Opportunistic Flow-level Latency Estimation using Consistent NetFlow

Myungjin Lee, Nick Duffield, Ramana Rao Kompella

***Abstract*—The inherent measurement support in routers (SNMP counters or NetFlow) is not sufficient to diagnose performance problems in IP networks, especially for flow-specific problems where the aggregate behavior within a router appears normal. Tomographic approaches to detect the location of such problems are not feasible in such cases as active probes can only catch aggregate characteristics. To address this problem, in this paper, we propose a Consistent NetFlow (CNF) architecture for measuring per-flow delay measurements within routers. CNF utilizes the existing NetFlow architecture that already reports the first and last timestamps per-flow, and proposes hash-based sampling to ensure that two adjacent routers record the same flows. We devise a novel Multiflow estimator that approximates the intermediate delay samples from other background flows to significantly improve the per-flow latency estimates compared to the naive estimator that only uses actual flow samples. In our experiments using real backbone traces and realistic delay models, we show that the Multiflow estimator is accurate with a median relative error of less than 20% for flows of size greater than 100 packets. We also show that Multiflow estimator performs 2-3 times better than a prior approach based on trajectory sampling, at an equivalent packet sampling rate.**

***Index Terms*—NetFlow, coordinated streaming, per-flow latency estimation, network management.**

I. Introduction

Although IP networks were designed to be best-effort, an increasing number of applications today require performance guarantees. These range from multimedia applications such as voice-over-IP, video, multi-player online gaming, to performance-critical enterprise applications such as online trading. ISPs often provide tight SLA guarantees to ensure that the performance of their network matches up with the expectations of their customers that want to run these applications. For example, Sprint offers 55ms RTT across sites within the US [1]. SLA violations typically have heavy financial implications for the ISPs; they, therefore, care about ensuring the health of their network.

IP networks, however, are notoriously hard to measure and debug today. Apart from SNMP counters [2] such as for interface packet drops, there exists very little support in the routers for diagnosing performance problems. For example, routers today do not even report aggregate performance statistics (*e.g.*, average delay, jitter, loss rate) of the forwarding paths (from one interface to another). Such performance statistics would be useful for operators to diagnose the root cause of end-to-end problems, and take necessary steps to fix it. Without these intrinsic measurement capabilities, network operators today are forced to use external means to infer such statistics. For instance, ISPs today routinely monitor their networks with the help of measurement boxes that inject 'active probes' between routers. From the path properties observed by these probes, operators typically use inference algorithms to isolate the location of the problem.

Several inference algorithms based on tomographic approaches exist in the literature today (*e.g.*, [3], [4], [5]). As noted before [4], [3], however, the problem is generally under-constrained in this sense that, depending on topology on disposition of measurement points, not all individual link performances are visible. Another major limitation is that they only provide aggregate statistics, which are not in general sufficient to diagnose and debug flow-specific network problems. For example, a flow (that belongs to a given customer's application) may have passed a router exactly when the busy period started for the router's queue, because of which that particular flow may have obtained high latency (in turn affecting its throughput). However, when packet delays are aggregated over an interval, it may appear that the forwarding path within the router is functioning normally. Thus, while ensemble averages may be sufficient for detecting generic SLA violations, they may not be enough for diagnosing any violations on a *per-customer* basis.

Unfortunately, despite their limitations in coverage and granularity, network operators still rely on active probes as they do not have any other alternative today. From the router vendors' point of view, routers are already burdened with the ever increasing suite of protocols they need to support and extremely high line rates, making it quite challenging to provide per-flow latency measurement capability. From the network operator's perspective, however, obtaining such measurements directly from the routers would definitely make the task of debugging their network easier. As a first attempt to bridge this gap between the capabilities of routers and the ISP needs, in this paper, we consider *the problem of obtaining hop-level latency measurements on a per-flow basis*.

At first glance, this goal may appear challenging to achieve. We need precise time-synchronization, which has been traditionally hard, as well as coordination between routers, which is often met with resistance since routers are already overburdened. The ground has, however, significantly shifted in terms of the capabilities of routers today. For instance, routers are being equipped with sophisticated time-synchronization

Portions of this manuscript appeared in IEEE INFOCOM 2010. This work was supported in part by the National Science Foundation through award CNS-0831647, and Cisco Systems.

