IEEE JSAC*, vol. 21, no. 6, pp. 908–921, 2003.

[20] E. Katz-Bassett, J. John, A. Krishnamurthy, T. Anderson, and Y. Chawathe,"Towards IP geolocation using delay and topology measurements," in Proc. *ACM IMC*, USA, 2006, pp. 71–84.

[21] V. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for Internet hosts," in Proc. *ACM SIGCOMM*, USA, 2001, pp. 173–185.

[22] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of Internet hosts," *IEEE/ACM ToN*, vol. 14, no. 6, pp. 1219–1232, 2006.

[23] Wall street's quest to process data at the speed of light. [Online]. Available: http://www.informationweek.com/news/199200297?pgno=1.

[24] R. Kompella, K. Levchenko, A. Snoeren, and G. Varghese, "Every microsecond counts: Tracking fine-grain latencies with a lossy difference aggregator," in Proc. *ACM SIGCOMM*, Spain, 2009, pp. 255–266.

[25] R. Percacci and A. Vespignani, "Scale-free behavior of the Internet global performance," *EPJ B*, vol. 32, no. 4, pp. 411–414, 2003.

[26] R. Carter and M. Crovella, "Measuring bottleneck link speed in packet switched networks," *Performance evaluation*, vol. 27 and 28, pp. 297–318, 1996.

[27] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in Proc. *USITS*, 2001, pp. 123–134.

[28] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in Proc. *ACM IMC*, 2003, pp. 39–44.

[29] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi, "CapProbe: A simple and accurate capacity estimation technique," in Proc. *ACM SIGCOMM*, USA, 2004, pp. 67–78.

[30] A. Pasztor and D. Veitch, "Active probing using packet quartets," in Proc. *ACM IMC*, USA, 2002, pp. 293–305.

[31] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet dispersion techniques and capacity estimation," *IEEE ToN*, vol. 12, no. 6, pp. 963–977, 2004.

[32] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE JSAC*, vol. 21, no. 6, pp. 879–894, 2003.

[33] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathchirp: Efficient available bandwidth estimation for network paths," in Proc. *PAM*, vol. 4, 2003, pp. 1–11.

[34] K. Harfoush, A. Bestavros, and J. Byers, "Measuring capacity bandwidth of targeted path segments," *IEEE/ACM ToN*, vol. 17, no. 1, pp. 80–92, 2009.

[35] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *SIGCOMM CCR*, vol. 37, no. 2, pp. 51–62, 2007.

[36] K. Salehin and R. Rojas-Cessa, "Combined methodology for measurement of available bandwidth and link capacity in wired packet networks," *IET Commun.*, vol. 4, no. 2, pp. 240–252, 2010.

[37] —-, "Packet-pair sizing for controlling packet dispersion on wired heterogeneous networks," in Proc. *IEEE ICNC*, USA, 2013, pp. 1031–1035.

[38] D. Bovet and M. Cesati, *Understanding the Linux Kernel, ch. 5*. CA, USA: O'Reilly, 2001.

[39] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. [Online]. Available: http://www.vlfeat.org/.

[40] A. Chatzipapas and V. Mancuso, "Measurement-Based Coalescing Control for 802.3az", in Proc. *IFIP Networking*, Austria, 2016, pp. 270–278.

[41] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and k. claffy, "Comparison of public end-to-end bandwidth estimation tools on high-speed links," in Proc. *PAM*, 2005, pp. 306–320.