Noise Can Help: Accurate and Efficient Per-flow Latency Measurement without Packet Probing and Time Stamping

Muhammad Shahzad and Alex X. Liu
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan, USA
{shahzadm, alexliu}@cse.msu.edu

ABSTRACT

With the growth in number and significance of the emerging applications that require extremely low latencies, network operators are facing increasing need to perform latency measurement on per-flow basis for network monitoring and troubleshooting. In this paper, we propose COLATE, the first per-flow latency measurement scheme that requires no probe packets and time stamping. Given a set of observation points, COLATE records packet timing information at each point so that later for any two points, it can accurately estimate the average and standard deviation of the latencies experienced by the packets of any flow in passing the two points. The key idea is that when recording packet timing information, COLATE purposely allows noise to be introduced for minimizing storage space, and when querying the latency of a target flow, COLATE uses statistical techniques to denoise and obtain an accurate latency estimate. COLATE is designed to be efficiently implementable on network middleboxes. In terms of processing overhead, COLATE performs only one hash and one memory update per packet. In terms of storage space, COLATE uses less than 0.1 bit per packet, which means that, on a backbone link with about half a million packets per second, using a 256GB drive, COLATE can accumulate time stamps of packets traversing the link for over 1.5 years. We evaluated COLATE using three real traffic traces that include a backbone traffic trace, an enterprise network traffic trace, and a data center traffic trace. Results show that COLATE always achieves the required reliability for any given confidence interval.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations – *Network management*; C.4 [**Performance of Systems**]: Measurement Techniques

Keywords

Latency Measurement; Passive Measurement; Flows

SIGMETRICS'14, June 16–20, 2014, Austin, TX, USA.

http://dx.doi.org/10.1145/2591971.2591988.

1. INTRODUCTION

1.1 Motivation

Although traditionally throughput has been the primary focus of network engineers, nowadays latency has seen growing importance because a wide variety of emerging applications and architectures require extremely low (in microseconds) and stable (low jitter) latencies. First, many emerging applications, such as financial trading applications [28], storage applications utilizing Fiber Channel over Ethernet [5], and high performance computing applications in data center networks [13], demand low latency. A small increase in latency may cause violations of service level agreements and result in significant revenue losses. For example, a one-millisecond advantage in financial trading applications can be worth $100 million a year for major brokerage firms [28]. Second, many emerging architectures, such as content delivery networks (CDNs) and mission-critical data center networks, demand low latency. CDN providers are mostly evaluated and ranked by content publishers based on latency. Companies such as Cedexis [3] and Turbobytes [10] constantly evaluate and rank CDN providers mostly based on latency. A one-millisecond disadvantage could put one CDN provider behind others and result in loss of business with content publishers. Similarly, the transit providers are primarily evaluated and ranked by CDN providers based on latency. For data centers running mission-critical applications, latency guarantee is a key requirement for the underlying networks. Low latency data center networks have become the primary focus of many data center network solution providers such as Sidera [7].

In managing networks with stringent latency demands, operators often need to measure the latency between two observation points for a particular flow. An *observation point* is either a port in a middlebox (such as a router or a switch) or a network card in an end host. Per-flow latency measurement can be used reactively by network operators to perform tasks such as detecting and localizing delay spikes in a network, isolating offending flows that are responsible for causing delay bursts, and rerouting them through other paths. It can also be used proactively by network operators to continuously monitor latencies between observation points for locating bottleneck links and replace them with higher capacity links.

Existing routers and switches provide little help for latency measurement and monitoring. SNMP counters measure the number of packets passing through a port. NetFlow measures basic statistics, such as the numbers of packets and

bytes, of a flow. Both provide no measurement on latency. Network operators often rely on injecting probe packets to measure end-to-end delays and then use tomographic techniques to infer link and hop properties [16, 17]. However, to achieve latency measurement with extremely high accuracy, the required number of probe packets will be extremely large; consequently, the probe packets will consume too much bandwidth and the measured latency does not reflect the real latency without probe packets. Although some specialized latency monitoring devices are commercially available, they are too costly to be widely deployed. For example, London, Singapore, and Tokyo stock exchanges use latency monitoring devices manufactured by Corvil [8, 9, 11] costing around USD 180,000 for a 2×10Gbps box [4].

1.2 Problem Statement

This paper addresses the fundamental problem of *per-flow latency measurement*: for any flow that passed through any two observation points, measure (or say estimate) the average and standard deviation of the latencies experienced by the packets of that flow in passing through the two observation points. Formally, *given a confidence interval* $\beta \in (0, 1]$ *and a required reliability* $\alpha \in [0, 1)$*, for any flow* f *that passed through any two observation points* S *and* R*, obtain estimates* $\tilde{\mu}_f$ *of the average* μ_f *and* $\tilde{\sigma}_f$ *of the standard deviation* σ_f *of the latencies experienced by the packets of* f *in passing through* S *and* R *so that* $P\{|\tilde{\mu}_f - \mu_f| \leq \beta\mu_f\} \geq \alpha$ *and* $P\{|\tilde{\sigma}_f - \sigma_f| \leq \beta\sigma_f\} \geq \alpha$.

An accurate per-flow latency measurement scheme should further satisfy the following two requirements. (1) *No packet probing*: As probe packets may use up a significant portion of network bandwidth, the latency measured with the insertion of probe packets may significantly deviate from the real latency. Thus, the estimates obtained with probe packets may not suffice for microsecond level accuracy. (2) *No time stamping*: First, IP headers do not have a time stamp field and the TCP time stamp option is meant for measuring end-to-end latencies. Embedding time stamps at observation points requires modifications to packet header formats, which further requires modifications to the data forwarding paths of existing routers and middleboxes. Furthermore, the added packet header fields may consume a significant portion of network bandwidth. Second, the process of attaching time stamps to each packet takes a non-negligible amount of time at observation points. Thus, the latency measured with time stamping may significantly deviate from the real latency.

1.3 Limitations of Prior Art

To the best of our knowledge, there are only two per-flow latency measurement schemes, namely RLI [23] and MAPLE [24]. However, neither of them satisfies both requirements because RLI uses packet probing and MAPLE attaches time stamps to every packet. Other than RLI and MAPLE, the closest work is LDA [21], which performs *aggregate latency measurement*, *i.e.*, *given any two observation points, measure (or say estimate) the average and standard deviation of the latencies experienced by all the packets that passed through the two observation points, regardless of the flow that each packet belongs to.* Aggregate latency measurement is useful; however, it does not provide fine grained per-flow latency information. Here is an important fact: for all flows that passed through two arbitrary observation points S and R, the latencies experienced by the packets of different flows in passing through S and R can be quite different. First, there may be multiple paths from S to R and different flows may be routed via different paths. Second, at each intermediate middlebox along a path from S to R, packets of different flows may take different amount of processing time due to mechanisms such as QoS. As aggregate latency measurement does not reflect the latency of every flow, it falls short in engineering latency sensitive networks. On one hand, when the aggregate latency between two observation points appears normal, the latency experienced by an individual flow may be wildly abnormal. On the other hand, when the aggregate latency between two observation points appears abnormal, aggregate latency measurement does not provide operators the per-flow latency information needed to identify the flows being hurt.

1.4 Proposed Approach

In this paper, we propose COLATE, a Counter based Per-flow Latency Estimation scheme. The key idea of COLATE is that it records timing information of packets at each observation point and purposely allows noise to be introduced in the recorded timing information for minimizing storage space. When querying the latency of a target flow, COLATE statistically denoises the recorded information to obtain an accurate latency estimate. COLATE has two phases: recording phase and querying phase. Next, we give an overview of these two phases.

Recording Phase: In this phase, at each observation point, COLATE records the timing information of each packet arriving at or departing from that point using a vector of counters in RAM, which we call *counter vector*. For each flow with a unique ID, COLATE maps it to a unique subset of these counters, which we call *counter subvector*. The ID of a flow can be any flow identifier such as the standard five tuple (*i.e.*, source IP, destination IP, source port, destination port, and protocol type). To make the mapping unique and memoryless (*i.e.*, using no memory to keep track of the mapping), COLATE maps each flow to a random subvector such that the probability of two different flows being mapped to the same subvector is practically zero. A counter may belong to multiple counter subvectors. Figure 1 shows an example counter vector and its three counter subvectors, from which we see that counters 5 and 8 belong to multiple counter subvectors. For each arriving or departing packet at the observation point, COLATE executes two simple steps: (1) randomly maps the packet to a counter in the counter subvector of the flow that the packet belongs to; (2) adds the current time to that counter. Before any counter overflows, COLATE dumps the counter vector to a permanent storage (such as a solid state drive (SSD)) and resets counters to zero. We call a dumped counter vector a *counter epoch*, which has two attributes: the time stamp of the first recorded packet and the time stamp of the last recorded packet. The recording module can be implemented in hardware to keep up with wire speed. For the hashing function, we can use hardware hash implementations such as those proposed in [20, 31]. For each counter, we can store its less significant bits as a counter in SRAM and the more significant bits as a counter in DRAM – when the counter in SRAM overflows, we increment the corresponding counter in DRAM.

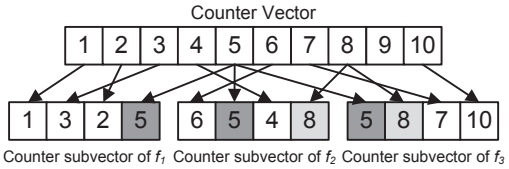


Figure 1: Counter vector and counter subvectors

Querying Phase: In this phase, given a latency measurement query of a flow f, which contains the flow ID, the starting and ending time of the flow, and two observation points that the flow passed through within the time frame, COLATE first finds all the counter epochs whose time frame overlaps with the starting and ending time of the flow f at each of the two given observation points. Second, for each counter epoch, COLATE applies statistical techniques to estimate the sum of time stamps contributed only by flow f from each counter in the counter subvector of f. COLATE uses these extracted values to estimate the average and standard deviation of the latencies experienced by the packets of flow f in passing through the two observation points. COLATE requires the clocks of different observation points to be accurately synchronized; otherwise, the measured latencies will contain a constant offset. This synchronization can be simply achieved by the standard time synchronization protocol IEEE 1588, which provides microsecond level time synchronization [6].