M. Lee and R. R. Kompella are with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 (e-mail: mjlee@purdue.edu, kompella@cs.purdue.edu).

N. Duffield is with AT&T Labs–Research, 180 Park Ave, Building 103, Florham Park, NJ 07932. (e-mail: duffield@research.att.com.)

primitives such as the IEEE 1588 protocol [6], GPS-based clocks, etc. There have been significant innovations in designing router primitives that facilitate coordination between routers without explicit communication [7]. Given these modern capabilities, we believe the time is ripe to investigate scalable mechanisms for obtaining and reporting per-flow latency measurements.

To describe our approach for obtaining per-flow latencies, we begin with sampled NetFlow widely deployed in today's routers as our starting point. There are two key ideas: The first idea is to exploit the fact that NetFlow already maintains two timestamps corresponding to the first and last packets of a flow. We ensure that two NetFlow processes running at two adjacent routers maintain the same flows and timestamps for the same first and last packets. This *Consistent NetFlow* architecture enables us to obtain two delay samples for the flow. We can easily achieve this using hash-based packet sampling. The second idea is to *opportunistically*[1] refine the latency estimates of a flow by utilizing the delay samples obtained from other flows that lie within the flow's duration. We show how our novel *Multiflow estimator* based on this idea provides significantly better estimates of per-flow latencies compared to the Endpoint estimator that uses the two delay samples obtained using consistent NetFlow and active probe approach that assigns an aggregate latency estimate as per-flow latency estimates.

There is one main challenge that remains in the above architecture. Our architecture provides per-flow latency estimates for only the sampled flows and not *all* flows. However, this problem is not fundamental to our approach; it is just an artifact of the fact that current routers often report only a subset of flows (*e.g.*, by sampling packets) to control memory and CPU resources. As technology improves, so will the router resources to support higher sampling rates, and the fact that CNF lies within the NetFlow makes this scaling automatic and straightforward. While the fact that CNF can provide reasonable estimates of per-flow latencies within routers is significant, it is not intended to serve as a replacement for sophisticated measurement frameworks (*e.g.*, GPS-enabled setup used in [8]). Instead, we envision our approach to provide low-cost ways of continuously monitoring latency characteristics of flows across routers. Thus, when any SLA violations are observed in the network, our approach will allow easy localization of the router or interface at which such a violation occurred.

Thus, the *main contributions* of this paper include the design of Consistent NetFlow (discussed in Section III-C)), an opportunistic latency estimator called the Multiflow estimator (discussed in Section III-F) that provides per-flow latency estimates on a per-hop basis, and the empirical evaluation of its accuracy using real backbone traces under different delay models and real traces generated by synthetic source (in Section IV). Our results indicate that the Multiflow estimator provides a median relative error of less than 20% for flows greater than 100 packets. In contrast, we observe that trajectory sampling (when adapted for this purpose) is 2-3 times worse than our Multiflow estimator.

[1]We refer to this as opportunistic latency estimation since the original intent of the timestamps is not for latency estimation purposes.

II. Related Work

NetFlow [9] is the main basis for our approach. While several variants of NetFlow (*e.g.*, Adaptive NetFlow [10], Flow slices [11]), and other passive measurement data structures [12], [13], [14] have been proposed, they are mainly geared towards measuring flow characteristics such as number of packets, bytes, flags in a scalable fashion. To the best of our knowledge, we are the first to propose integrating latency measurements into the NetFlow of aggregated flow reporting, and we propose a concrete architecture and estimators to achieve such an integration.

Active probing techniques [15], [16], [17], [18], [19] have been important for network operators to monitor their network. Systems such as RIPE's TTM [18] rely on GPS to guarantee time-synchronization in two measurement vantage points. The authors in [15] compared the accuracy of GPS synchronized measurement box with that of NTP. Some techniques provide per-hop latency statistics, but per-flow latency measurements are not provided by any of these tools.

A large body of work on tomography approaches exists for predicting the average latency characteristics of links and router forwarding paths. For example, the work by Chen *et al.* in [5] and Duffield *et al.* in [4] solve the problem of predicting the per-hop loss and latency characteristics based on end-to-end measurements (*e.g.*, conducted using active probing tools mentioned above) and routing information obtained from the network. Our work differs from theirs in two major fronts. First, we use passive measurements obtained from routers and as such do not inject any active probes. Second, our approach involves per-flow latency estimation; based on a few collected samples, the problem we attempt to solve is whether we can accurately estimate the latency characteristics on a per-flow basis.