Key Intuition: The intuition behind mapping a flow to multiple counters (in a counter subvector), instead of a single counter, is to mitigate counter overflow for elephant flows. The motivation behind sharing counters among multiple flows, instead of allocating unique counters for each individual flow, is to save memory. Due to the sharing of counters among multiple flows, the counter subvector of a flow f contains not only the timing information of the packets in f but also that of the packets in other flows. The intuition behind allowing this "mixing" is that later in the querying phase, we can extract the timing information of only the packets in the flow f using statistical techniques by treating the mixed-in timing information of the packets from other flows as noise.

Deployability: COLATE is designed to be efficiently implementable on network middleboxes (such as routers and switches) from both processing overhead and storage space perspectives. In terms of processing overhead, COLATE performs only one hash and one memory update per packet. On traditional memory architecture, one memory update requires two memory accesses (*i.e.*, one read and one write); however, in modern memory architecture used in high speed routers (such as the smart memory architecture developed by Huawei [27] and the bandwidth engine developed by MoSys [29]), where each memory location has built-in circuitry for handling updates on site, one memory update (such as incrementing by up to a 64-bit number) requires only one memory access. In terms of storage space, COLATE uses less than 0.1 bit per packet. To get an idea of the length of time for which COLATE can accumulate time stamps using commodity permanent storage devices, consider the 10GigE backbone link in San Jose monitored by CAIDA. At the time of writing this paper, an interactive tool at CAIDA's website reported that on average approximately 0.484 million packets traversed this link per second between Nov. 01 and Nov. 29, 2013 [1]. With 0.1 bits per packet, on a commodity 256GB SSD, COLATE can accumulate time stamps of packets traversing this link for about 1.5 years, which means that a network operator can measure average and standard deviation of up to 1.5 years old flows. This gives not only enough time to identify and debug any problems, but also enough information to study other aspects related to packet delays such as diurnal patterns in flow latencies. On a 12TB SSD, as recently showcased by OCZ at CES-2012 [12], COLATE can accumulate time stamps of packets traversing this link for more than 69 years.

Packet Losses: COLATE, as described above, assumes no packet losses. However, to handle packet losses in COLATE, we can easily adapt the strategy proposed in [21] for handling packet losses in the aggregate latency measurement scheme LDA. According to this strategy, instead of maintaining a single counter vector, COLATE can maintain a set of counter vectors at each observation point, where each counter vector has a sampling probability and a packet counter associated with it. The sum of the sampling probabilities of all counter vectors is 1. The sampling probability of a counter vector is the fraction of packets whose time stamps COLATE will add to this counter vector. The packet counter of a counter vector keeps track of the number of packets whose time stamps have been added to this counter vector. In the recording phase, when a packet arrives at or departs from an observation point, COLATE first uses a hash function to map the packet to a counter vector such that in the long run, the fraction of packets that it maps to any counter vector is equal to its sampling probability. Then, COLATE adds the time stamp of the packet to this vector as described earlier. Note that all observation points use the same hash function to guarantee that the same packet is mapped to the same counter vector at all observation points. In the querying phase, for a target flow between two observation points, COLATE compares the packet counter of each counter vector at one observation point with that of the corresponding counter vector at the other observation point. Then, COLATE selects those two counter vectors at the two observation points that have the highest sampling probability associated with them and equal values of packet counters. After this, COLATE follows the procedure of the querying phase described earlier. For simplicity, in the rest of this paper in describing COLATE, we assume no packet losses.

1.5 Technical Challenges and Solutions

The first challenge is to denoise the recorded information to extract the sum of the time stamps of the packets in the target flow from the counter subvector of the flow. To address this challenge, we first show that for each counter in the counter subvector of the target flow, the value contributed by the time stamps from the packets in the target flow and that from the packets in other flows can both be modeled with binomial distributions. We then derive an expression to calculate the expected value of each counter in the counter subvector of the target flow. From this expression, we estimate the sum of the time stamps of all the packets of the target flow. Using this estimate in conjunction with maximum likelihood estimation, we extract the sum of the time stamps of the packets in the target flow from each counter in the counter subvector.

The second challenge is to calculate the sum of the squares of the latencies of each packet in a target flow, which is needed for calculating the standard deviation of packet latencies. To address this challenge, we first use the time stamp sum extracted from counter subvectors to construct a virtual deviation vector and then use this vector to estimate this sum of the squares of the latencies.

1.6 Advantages of COLATE over Prior Art

COLATE brings forward the state-of-the-art in per-flow latency measurement on the following fronts: reliability, passiveness, scalability, memory, efficiency, and flexibility. For reliability, COLATE takes the required reliability and confidence interval specified by network operators as input whereas existing schemes do not. For passiveness, COLATE neither sends probe packets nor attaches time stamps to packets. For scalability, COLATE maintains only one counter vector at each observation point regardless of how many other observation points are sending and receiving packets from it; in contrast, LDA and RLI have to maintain separate vectors of counters for each pair of sender and receiver. For memory, COLATE uses less than 0.1 bit of storage space per packet, which is over 128 *times* improvement compared to 12.8 bits of storage space per packet used by MAPLE. Due to this, on a commodity 256GB SSD, where COLATE can accumulate time stamps of packets traversing the San Jose backbone link for about one and a half year, MAPLE can accumulate the time stamps for only 4.1 days. For efficiency, COLATE performs only 1 hash and 1 memory update per packet whereas MAPLE uses 9 hashes and 12 memory accesses per packet. For flexibility, COLATE allows different observation points to allocate different amount of RAM based on their available resources whereas LDA requires both the sender and receiver to allocate the same amount of RAM.

2. RELATED WORK

To the best of our knowledge, there are only two per-flow latency measurement schemes, namely MAPLE [24] and RLI [23]. In MAPLE, for any packet passing through two observation points S and R, S attaches a time stamp to the packet and R calculates the latency of the packet from S to R by subtracting the time stamp from its current time. To reduce space for storing latency values of all packets, MAPLE maps the calculated latency of each packet to the closest value in a set of predetermined latency values. Thus, instead of storing the latency of every packet, MAPLE only stores these predetermined latency values and for each predetermined latency value MAPLE uses a Bloom filter to store all packets mapped to it. To query the latency of a given packet, MAPLE first finds the predetermined latency value that the packet was mapped to by querying Bloom filters and uses that value as the estimated latency of the packet. Compared with COLATE, MAPLE falls short from a few perspectives. First, MAPLE is based on a strong assumption that packet latencies between two observation points are tightly clustered around a set of predetermined latency values. We have not found any theoretical or empirical validation of this assumption in prior literature. Second, MAPLE requires attaching time stamps to packets and thus has the limitations we pointed out in Section 1.2 for time stamping based latency measurement schemes. Furthermore, time stamping individual packets can consume up to 10% of the available bandwidth [21]. Third, MAPLE has large memory overhead (12.8 bits/packet at each observation point) and large processing overhead (9 hash computations and 12 memory accesses per packet: 9 for hash functions, 2 for updating counters, and 1 for determining the cluster a packet's latency belongs to), whereas COLATE uses 0.1 bits/packet and performs 1 hash and 1 memory update per packet.

In RLI, for a flow passing through two observation points S and R, S inserts probe packets with time stamps into the flow and R calculates the latency of each probe packet similarly to MAPLE. To calculate the latency of the regular packets between two probe packets whose latency has been calculated as l_1 and l_2, R simply uses the straight line equation to calculate the latency of these regular packets based on their arrival time and the latency of the two probe packets. Compared with COLATE, RLI has the following limitations. First, RLI is based on the strong assumption that packet latency between S and R increases or decreases linearly in the time interval between receiving any two probe packets at R. When the time interval between two probe packets is extremely small, this assumption may practically hold but the extremely small time interval implies that the number of probe packets is extremely large. For example, to achieve an accuracy of only 81%, RLI inserts, on average, 1 probe packet every 4.78 regular packets. Furthermore, as we mentioned in Section 1.2, latency measured with a large number of probe packets may significantly deviate from the real latency when there are no probe packets. When the time interval between two probe packets is large, this assumption does not make intuitive sense and may not hold in practice. Similarly, we have not found any theoretical or empirical validation of this assumption in prior literature.

The other latency measurement schemes are LDA [21] and FineComb [25], which provide aggregate, not per-flow, latency measurement between a sender and a receiver. In LDA, both the sender and the receiver maintain several counter vectors where each element is a pair of counters: time stamp counter for accumulating packet time stamps and packet counter for counting the number of arriving/departing packets. Each vector has a sampling probability and a sampling function. For each arriving or departing packet, LDA first maps the packet to a counter vector such that in the long run, the fraction of packets that LDA maps to any counter vector is equal to its sampling probability. It then randomly maps the packet to a counter pair in this counter vector and adds the time stamp of the packet to the time stamp counter and increments the packet counter by one. To obtain the aggregate latency estimate between the sender and receiver, for each counter pair in each vector, LDA checks whether they have the same packet counter value and selects all counter pairs that have the same packet counter value for both the sender and receiver. Then, LDA can easily calculate the total number of successfully delivered packets and the sum of their time stamps at both sides. Finally, to obtain the aggregate average latency between the sender and receiver, it subtracts the sum of time stamps at the sender side from that at the receiver side and divides it with the total number of successfully delivered packets.

3. COLATE – RECORDING PHASE

In this section, we present the recording phase and the statistical modeling of COLATE. Table 1 summarizes the symbols used throughout this paper.

Table 1: Symbols used in the paper

Symbol	Description
X, S, R	observation points
α	overall required reliability
β	overall required confidence interval
A	individual required reliability
B	individual required confidence interval
$f, \overline{f}$	different flows
μ_f	average latency of packets in flow f
$\tilde{\mu}_f$	estimated value of μ_f
$g\{.\}$	expected value function of C
σ_f	standard deviation of latency of packets in f
$\tilde{\sigma}_f$	estimated value of σ_f
$h\{.\}$	standard deviation function of C
$\mathbf{C}_X$	vector of all counters at X
$\mathbf{S}_X^f$	counter subvector of flow f at X
n	# of counters in vector $\mathbf{C}_X$
m	# of counters in $\mathbf{S}_X^f$
b	size of each counter in bits
H	hash function with output following $\sim U(1, n)$
C_f	random variable for sum of time stamps of f in $\mathbf{S}_X^f[j]$
C_r	random variable for sum of time stamps contributed by all flows, other than f, in $\mathbf{S}_X^f[j]$
C	random variable for value of counter $\mathbf{S}_X^f[j]$
$\tilde{C}$	observed average value of C from m counters
z	an arbitrary packet of flow f
$u_{f,X}[z]$	actual time stamp of packet z of f at X
$w_{f,X}[j]$	sum of time stamps contributed by f to $\mathbf{S}_X^f[j]$
$\tilde{w}_{f,X}[j]$	maximum likelihood estimate of $w_{f,X}[j]$
$v_{f,X}[l]$	a value of the deviation vector
$t_{f,X}$	sum of all time stamps of f at X
$\tilde{t}_{f,X}$	estimated value of $t_{f,X}$
$t_{f,X}^{max}$	maximum value of $t_{f,X}$ at X
$t_{f,X}^{min}$	minimum value of $t_{f,X}$ at X
t_X	sum of all time stamps in a counter epoch at X
T	maximum value that t_X can take on
$\mathcal{T}_f$	random variable for time stamp values of flow f
Q	random array of all m values in the range $[1, m]$
$\mathcal{Y}_{lX}$	set of all time stamps contributed by f to $\mathbf{S}_X^f[Q[2l]]$ and $\mathbf{S}_X^f[Q[2l-1]]$
p_f	# of packets in flow f
$P_{f,j}$	random variable for # of packets of f that contributed time stamps to $\mathbf{S}_X^f[j]$
D_f	random variable for latency of any packet in f
G_z	random variable with values $+1$ and -1
g_z	value that G_z takes on
ξ	$\mathbf{S}_X^f[j] - w_{f,X}[j]$
τ	$t_X - t_{f,X}$
γ	# of virtual repetitions
Z	standard normal random variable
M	amount of RAM allocated to COLATE

3.1 Noisy Accumulation of Time Stamps

At each observation point X, COLATE maintains a vector $\mathbf{C}_X$ of n counters where each counter $\mathbf{C}_X[i]$ $(1 \le i \le n)$ has b bits with initial value 0. When a packet arrives at or departs from an observation point X, COLATE extracts its flow ID f, chooses a random number j from a uniform distribution in the range $[1, m]$ where $m << n$, calculates the hash function $H(f, j)$ whose output is uniformly distributed in range $[1, n]$, and adds the time stamp of this packet (*i.e.*, the current time at observation point X) to the counter $\mathbf{C}_X[H(f, j)]$. Thus, the time stamp of all packets in flow f will be uniformly distributed to m counters: $\mathbf{C}_X[H(f,1)], \mathbf{C}_X[H(f,2)], \cdots, \mathbf{C}_X[H(f,m)]$. These m counters constitute the counter subvector of flow f, which is denoted by $\mathbf{S}_X^f$ where $\mathbf{S}_X^f[j] = \mathbf{C}_X[H(f, j)]$ for $j \in [1, m]$. In this recording phase, for each packet, COLATE performs one memory update to update the value of $\mathbf{C}_X[H(f, j)]$. For different flows, the probability that COLATE maps them to the same counter subvector is $\frac{m!}{n^m}$ $(=2.4 \times 10^{-62}$ for $n = 10000$ and $m = 20$, for example), which is practically zero. Note that an observation point is a port of a middlebox, not a middlebox itself, because the arriving and departing time of a packet at a middlebox are different due to the non-negligible packet processing time within the middlebox.

3.2 Analysis of Noisy Accumulation

First, by Lemma 1, we show that in any counter epoch, on average, a flow contributes the same amount to each counter in its counter subvector. We further show that the amount contributed by a flow to each counter in its counter subvector can be modeled by a binomial distribution. Second, we derive expressions for the expected value and variance of a counter in counter vector in Theorem 1. In Section 4, we use the expression of expected value to estimate the average and standard deviation of packet latencies for any given flow. In Section 5, we use the expression of variance to determine the parameter values that can ensure that the actual reliability achieved by COLATE is no less than the required reliability.

LEMMA 1. *Let C_f be the random variable representing the sum of time stamps contributed by flow f to a counter $\mathbf{S}_X^f[j]$, $1 \le j \le m$, in its counter subvector of length m at observation point X. Let p_f be the number of packets in flow f that contributed time stamps to the current counter epoch $\mathbf{C}_X$. Let $\mathcal{T}_f$ be an independent random variable representing the time stamp contributed by each packet of flow f to $\mathbf{C}_X$. Then, we have $E[C_f] = \frac{p_f \times E[\mathcal{T}_f]}{m}$.*

PROOF. Let $P_{f,j}$ be a random variable representing the number of packets in flow f that contributed time stamps to counter $\mathbf{S}_X^f[j]$. Therefore, $E[C_f] = E\big[\sum_{P_{f,j}} \mathcal{T}_f\big]$. Applying Wald's Lemma, we get $E[C_f] = E[P_{f,j}] \times E[\mathcal{T}_f]$. As the output of hash function H is uniformly distributed in range $[1, n]$, its output is also uniformly distributed in $[1, m]$ for the packets in flow f. Thus, the probability that COLATE adds the time stamp of a packet of the flow f to counter $\mathbf{S}_X^f[j]$ is $1/m$. The random variable $P_{f,j}$ follows the binomial distribution *i.e.*, $P_{f,j} \sim \text{Binom}(p_f, 1/m)$. Therefore, $E[C_f] = \frac{p_f}{m} \times E[\mathcal{T}_f]$. □

Figure 2 plots the CDF of the ratio of the observed values of C_f from simulations of our ICSI traffic trace to the $E[C_f]$ from Lemma 1. We observe from this figure that there is a steep rise in the CDF when the value of this ratio is around 1. We also observe from the simulations that the mean and median of the ratio are both equal to 1. This empirically establishes the result in Lemma 1.

Lemma 1 shows that the sum of time stamps of all packets of flow f is equally divided among all counters in its counter subvector, which conforms to binomial distribution. Thus, we approximate the distribution of C_f with a binomial distribution as $C_f \sim \text{Binom}(t_{f,X}, 1/m)$, where $t_{f,X}$ represents sum of all times-stamps contributed by flow f to the counters in its counter subvector at observation point X. Let C_r be the random variable representing the sum of time stamps

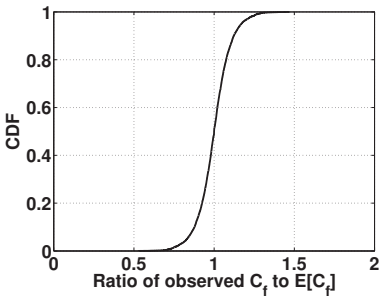


Figure 2: CDF of *observed* $C_f/E[C_f]$

contributed by packets of all flows other than f to counter $\mathbf{S}_X^f[j]$. Using similar reasoning as for C_f, we approximate the distribution of C_r with a binomial distribution. The probability that a packet of a flow $\bar{f} \neq f$ contributes a time stamp to counter $\mathbf{S}_X^f[j]$ is the product of the probability that H maps the packet to $\mathbf{S}_X^f[j]$ given that $\mathbf{S}_X^f[j] \in \mathbf{S}_X^{\bar{f}}$, which is $1/m$, and the probability that counter $\mathbf{S}_X^{\bar{f}}[j]$ is in the counter subvector of $\bar{f}$, which is denoted by $P\{\mathbf{S}_X^f[j] \in \mathbf{S}_X^{\bar{f}}\}$ and calculated as

$$\begin{aligned} P\{\mathbf{S}_X^f[j] \in \mathbf{S}_X^{\bar{f}}\} &= 1 - \left\{ \binom{m}{0} \left(\frac{1}{n}\right)^0 \left(1 - \frac{1}{n}\right)^m \right\} \\ &= 1 - \left\{ 1 - \frac{m}{n} + \frac{(m)(m-1)}{n^2 \times 2!} - \cdots \right\} \\ &\approx \frac{m}{n} \qquad \because m << n \end{aligned}$$

Thus, the probability that a packet of a flow $\bar{f} \neq f$ contributes a time stamp to counter $\mathbf{S}_X^f[j]$ is $\frac{1}{m} \times \frac{m}{n} = \frac{1}{n}$. Thus, $C_r \sim \text{Binom}(t_X - t_{f,X}, 1/n)$.

THEOREM 1. *Let C be the random variable representing the value of a counter $\boldsymbol{S}_X^f[j]$, $1 \leq j \leq m$, in the counter subvector of a flow f. Let $t_{f,X}$ be the sum of all time stamps contributed by packets of flow f to all counters in its counter subvector and t_X be the sum of all time stamps contributed by packets of all flows in the counter epoch at observation point X. Let $C_f \sim Binom(t_{f,X}, 1/m)$ represent the sum of time stamps contributed by packets of flow f to $\mathbf{S}_X^f[j]$ and let $C_r \sim Binom(t_X - t_{f,X}, 1/n)$ represent the sum of time stamps contributed by packets of all flows other than f to counter $\boldsymbol{S}_X^f[j]$. Let C_f and C_r be independent of each other. The expected value and variance of C are calculated as follows*

$$E[C] = \frac{t_{f,X}}{m} + \frac{t_X - t_{f,X}}{n} \tag{1}$$

$$\text{Var}(C) = \left(\frac{t_{f,X}}{m}\right)\left(1 - \frac{1}{m}\right) + \left(\frac{t_X - t_{f,X}}{n}\right)\left(1 - \frac{1}{n}\right) \tag{2}$$

PROOF. As $C = C_f + C_r$, we get, $E[C] = E[C_f] + E[C_r] = \frac{t_{f,X}}{m} + \frac{t_X - t_{f,X}}{n}$. As C_f and C_r are assumed to be independent, $\text{Var}(C) = \text{Var}(C_f) + \text{Var}(C_r)$. As variance of $Binom(v, w)$ is $vw(1-w)$, we get $\text{Var}(C_f) = \left(\frac{t_{f,X}}{m}\right)\left(1 - \frac{1}{m}\right)$ and $\text{Var}(C_r) = \left(\frac{t_X - t_{f,X}}{n}\right)\left(1 - \frac{1}{n}\right)$. □

In Theorem 1, we assumed C_f and C_r to be independent of each other, which is true whenever $t_{f,X} << t_X$. In practice $t_{f,X}$ indeed is much smaller than t_X because $t_{f,X}$ is the sum of the time stamps added by a single flow in the counter epoch while t_X is the sum of the time stamps added by all flows (in the order of tens and hundreds of thousands). Furthermore, in Theorem 1, we approximate the distributions of C_f and C_r with binomial distributions because when there are a large number of packets that contribute time stamps to the counters in the counter vector, we can approximate each time stamp to be smeared over the counters. This approximation makes the formal development of variance of C and its subsequent use in calculating parameters for COLATE tractable. If the exact equation for variance of C is desired, it can be obtained as follows. Consider any packet with time stamp t_p not belonging to a flow f. This packet has a probability $\frac{1}{n}$ of being mapped to a counter in the counter subvector of the flow f. Thus, the time stamp for each such packet will contribute a variance of $t_p^2(\frac{1}{n})(1 - \frac{1}{n})$ to the overall variance of C. In Theorem 1, $t_{f,X}/m$ models the average sum of the time stamps contributed by flow f, and $(t_X - t_{f,X})/m$ models the average *noise* contributed by all other flows to counter $\mathbf{S}_X^f[j]$.

4. COLATE – QUERYING PHASE

In this section, we present the methods that COLATE uses to estimate the average and standard deviation of the latencies of the packets of a flow in passing any two observation points.

4.1 Estimating Latency Average

For a flow f passing through observation point S and then observation point R, we want to calculate $\tilde{\mu}_f$, the estimate of average latency μ_f of flow f. For a packet z in flow f, let $u_{f,S}[z]$ be its time stamp at observation point S and $u_{f,R}[z]$ be its time stamp at observation point R. The delay experienced by this packet in traveling from S to R is thus $u_{f,R}[z] - u_{f,S}[z]$. Let $t_{f,S}$ and $t_{f,R}$ be the sums of all time stamps of the packets in flow f at S and R, respectively. Recall that p_f denotes the number of packets in f. Thus,

$$\begin{aligned} \mu_f &= \frac{1}{p_f} \sum_{\forall z} \big(u_{f,R}[z] - u_{f,S}[z]\big) \\ &= \frac{1}{p_f}\Big(\sum_{\forall z} u_{f,R}[z] - \sum_{\forall z} u_{f,S}[z]\Big) = \frac{1}{p_f}(t_{f,R} - t_{f,S}) \end{aligned}$$

Note that the value of p_f can be measured by tools such as NetFlow available on Cisco routers or by schemes proposed in [22, 26]. Thus, to estimate the value of μ_f, we need to obtain the estimated values $\tilde{t}_{f,S}$ and $\tilde{t}_{f,R}$ for $t_{f,S}$ and $t_{f,R}$, respectively. Then, we can calculate

$$\tilde{\mu_f} = \frac{1}{p_f}(\tilde{t}_{f,R} - \tilde{t}_{f,S}) \tag{3}$$

Theorem 2 shows how to obtain $\tilde{t}_{f,X}$ at X.

THEOREM 2. *Given a counter epoch $\mathbf{C}_X$ of length n at observation point X where each counter subvector is of length m, let t_X denote the sum of all counters in $\mathbf{C}_X$, the estimate $\tilde{t}_{f,X}$ of the sum of all time stamps of the packets in flow f is calculated as follows*

$$\tilde{t}_{f,X} = \frac{1}{n-m}\left\{ n \sum_{j=1}^{m} \mathbf{S}_X^f[j] - m t_X \right\} \tag{4}$$

PROOF. Given $\mathbf{C}_X$ and flow f, we can easily obtain the values of every counter in the counter subvector $\mathbf{S}_X^f$ of f. Thus, we can calculate $E[C]$ as $E[C] = \frac{1}{m}\sum_{j=1}^{m} \mathbf{S}_X^f[j]$. Substituting $E[C]$ in Equation (1) by $\frac{1}{m}\sum_{j=1}^{m} \mathbf{S}_X^f[j]$, replacing $t_{f,X}$ by $\tilde{t}_{f,X}$, and solving for $\tilde{t}_{f,X}$, gives Equation (4). □

4.2 Estimating Latency Standard Deviation

Let D_f be the random variable representing the latency experienced by a packet in flow f. The standard deviation of D_f can be calculated by $\tilde{\sigma}_f = \sqrt{\mathrm{Var}(D_f)}$, where $\mathrm{Var}(D_f)$ can be calculated as follows:

$$\begin{aligned}\mathrm{Var}(D_f) &= E[D_f^2] - E^2[D_f] \\ &= \left\{\frac{1}{p_f}\sum_{\forall z}\left(u_{f,R}[z]-u_{f,S}[z]\right)^2\right\} - \left\{\frac{1}{p_f}\sum_{\forall z}\left(u_{f,R}[z]-u_{f,S}[z]\right)\right\}^2 \\ &= \left\{\frac{1}{p_f}\sum_{\forall z}\left(u_{f,R}[z]-u_{f,S}[z]\right)^2\right\} - \left\{\frac{1}{p_f}\left(t_{f,R}-t_{f,S}\right)\right\}^2 \qquad (5)\end{aligned}$$

We can calculate the second term $\{\frac{1}{p_f}(t_{f,R} - t_{f,S})\}^2$ in Equation (5) based on Theorem 2. Now the key challenge is to calculate the first term $\{\frac{1}{p_f}\sum_{\forall z}(u_{f,R}[z] - u_{f,S}[z])^2\}$ in Equation (5). Our solution to this challenge is based on the statistical technique proposed by Alon *et al.* in [14]. The main idea is to introduce a random variable G_z where the value g_z that this random variable takes on is either $+1$ or -1 with equal probability. Before adding the time stamp of a packet z to a counter, if we randomly multiply the time stamp with g_z and then add it to the counter, then we will get Equation (6).

$$E\left[\left\{\sum_{\forall z}\left(g_z u_{f,R}[z] - g_z u_{f,S}[z]\right)\right\}^2\right] = \sum_{\forall z}\left(u_{f,R}[z]-u_{f,S}[z]\right)^2 \quad (6)$$

This can be proven as follows:

$$\begin{aligned}E\left[\left\{\sum_{\forall z} g_z\left(u_{f,R}[z] - u_{f,S}[z]\right)\right\}^2\right] &= E\left[\sum_{\forall z} g_z^2\left(u_{f,R}[z] - u_{f,S}[z]\right)^2\right. \\ &\left. + \sum_{\forall z\neq\overline{z}} g_z g_{\overline{z}}\left(u_{f,R}[z] - u_{f,S}[z]\right)\left(u_{f,R}[\overline{z}] - u_{f,S}[\overline{z}]\right)\right]\end{aligned}$$

Using the well known result that expectation of sum of random variables is the sum of their individual expectations and that $g_z^2 = 1$, we get:

$$\begin{aligned}&= \sum_{\forall z}\left(u_{f,R}[z] - u_{f,S}[z]\right)^2 \\ &+ \sum_{\forall z\neq\overline{z}}\left(u_{f,R}[z] - u_{f,S}[z]\right)\left(u_{f,R}[z] - u_{f,S}[z]\right) \times E[G_z G_{\overline{z}}]\end{aligned}$$

Note that $\{G_1, G_2, G_3, \dots\}$ is a set of independent and identically distributed random variables. So, $E[G_z G_{\overline{z}}] = E[G_z] \times E[G_{\overline{z}}]$. As $E[G_z] = 0$ for all values of z, this implies $E[G_z G_{\overline{z}}] = 0$. Thus, the second term in the equation above is 0, which proves Equation (6).

We now present our method for calculating the first term in Equation (5). First, for each counter in the counter subvector of f, whose value is contributed by the time stamps of the packets in flow f and those of the packets in other flows, we use Theorem 3 to extract an estimate of the value that is contributed by only the time stamps of the packets in f. In other words, we eliminate the noise introduced by the packets of flows other than f from the counter subvector of f. Second, we statistically simulate the process of multiplying the time stamp of a packet in f with the random variable G_z and then adding the multiplication result to the corresponding counter in the counter subvector of f. By repeating this process a statistically sufficient number of times, we obtain an accurate estimate of $\sum_{\forall z}(u_{f,R}[z] - u_{f,S}[z])^2$ based on Theorem 4. Note that this simulation does not require any changes to the recording phase. Next, we present our denoising solution and statistical simulation process.

4.2.1 Denoising Counter Subvectors

Our denoising solution is based on Theorem 3. The numerical solution of Equation (7) gives us the estimate of the value that is contributed by only the time stamps of the packets in f for each counter in f's subvector.

THEOREM 3. *Let $w_{f,X}[j]$ $(1 \le j \le m)$ be the sum of the time stamps of flow f's packets that are mapped to counter $\mathbf{S}_X^f[j]$ at observation point X. Let $\tilde{t}_{f,X}$ be the estimate of sum of all time stamps contributed by packets of flow f to all counters in f's counter subvector and t_X be the sum of all time stamps contributed by packets of all flows in the counter epoch at observation point X. The maximum likelihood estimate $\tilde{w}_{f,X}[j]$ of $w_{f,X}[j]$ satisfies the following equation:*

$$\begin{aligned}\ln\{n-1\} = \psi^{(0)}\{t_X - \tilde{t}_{f,X} - \mathbf{S}_X^f[j] + \tilde{w}_{f,X}[j] + 1\} \\ - \psi^{(0)}\{\mathbf{S}_X^f[j] - \tilde{w}_{f,X}[j] + 1\} \qquad (7)\end{aligned}$$

where $\psi^{(0)}\{.\}$ is the 0^{th} order polygamma function.

PROOF. The maximum likelihood estimate of $w_{f,X}[j]$ is the value of $w_{f,X}[j]$ that maximizes the probability that the counter $\mathbf{S}_X^f[j]$ takes the observed value.

$$\arg\max_{w_{f,X}[j]} P\{C = \mathbf{S}_X^f[j] | w_{f,X}[j]\}$$

This value of $w_{f,X}[j]$ can be obtained by differentiating $P\{C = \mathbf{S}_X^f[j] | w_{f,X}[j]\}$ w.r.t $w_{f,X}[j]$ and equating to 0.

$$\frac{d}{d(w_{f,X}[j])} P\{C = \mathbf{S}_X^f[j] | w_{f,X}[j]\} = 0$$

As $C = C_f + C_r$ and $C_f = w_{f,X}[j]$, the L.H.S. becomes

$$\frac{d}{d(w_{f,X}[j])} P\{C_r = \mathbf{S}_X^f[j] - w_{f,X}[j]\}$$

For simplicity, let $\xi = \mathbf{S}_X^f[j] - w_{f,X}[j]$ and $\tau = t_X - t_{f,X}$. As C_r is a binomial random variable, this derivative further becomes

$$\begin{aligned}&\frac{d}{d(w_{f,X}[j])}\binom{\tau}{\xi}\left(\frac{1}{n}\right)^{\xi}\left(1-\frac{1}{n}\right)^{\tau-\xi} = \left\{\binom{\tau}{\xi}\left(\frac{1}{n}\right)^{\xi}\left(1-\frac{1}{n}\right)^{\tau-\xi}\right\} \\ &\times\left\{\psi^{(0)}\{\xi+1\} - \psi^{(0)}\{\tau-\xi+1\} + \ln\left\{1-\frac{1}{n}\right\} - \ln\left\{\frac{1}{n}\right\}\right\} \quad (8)\end{aligned}$$

Due to space limitations, we have skipped the intermediate derivation steps, which use the following identity:

$$\frac{d}{dw}\binom{v}{w} = \binom{v}{w}\left(\psi^{(0)}\{v-w+1\} - \psi^{(0)}\{w+1\}\right)$$

By replacing $t_{f,X}$ with $\tilde{t}_{f,X}$, which is calculated using Theorem 2, in $\tau = t_X - t_{f,X}$ and further in the R.H.S of Equation (8) and equating it to zero, we obtain the maximum likelihood estimate $\tilde{w}_{f,X}[j]$ of $w_{f,X}[j]$. As $\binom{\tau}{\xi}\left(\frac{1}{n}\right)^{\xi}\left(1-\frac{1}{n}\right)^{\tau-\xi}$ is $P\{C_f = \mathbf{S}_X^f[j] | w_{f,X}[j]\}$, which is not equal to zero, we have

$$\left\{\psi^{(0)}\{\xi+1\} - \psi^{(0)}\{\tau-\xi+1\} + \ln\left\{1-\frac{1}{n}\right\} - \ln\left\{\frac{1}{n}\right\}\right\} = 0$$

Simplifying the $\ln\{.\}$ terms results in Equation (7). □

4.2.2 Statistical Simulations

We have obtained the m values extracted using Theorem 3 from f's counter subvector at observation point X. For flow f that passes observation point X, each unique permutation of the m distinct integers from 1 to m, denoted by vector Q, defines a unique *deviation vector* $v_{f,X}$ of size $m/2$ as follows. To ensure that $m/2$ is an integer, we choose m to be an even number.

$$v_{f,X}[l] = \tilde{w}_{f,X}\big[Q[2l]\big] - \tilde{w}_{f,X}\big[Q[2l-1]\big] \quad 1 \le l \le m/2 \qquad (9)$$

Each unique permutation of the m distinct integers from 1 to m is essentially a unique simulation of the aforementioned statistical process of multiplying half the time stamps with $G_z = +1$ and the other half with $G_z = -1$. Theorem 4 gives us the way to estimate $\sum_{\forall z}(u_{f,R}[z] - u_{f,S}[z])^2$, which is needed for calculating $\mathrm{Var}(D_f)$ based on Equation (5).

THEOREM 4. *Given any two observation points S and R, for any permutation Q of the m distinct integers from 1 to m, let $v_{f,S}$ and $v_{f,R}$ be the corresponding deviation vectors of flow f at observation points S and R, respectively. The following equation holds:*

$$\sum_{\forall z}\big(u_{f,R}[z] - u_{f,S}[z]\big)^2 = E\Big[\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2\Big] \qquad (10)$$

PROOF. Let $\mathcal{Y}_{lS}$ be the set of all the time stamps contributed by the packets of flow f to counters $\boldsymbol{S}^f_S[Q[2l]]$ and $\boldsymbol{S}^f_S[Q[2l-1]]$. Similarly, let $\mathcal{Y}_{lR}$ be the set of all the time stamps contributed by the packets of flow f to counters $\boldsymbol{S}^f_R[Q[2l]]$ and $\boldsymbol{S}^f_R[Q[2l-1]]$. Let $y_{lS}[i]$ be the i-th element of $\mathcal{Y}_{lS}$, where $1 \le i \le |\mathcal{Y}_{lS}|$. Similarly, let $y_{lR}[i]$ be the i-th element of $\mathcal{Y}_{lR}$, where $1 \le i \le |\mathcal{Y}_{lR}|$. Starting from the R.H.S of Equation (10), we have:

$$\begin{aligned}
&= \sum_{l=1}^{\frac{m}{2}} E\Big[(v_{f,R}[l] - v_{f,S}[l])^2\Big] \\
&= \sum_{l=1}^{\frac{m}{2}} E\Big[\Big\{\sum_{i=1}^{|\mathcal{Y}_{lR}|} g_i.y_{lR}[i] - \sum_{i=1}^{|\mathcal{Y}_{lS}|} g_i.y_{lS}[i]\Big\}^2\Big] \\
&= \sum_{l=1}^{\frac{m}{2}} E\Big[\Big\{\sum_{i=1}^{|\mathcal{Y}_{lR}|} (g_i.y_{lR}[i] - g_i.y_{lS}[i])\Big\}^2\Big] \because |\mathcal{Y}_{lR}| = |\mathcal{Y}_{lS}| \\
&= \sum_{l=1}^{\frac{m}{2}} \sum_{i=1}^{|\mathcal{Y}_{lR}|} \big(y_{lR}[i] - y_{lS}[i]\big)^2, \text{ using Equation (6)} \\
&= \sum_{\forall z}\big(u_{f,R}[z] - u_{f,S}[z]\big)^2
\end{aligned}$$

The last equality follows from the fact that each $y_{lX}[i]$ is actually the value of some time stamp $u_{f,X}[z]$ at observation point X. □

For any permutation Q of the m distinct integers from 1 to m, we calculate $v_{f,R}[l]$ and $v_{f,S}[l]$ for each $1 \le l \le m/2$ based on Equation (9), and then calculate $\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2$, which is one estimate of $\sum_{\forall z}\big(u_{f,R}[z] - u_{f,S}[z]\big)^2$ according to Theorem 4. As $\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2$ is a random variable, its variance can be reduced by γ times if we repeat the above process γ times using a different random permutation Q each time and use the average of the γ values of $\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2$ as the estimate of $\sum_{\forall z}\big(u_{f,R}[z] - u_{f,S}[z]\big)^2$. As the R.H.S. of Equation (10) is an expected value, to get an accurate estimate of $\sum_{\forall z}\big(u_{f,R}[z] - u_{f,S}[z]\big)^2$, we need to calculate $\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2$ for a statistically sufficient number of unique permutations of the m distinct integers from 1 to m. The process of calculating $\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2$ for different permutations of Q is essentially simulating the aforementioned *random* statistical process of multiplying the time stamp of each packet with random variable G_z that takes the value of $+1$ and -1 with equal probability *without having to perform this process in the recording phase*. We name this process of calculating $\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2$ using different permutations of Q as *virtual repetitions*. The number of distinct ways in which we can repeat this process is $\prod_{i=1}^{\frac{m}{2}}\binom{m-2(i-1)}{2}$, which is large enough for us to obtain any required reliability α for estimating the standard deviation. For example, when $m = 20$, $\prod_{i=1}^{\frac{m}{2}}\binom{m-2(i-1)}{2} = 2.38 \times 10^{15}$.

4.2.3 Steps of Estimating Standard Deviation

To summarize, COLATE performs the following six steps to estimate the standard deviation of the latencies that the packets in flow f experienced in traversing from observation points S to R. (1) Obtain the number of packets in flow f, denoted by p_f, using NetFlow or the schemes proposed in [22, 26]. (2) Obtain the estimates of $t_{f,S}$ and $t_{f,R}$, which are the sum of the time stamps of all packets in flow f at observation points S and R respectively, using Theorem 2. (3) Extract the values of $w_{f,S}[j]$ and $w_{f,R}[j]$, which are the sum of the time stamps of flow f's packets that are mapped to counter $\boldsymbol{S}^f_S[j]$ at observation point S and to counter $\boldsymbol{S}^f_R[j]$ at observation point R, respectively, for all $1 \le j \le m$, using Theorem 3. (4) Randomly choose γ permutations of the m distinct integers from 1 to m. For each permutation Q, first calculate $v_{f,R}[l]$ and $v_{f,S}[l]$ for all $1 \le l \le m/2$ using Equation (9) and then calculate $\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2$. (5) Calculate the average of the γ values of $\sum_{l=1}^{\frac{m}{2}}(v_{f,R}[l] - v_{f,S}[l])^2$, which is the estimated value of $\sum_{\forall z}\big(u_{f,R}[z] - u_{f,S}[z]\big)^2$ (6) Estimate of the standard deviation using Equation (5).

5. COLATE – RELIABILITY

COLATE has four parameters: (1) the total number of counters denoted by n, (2) the number of counters in each counter subvector denoted by m, (3) the number of bits in each counter denoted by b, and (4) the vector threshold denoted by T. Note that when the sum of all n counters in a counter vector reaches T, COLATE dumps the counter vector into permanent storage as a counter epoch and then resets all counter values to be zero. In this section, we present solutions to find the values for these parameters so that our estimated average latency achieves the required reliability $\alpha \in [0, 1)$ for the given confidence interval $\beta \in (0, 1]$. Note that for standard deviation, we have already presented a method in Section 4 that can achieve arbitrarily high required reliability. Recall $\tilde{\mu}_f = \frac{1}{p_f}(\tilde{t}_{f,R} - \tilde{t}_{f,S})$ (in Equation (3)), which shows that the estimate $\tilde{\mu}_f$ depends on two other estimates $\tilde{t}_{f,S}$ and $\tilde{t}_{f,R}$. Next, we find the confidence interval B and required reliability A that the estimate $\tilde{t}_{f,X}$ for $t_{f,X}$ at each observation point X must satisfy so that the estimate $\tilde{\mu}_f$ for μ_f satisfies the confidence interval β and required reliability α. That is, we want to find the values of B and A so that if for every observation point X we have $P\{|\tilde{t}_{f,X} - t_{f,X}| \le Bt_{f,X}\} \ge A$, then we will have $P\{|\tilde{\mu}_f - \mu_f| \le \beta\mu_f\} \ge \alpha$. After we find the values for B and A, we present a solution to calculate the optimal values of the four parameters n, m, b, and T.

5.1 Individual Reliability Requirements

Individual Required Reliability: The maximum fraction of estimated values $\tilde{\mu}_f$ that can violate the requirement of $|\tilde{\mu}_f - \mu_f| \leq \beta\mu_f$, while the overall estimate still satisfies the required reliability α, is $1-\alpha$. Thus, the maximum fraction of estimates $\tilde{t}_{f,X}$ at either observation points of S and R that can violate the requirement $|\tilde{t}_{f,X} - t_{f,X}| \leq Bt_{f,X}$ must be no greater than $(1-\alpha)/2$. Thus,

$$A = 1 - (1-\alpha)/2 = (1+\alpha)/2 \tag{11}$$

Individual Confidence Interval: The estimate $\tilde{\mu}_f$ obtained by COLATE needs to satisfy the requirement of $|\tilde{\mu}_f - \mu_f| \leq \beta\mu_f$ with probability of at least α. As $\tilde{\mu}_f = \frac{1}{p_f}(\tilde{t}_{f,R} - \tilde{t}_{f,S})$ and $\mu_f = \frac{1}{p_f}(t_{f,R} - t_{f,S})$, the confidence interval requirement $|\tilde{\mu}_f - \mu_f| \leq \beta\mu_f$ becomes:

$$= \frac{\left|(\tilde{t}_{f,R} - t_{f,R}) - (\tilde{t}_{f,S} - t_{f,S})\right|}{\left|(\tilde{t}_{f,R} - \tilde{t}_{f,S}) - (t_{f,R} - t_{f,S})\right|} \leq \beta(t_{f,R} - t_{f,S})$$

The largest value of $\left|(\tilde{t}_{f,R} - t_{f,R}) - (\tilde{t}_{f,S} - t_{f,S})\right|$ is $Bt_{f,R} + Bt_{f,S}$, which must be no greater than $\beta(t_{f,R} - t_{f,S})$.

$$Bt_{f,R} + Bt_{f,S} \leq \beta(t_{f,R} - t_{f,S})$$

Thus, we get

$$B \leq \beta\left(\frac{t_{f,R} - t_{f,S}}{t_{f,R} + t_{f,S}}\right) \tag{12}$$

To determine B for a given network, we conduct measurement of $(t_{f,R} - t_{f,S})/(t_{f,R} + t_{f,S})$ on the network to find the appropriate value so that Equation (12) statistically holds.

5.2 Reliability Centered Parameter Selection

As we have four unknown parameters (*i.e.*, n, m, b, and T), we need at least four equations so that we can calculate the values of these parameters by solving the four equations. Next we develop these four equations.

Equation 1: Let M be the total number of bits of the RAM that an observation point can allocate for storing the counter vector, which requires $n \times b$ bits. Thus,

$$n \times b = M \tag{13}$$

Equation 2: Based on Lemma 1, the sum of all the time stamps of all packets of all flows is equally divided among all n counters on average. Thus, the value of each counter on average can go up to T/n. Thus, the number of bits in each counter, which is b, needs to satisfy the following equation.

$$b = \log_2\left\{\frac{T}{n}\right\} + 1 \tag{14}$$

Note that we add 1 in the R.H.S of this equation to double the capacity of each counter to avoid overflows.

Equation 3: As the expected value of a counter in a counter subvector, which is specified in Equation (1), should never exceed the maximum capacity of the counter, we have

$$\frac{t_{f,X}}{m} + \frac{T - t_{f,X}}{n} \leq 2^b - 1$$

Let $t_{f,X}^{max}$ be the maximum value of $t_{f,X}$ for all flows on a network. Thus, the value of b should satisfy the following equation:

$$\frac{t_{f,X}^{max}}{m} + \frac{T - t_{f,X}^{max}}{n} = 2^b - 1 \tag{15}$$

Here $t_{f,X}^{max}$ can be obtained by some measurement on the sum of the time stamps of all packets on a per-flow basis on the given network.

Equation 4: To achieve the required reliability, $P\left\{|\tilde{t}_{f,X} - t_{f,X}| \leq Bt_{f,X}\right\}$ should at least be equal to its lower bound A, *i.e.*,

$$P\left\{(1-B)t_{f,X} \leq \tilde{t}_{f,X} \leq (1+B)t_{f,X}\right\} = A \tag{16}$$

Based on Equation (1), we can represent $E[C]$ as a function of $t_{f,X}$; denoting this function by g, we have $E[C] = g\{t_{f,X}\}$. Thus, $t_{f,X} = g^{-1}\{E[C]\}$. Let $\tilde{C}$ be the observed value of $E[C]$. Then, we have $\tilde{t}_{f,X} = g^{-1}\{\tilde{C}\}$. Equation (16) becomes

$$\begin{aligned} A &= P\left\{(1-B)t_{f,X} \leq g^{-1}\{\tilde{C}\} \leq (1+B)t_{f,X}\right\} \\ &= P\left\{g\left\{(1-B)t_{f,X}\right\} \leq \tilde{C} \leq g\left\{(1+B)t_{f,X}\right\}\right\} \end{aligned} \tag{17}$$

Similarly, based on Equation (2), we can represent standard deviation of C as a function of $t_{f,X}$; denoting this function by h, we have $\mathrm{Var}(C) = h^2\{t_{f,X}\}$. Based on the fact that the variance of a random variable reduces by m times if the random event is repeated m times, by observing the values of C from m counters, the variance of C becomes $\frac{h^2\{t_{f,X}\}}{m}$ and the standard deviation of C becomes $\frac{h\{t_{f,X}\}}{\sqrt{m}}$. Let Z denote $\frac{\tilde{C} - g\{t_{f,X}\}}{h\{t_{f,X}\}/\sqrt{m}}$. Thus, Equation (17) becomes

$$P\left\{\frac{g\left\{(1-B)t_{f,X}\right\} - g\left\{t_{f,X}\right\}}{h\left\{t_{f,X}\right\}/\sqrt{m}} \leq Z \leq \frac{g\left\{(1+B)t_{f,X}\right\} - g\left\{t_{f,X}\right\}}{h\left\{t_{f,X}\right\}/\sqrt{m}}\right\} = A \tag{18}$$

By the central limit theorem, Z approximates a standard normal random variable. The area under the standard normal curve gives the success probability, which is the required reliability in our context. As our confidence interval requirement is symmetric on both the upper and lower sides of $t_{f,X}$, we can represent the required reliability A in terms of a constant k as follows:

$$P\left\{-k \leq Z \leq k\right\} = A \tag{19}$$

Let Φ be the cumulative distribution function (CDF) of a standard normal distribution and $\mathrm{erf}\{.\}$ be the standard error function, we get

$$P\left\{-k \leq Z \leq k\right\} = \Phi(k) - \Phi(-k) = \mathrm{erf}\left\{\frac{k}{\sqrt{2}}\right\} \tag{20}$$

From Equations (19) and (20), we get

$$k = \sqrt{2}\ \mathrm{erf}^{-1}\left\{A\right\}$$

We observe that the absolute values of the upper and lower bounds of Z in Equation (18) are the same. Thus, equating the lower bound with $-k$ or the upper bound with k results in the following equation:

$$B^2 t_{f,X}^2 = \frac{k^2 n^2 m}{n-m}\left\{\left(\frac{t_{f,X}}{m}\right)\left(1 - \frac{1}{m}\right) + \left(\frac{T - t_{f,X}}{n}\right)\left(1 - \frac{1}{n}\right)\right\} \tag{21}$$

By rearranging Equation (21), we get.

$$\begin{aligned} B^2 = &\frac{1}{t_{f,X}}\left[\frac{k^2 n^2 m}{n-m}\left\{\left(\frac{1}{m}\right)\left(1 - \frac{1}{m}\right) - \left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right)\right\}\right] \\ &+ \frac{1}{t_{f,X}^2}\left[\frac{k^2 n^2 m}{n-m}\left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right)T\right] \end{aligned}$$

This equation shows that B is inversely proportional to $t_{f,X}$ when other parameters are fixed. This makes intuitive sense because the more packets in flow f in passing observation point X (*i.e.*, the larger $t_{f,X}$ is), the more timing information we can obtain from the packets, and the smaller confidence interval can be achieved. Thus, we should use the statistically minimum observable value of $t_{f,X}$, denoted $t_{f,X}^{min}$, for

the given network, in Equation (21). Here $t_{f,X}^{min}$ can be obtained by some measurement on the sum of the time stamps of all packets on a per-flow basis on the given network. The parameter values obtained using $t_{f,X}^{min}$ in Equation (21) will ensure that the estimates for all flows whose sum of time stamps are $\geq t_{f,X}^{min}$ satisfy $P\left\{|\hat{t}_{f,X} - t_{f,X}| \leq Bt_{f,X}\right\} \geq A$. By replacing $t_{f,X}$ by $t_{f,X}^{min}$ in Equation (21), we get

$$B^2 {t_{f,X}^{min}}^2 = \frac{k^2 n^2 m}{n-m}\left\{\left(\frac{t_{f,X}^{min}}{m}\right)\left(1-\frac{1}{m}\right) + \left(\frac{T - t_{f,X}^{min}}{n}\right)\left(1-\frac{1}{n}\right)\right\} \quad (22)$$

Solving Equations: COLATE takes M, α, β, $t_{f,X}^{min}$, and $t_{f,X}^{max}$ as input. The values of required reliability α and confidence interval β are provided by network operators. The value of RAM space M depends on the amount of RAM available at an observation point. For $t_{f,X}^{min}$ and $t_{f,X}^{max}$, network operators can obtain them by measurements on targeted flows in the given network. With the values of M, α, β, $t_{f,X}^{min}$, and $t_{f,X}^{max}$, COLATE simultaneously solves the four equations (*i.e.*, (13), (14), (15), and (22)) to obtain the appropriate values of the four parameters n, m, b, and T. To simultaneously solve these equations, we express m, b, and T in terms of n using Equations (13), (14), and (15) and replace them in Equation (22). This results in an expression with only one unknown parameter n. We numerically solve this expression to obtain n and then the other three unknown parameters.

RAM Space vs. Storage Space: From the simultaneous solution of these four equations, we have an interesting observation that COLATE requires smaller amount of permanent storage space for storing counter epochs when it is allocated with more RAM for storing counter vectors. Figure 3 plots an example graph of the permanent storage size vs. RAM size for COLATE.

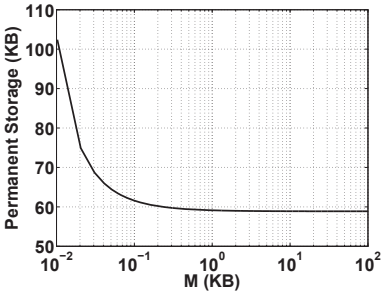


Figure 3: Permanent storage vs. RAM size

While this observation seems surprising, it makes intuitive sense because the sum of the two maximum values of two b-bit numbers is $2 \times 2^b = 2^{b+1}$ whereas the maximum value of a $2b$-bit number is $2^{2b} >> 2^{b+1}$. As we increase the total number of bits in a counter vector, *i.e.*, M, the counter value threshold T increases as shown in Figure 4. This implies that the frequency of writing the counter vector into permanent storage is reduced, and although each counter epoch takes more space, the overall required storage space is reduced as shown in Figure 3. The M value at the knee of the curve in Figure 3 represents the best tradeoff point between RAM space and permanent storage space.

5.3 Flexibility in Parameter Selection

If we only measure average per-flow latency, each observation point can choose its own values for the parameters of n, m, b, and T based on its available resources and traffic condition without global coordination among observation points.

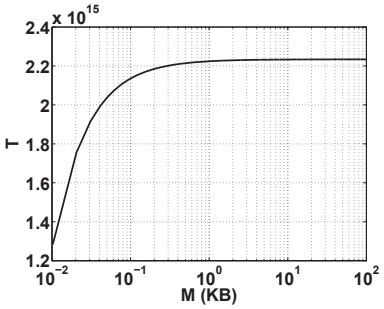


Figure 4: Threshold T vs. RAM size

If we also need to measure standard deviation, then all observation points need to use the same value of m because Theorem 4 requires that the two sets $\mathcal{Y}_{lS}$ and $\mathcal{Y}_{lR}$ contain the time stamps from the same set of packets at the two observation points, which is possible only when the value of m is the same at both observation points. The remaining three parameters can still be chosen independently at each observation point.

6. PERFORMANCE EVALUATION

We implemented COLATE in Matlab. We also implemented RLI [23] in Matlab for comparison purposes. We did not implement LDA [21] and MAPLE [24] because LDA can not provide per-flow latency measurement and MAPLE requires attaching time stamps to every packet *i.e.*, MAPLE is not a latency estimation scheme but a storage scheme. In this section, we present our evaluation results of COLATE in comparison with RLI. We first give details of the three network traces that we used. Second, we evaluate the accuracy of COLATE as well as the impact of RAM space on permanent storage space used by COLATE. Last, we compare COLATE with RLI and with Count-Min (CM) sketch [19], a summarizing data structure for queries on data streams.

6.1 Network Traces

To evaluate COLATE, we need real packet traces with high-resolution time stamps collected simultaneously from at least two observation points. Unfortunately, no such traces are publicly available. Thus, we resort to three real network traces where each is collected at a single observation point at a time. These traces include CHIC [2], ICSI [30], and DC [15]. CHIC is a backbone header trace, published by CAIDA, which includes the arrival times of packets at a 10GigE link interface. We used traces generated from 5 minutes of packet capture. Note that the authors of RLI and MAPLE also evaluated their schemes on the header trace of this backbone link collected by CAIDA. ICSI is an enterprise network traffic trace, collected at a medium-sized site, which includes the arrival times of packets on an ethernet link for a duration of over 41.1 hours. ICSI is available in the form of 41 trace files collected at 17 different ports in an enterprise network. DC is a data center traffic trace, collected at a university data center, which includes the arrival times of packets on an ethernet link for a duration of a little more than an hour. DC is available in the form of 20 trace files collected at the same port. Figure 5 shows the CDFs of sizes of flows in each trace. We observe that the traces contain both mice flows as well as elephant flows. Table 2 reports the total duration, number of packets, number of flows, and average data rate of each trace.

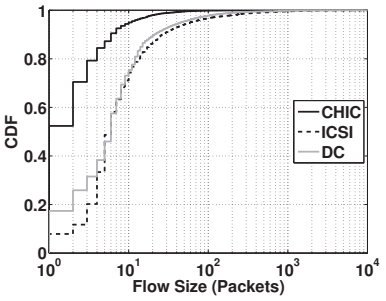


Figure 5: CDFs of flow sizes

As these network traces contain only the arrival time stamps of packets, we adopt the simulation strategy used by RLI and MAPLE, which simulates the traversal of packets in each trace through a queue to get a departure time stamp for each packet and uses random early detection (RED) [18] as the queue management strategy because RED is most popular. As modern routers typically use a queue size that can hold 0.5 seconds of traffic at their maximum line rates, we also use the same queue size. For the remaining parameters of RED queue management strategy, we use $\min_{th} = 0.475\times$queue size, $\max_{th} = 0.95\times$queue size, $w_q = 0.002$, and $\max_p = \frac{1}{50}$ as per the guidelines in [18].

Table 2: Summary of network traces

Trace	Duration	# pkts	# flows	Mbps
CHIC	5 mins	37.3M	3.01M	411
ICSI	41.1 hours	46.9M	0.387M	1.31
DC	1.08 hours	19.9M	0.439M	49.6

6.2 COLATE Accuracy

Now we evaluate the accuracy of the average and standard deviation of the flows in the three network traces estimated by COLATE.

6.2.1 Average Latency

We evaluated COLATE for both the scenario of only two observation points (where one sender sends and one receiver receives) and that of more than two observation points (where multiple senders send and multiple receivers receive). For the scenario with more than two observation points, we choose three observation points forming a triangle topology where everyone sends to and receives from everyone else. We choose three observation points and the triangle topology for the sake of simplicity as the number of senders and receivers does not affect the accuracy of COLATE. For a triangle topology, there are 6 unidirectional links. We used the largest 6 of the 41 trace files from ICSI data set, each trace file representing the traffic on one of the 6 links. This choice is arbitrary.

We performed our evaluation for three different accuracy requirements: low ($\alpha = 0.90$, $\beta = 0.10$), medium ($\alpha = 0.95$, $\beta = 0.05$), and high ($\alpha = 0.99$, $\beta = 0.01$). For each of these three accuracy requirements, we evaluated COLATE using three values of available RAM: small ($M = 1$MB), medium ($M = 10$MB), and large ($M = 100$MB). We obtained the values of $t_{f,X}^{min}$ and $t_{f,X}^{max}$ from simple measurement of the network traces. We calculated the values of the parameters b, m, n, and T using the method described in Section 5. For example, for $M = 1$MB, $\alpha = 0.95$, and $\beta = 0.05$, typical values of these parameters are $b = 19$, $m = 20$, $n = 455334$, and $T = 8 \times 10^{10}$.

Our results show that COLATE always achieves the required reliability. Figures 6(a), 6(b), and 6(c) show the CDFs of the observed values of β *i.e.*, $|\tilde{\mu}_f - \mu_f|/\mu_f$, for the average latency estimated by COLATE for the three traces under the scenario of one sender and one receiver using the low, medium, and high accuracy requirements, respectively. Figures 8(a), 8(b), and 8(c) show the CDFs of the observed values of β *i.e.*, $|\tilde{\mu}_f - \mu_f|/\mu_f$, for the average latency estimated by COLATE for the six links under the scenario of multiple senders and receivers using the low, medium, and high accuracy requirements, respectively. The horizontal line in each of these figures shows the required reliability. We see that every plot of CDF always crosses the horizontal line for an observed value of β that is smaller than the required confidence interval. This shows that COLATE always achieves the required reliability. Due to lack of space, we only show plots for $M = 10$MB. Observations from $M = 1$MB and $M = 100$MB are the same.

6.2.2 Standard Deviation

Our results show that the relative error in the standard deviation estimates of over 91% *flows is less than* 0.05 *with only 1000 virtual repetitions.* Relative error is defined as (`actual value` − `estimated value`)/`actual value`. Figure 7 plots the CDFs of the relative errors in standard deviation estimated by COLATE for each of the three traces.

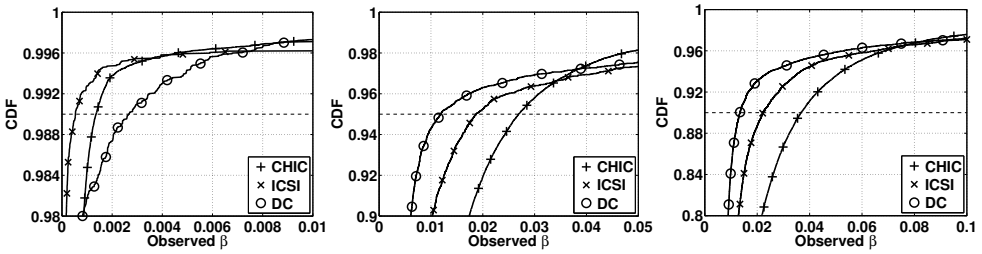
Our results also show that the percentage of flows, for which the relative error is less than 0.05*, increases with the increase in the number of virtual repetitions.* Figure 9 plots this percentage versus the number of virtual repetitions for the three traces. With 10^5 iterations, this percentage reaches 98%. Although 10^5 iterations may take some time depending on the available computing power, it is not an issue as the estimation of standard deviation is an offline process and does not have to keep up with high line rates.

6.3 RAM and Storage Size

Our results show that COLATE uses less than 0.1 bit of permanent storage per packet. Figure 10 shows the bar graph of the number of bits per packet used by each of the three traces for all three values of M, when $\alpha = 0.99$ and $\beta = 0.01$. This small size of storage required per packet results in a very low frequency of transferring the counter vectors from RAM to permanent storage. For example, for ICSI network trace, COLATE transfers a counter vector to SSD every 24.6 hours when $M = 1$MB, $\alpha = 0.95$, and $\beta = 0.05$. Transferring 1MB of content from RAM to SSD once a day is trivial for modern devices. The number of bits per packet in permanent storage decreases with the increase in RAM size M, which confirms our analysis based on Figures 3 and 4. However, this decrease is hard to observe from Figure 10 because the difference is small. Nevertheless, this difference becomes significant for longer time durations (on the order of say days and weeks).

6.4 Comparison with RLI

Our results show that COLATE always achieves higher accuracy than RLI. RLI requires two inputs, namely the lower and upper limits of the Probe packet Injection Rate (PIR). The authors of RLI used the lower limit as 1 probe packet per 300 regular packets and the upper limit as 1 probe packet per 10 regular packets in [23]. We first evaluated RLI using this pair of PIR values. Because the accuracy of RLI increases as



(a) $\alpha = 0.99, \beta = 0.01$ (b) $\alpha = 0.95, \beta = 0.05$ (c) $\alpha = 0.9, \beta = 0.1$

Figure 6: CDF of observed β in average estimate (one sender & one receiver)

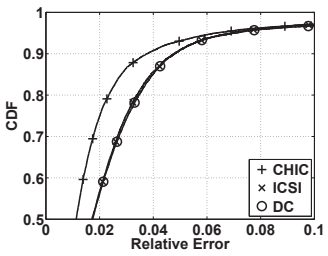
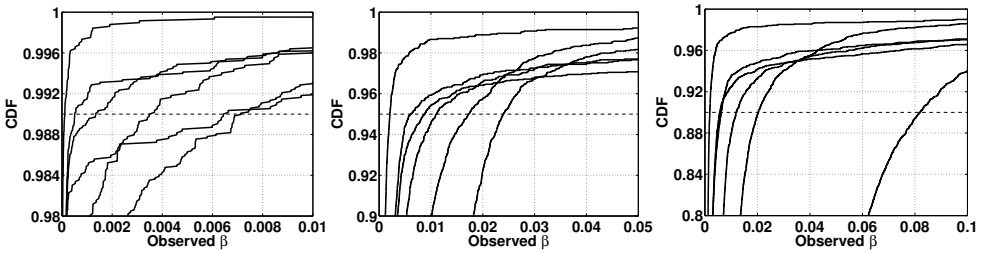


Figure 7: CDFs of relative errors in STD



(a) $\alpha = 0.99, \beta = 0.01$ (b) $\alpha = 0.95, \beta = 0.05$ (c) $\alpha = 0.9, \beta = 0.1$

Figure 8: CDF of observed β in average estimate (multiple senders & receivers)

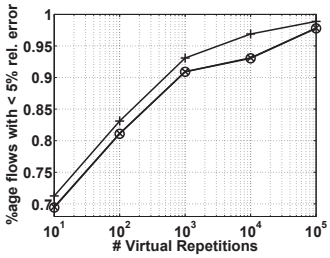


Figure 9: Relative error in STD vs. #repetitions

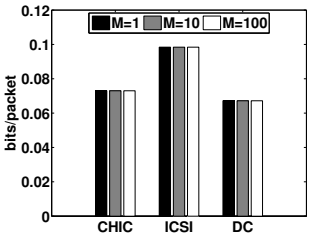


Figure 10: Storage bits per packet

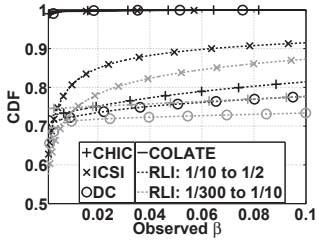


Figure 11: Comparison of delay estimates

PIR increases, to improve the estimation accuracy of RLI, although at the cost of larger bandwidth usage, we also evaluated RLI using much higher PIRs – 1 probe packet per 10 regular packets as the lower limit and 1 probe packet per 2 regular packets as the upper limit. Figure 11 plots the CDFs of the observed value of β in the average latency estimated by RLI in the three traces for these two configurations of PIRs. For comparison, we also plot the CDF of the observed value of β in the estimates obtained using COLATE for high accuracy requirement ($\alpha = 0.99, \beta = 0.01$). Note that the observed value of β is essentially the relative error in the estimated values of average latency. Figure 11 shows that the relative error of COLATE is much smaller than that of RLI. This relative error can be made arbitrarily small by specifying smaller values of β and larger values of α. This figure further shows that the relative error of RLI is smaller when PIR is larger. With the PIR proposed by the authors, on average, only 77% flows have relative error less than 5%. At this rate, on average, RLI inserts one probe packet after 21.66 regular packets. With our high PIR configuration, on average, only 81% flows have a relative error less than 5%, but at this rate, on average, RLI inserts one probe packet every 4.78 regular packets in the three traces. Table 3 shows the average number of regular packets after which RLI inserts a probe packet for each of the three traces. In contrast, COLATE does not insert any probe packet at the cost of a small amount of memory at observation points.

Table 3: Average number of regular packets after which RLI inserts a probe packet

Trace	# reg. pkts 1:300 to 1:10	# reg. pkts 1:10 to 1:2
CHIC	18.66	10.0
ICSI	17.19	2.97
DC	245.7	9.06

6.5 Comparison with Count-Min Sketch

Count-Min (CM) sketches can theoretically be used for latency measurement but practically, there is a fundamental limitation. Let t_{f_i} represent the sum of time stamps of all packets in flow f_i and let there be j flows in total whose time stamps need to be stored for latency measurements. We can use a CM-sketch to store time stamps of multiple flows and obtain the estimate $\tilde{t}_{f_i}$ of the sum of time stamps of any flow f_i as per the method described in [19]. The estimate $\tilde{t}_{f_i}$ obtained through CM-sketch can satisfy the condition $\tilde{t}_{f_i} \leq t_{f_i} + B \times \sum_{\forall j} t_{f_j}$ with probability α. This is problematic because we require the estimate to satisfy the condition $\tilde{t}_{f_i} \leq t_{f_i} + B \times t_{f_i}$ with probability α. Therefore, to achieve the required reliability using CM-sketch, we need to ensure that $\sum_{\forall j} t_{f_j} \leq t_{f_i}$, which is possible if and only if we do not add time stamps of any flow other than f_i to the CM-sketch. Consequently we need to maintain a CM-sketch for each flow, which results in large memory requirements. For example, for $\alpha = 0.95$ and $\beta = 0.05$, CM-sketch requires 165 counters per flow and 3 hash functions and 3 memory accesses per packet. Assuming the same counter size of 19 bits as for COLATE in this scenario, CM-sketch requires $165 \times 19 = 3135$ bits per flow. In comparison, COLATE requires 0.1 bit per packet. The memory requirement of CM-sketch matches that of COLATE only if each flow has at least 31350 packets, which is impractical as seen in Figure 5. The number of memory accesses and the number of hash functions per packet for CM-sktech are always greater than those for COLATE.

7. CONCLUSION

The key contribution of this paper is in proposing an accurate and efficient per-flow latency measurement scheme without packet probing and time stamping. The key novelty of this paper is that we purposely allow noise to be introduced in recording packet timing information for minimizing storage space and use statistical techniques to denoise the recorded information to obtain accurate latency estimates when latency of a target flow is queried. The key technical depth of this paper is in the mathematical development of the estimation theory that our scheme is based upon. Our theoretical analysis and experimental results show that our scheme always achieves the required reliability. Our scheme has a much smaller processing overhead in terms of number of hash computations and memory updates compared to existing schemes, which further require sending probe packets or attaching time stamps to every packet. Our scheme is scalable in that the amount of memory required at each observation point is only dependent on the number of packets and not on the number of sending and receiving observation points. The memory requirement is so low that a commodity storage device can store time stamps of several years worth of flows.

8. REFERENCES

[1] CAIDA passive network monitors. http://www.caida.org/data/realtime/passive/.

[2] The CAIDA UCSD anonymized 2011 internet traces. http://www.caida.org/data/passive/passive_2011_dataset.xml.

[3] Cedexis. http://www.cedexis.com/.

[4] Corvil claims to minimize network latency. http://www.pcworld.idg.com.au/article/196828/corvil_claims_minimize_network_latency/.

[5] *Fibre Channel Backbone - 5 (FC-BB-5) REV 2*.

[6] IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems.

[7] Sidera. http://www.sidera.net/.

[8] Singapore exchange (sgx) selects corvil for latency management. http://www.corvil.com/News/Press-Releases/Singapore-Exchange-(SGX)-Selects-Corvil-For-Latenc.aspx.

[9] Tokyo stock exchange select corvil. http://www.corvil.com/News/Press-Releases/Tokyo-Stock-Exchange-Select-Corvil.aspx.

[10] Turbobytes. http://www.turbobytes.com/.

[11] While london stock exchange selects corvil for low latency network monitoring and analysis solution. http://low-latency.com/article/\%E2\%80\%A6-while-london-stock-exchange-selects-corvil-low-latency-network-monitoring-and-analysis-sol.

[12] Z-Drive R4 and R5 PCIe SSD. http://lensfire.in/2012/01/ocz-launches-new-z-drive-r4-and-r5-pcie-ssd-ces-2012-2012/.

[13] HP expands high-performance computing offering with infiniband solutions from cisco. http://www.hp.com/hpinfo/newsroom/press/2007/070524xa.html, May 2007.

[14] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proc. ACM SoTC*, pages 20–29, 1996.

[15] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proc. ACM IMC*, pages 267–280, 2010.

[16] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *Proc. ACM SIGCOMM*, pages 55–66, 2004.

[17] N. Duffield. Simple network performance tomography. In *Proc. ACM IMC*, pages 210–215, 2003.

[18] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[19] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[20] N. Hua, E. Norige, S. Kumar, and B. Lynch. Non-crypto hardware hash functions for high performance networking ASICs. In *Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 156–166, 2011.

[21] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese. Every microsecond counts: tracking fine-grain latencies with a lossy difference aggregator. In *Proc. ACM SIGCOMM*, pages 255–266, 2009.

[22] A. Kumar, J. J. Xu, J. Wang, O. Spatschekt, and L. E. Lit. Space-code bloom filter for efficient per-flow traffic measurement. In *Proc. IEEE INFOCOM*, pages 1762–1773, 2004.

[23] M. Lee, N. Duffield, and R. R. Kompella. Not all microseconds are equal: fine-grained per-flow measurements with reference latency interpolation. In *Proc. ACM SIGCOMM*, pages 27–38, 2010.

[24] M. Lee, N. Duffield, and R. R. Kompella. A scalable architecture for maintaining packet latency measurements. In *Proc. ACM IMC*, pages 101–114, 2012.

[25] M. Lee, S. Goldberg, R. R. Kompella, and G. Varghese. Fine-grained latency and loss measurements in the presence of reordering. In *Proc. ACM SIGMETRICS*, pages 329–340, 2011.

[26] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani. Counter braids: a novel counter architecture for per-flow measurement. In *Proc. ACM SIGMETRICS*, pages 121–132, 2008.

[27] B. Lynch and S. Kumar. Smart memory for high performance network packet forwarding. In *Proc. Hot Chips Symposium*, 2010.

[28] R. Martin. Wall street's quest to process data at the speed of light. *Information Week*, 4(21), 2007.

[29] M. J. Miller. Bandwidth engine serial memory chip breaks 2 billion accesses/sec. In *Proc. Hot Chips Symposium*, 2011.

[30] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *Proc. ACM IMC*, pages 15–28, 2005.

[31] M. Ramakrishna, E. Fu, and E. Bahcekapili. Efficient hardware hashing functions for high performance computers. *IEEE Transactions on Computers*, 46(12):1378–1381, 1997.