Numerous measurement studies [20], [8], [21], [22] to characterize latency have been reported in the literature. Single-hop delay characteristics have been studied extensively in [8], [21]. Choi *et al.* [22] study latency characteristics across PoPs. These studies however are characterization studies and do not propose any architecture to allow network operators to conduct per-flow latency estimates.

Our work shares some similarity with trajectory sampling proposed by Duffield *et al.* in [7]. The main idea behind trajectory sampling is to collect consistent packet samples throughout the network to facilitate various direct measurements. They propose a technique called consistent hashing that we borrow in our effort for ensuring that consistent streams of packets are observed at two routers. The notion of a flow plays a fundamental role in our work (and therefore the NetFlow collection framework) while trajectory sampling relies on flat packet labels with no flow-level aggregation. Passive measurement of loss and delay by directly comparing trajectory samples of the same packet observed at different points has been studied by Zseby *et al.* in [23] and Duffield *et al.* in [24]. Although our proposal is targeted toward the

aggregate reporting paradigm on NetFlow, our approach could potentially also be used to augment collector side analysis in packet-level passive delay measurement of [23], [24].

While we have deliberately restricted our effort to primitives which are either already available in routers today or (being) standardized, there have been prior approaches that attempted to obtain direct performance measurements from routers. For example, Machiraju *et al.* suggest a measurement-friendly network (MFN) architecture as a router primitive which uses hop-dependent priority queuing to address issues related to multi-hop queuing effect and to control intrusiveness of active measurement [25]. This architecture not only requires a lot of intrusive changes in routers, but also does not automatically provide per-flow latency estimates.

There are a couple of previous works [26], [27] to obtain latency statistics passively. Given a pair of routers and the packet stream traversing these two routers, Packet Doppler in [26] is a technique to observe how packets that arrived at a router in a window interval get distributed in a discrete time series of the same window interval at the other router. However, it does not provide per-flow latency measurements. Another very recent work by Kompella *et al.* in [27] argues for a high-speed latency detection data structure called lossy difference aggregator (LDA) that can be implemented in routers at high speeds for measuring various latency characteristics with μsecond precision. Our work while sharing some similarity has two fundamental differences. First, our goal is not to propose a new data structure, but instead leverage existing passive measurement framework in the form of NetFlow as much as possible. Second, we wish to provide per-flow latency measurements whereas LDA is designed to report only aggregate characteristics. If we relax the requirement that we only harness existing primitives, there might be a way to combine the insights in LDA with our architectural framework. We plan to pursue this as future work.

III. Flow collection architecture

In this section, we first present an overview of the current NetFlow-based flow collection architecture. Next, we discuss basic assumptions that our approach relies on before we motivate our *Consistent NetFlow* architecture for supporting delay measurements and pin-point the changes required in the current NetFlow architecture to support them. We then describe how we can obtain per-flow delay measurements from this architecture using *flow correlation*. Next, we discuss our central premise of delay correlation between closely separated packets, and give some analytical and heuristic justification. We proceed to describe two delay estimation methods. In the Endpoint method we use only delay of the first and last flow packets to estimate the average delay of packets in a flow. In the Multiflow method we also use delays of the first and last packet of other flows to supplement the delay average. A Hybrid estimator combines the best performance of each. We describe statistical properties of these estimators. We also want to understand the inherent accuracy with which packet delays from one flow can be used to estimate packet delays of another. To this end, we construct a piecewise linear interpolation of delays from other flow packets. Although this construction does not yield an estimator we can use in practice, it does serve as a reference point by providing the best likely performance attainable by any method which is able to exploit supplemental packet delays reported from different flows.

A. Current NetFlow architecture

In today's flow collection architecture, individual routers in a network run NetFlow [9] on various interfaces. While NetFlow could potentially be turned on in both directions of traffic on a line card, it is customary for ISPs to run NetFlow typically only on the ingress direction of a given interface. For example, in Figure 1, we show a network with three routers $R1$, $R2$, and $R3$. NetFlow instances labeled A and D are in the ingress directions of two interfaces of router $R1$. Flows in the egress directions of these interfaces of $R1$ will be captured at the ingress direction of routers $R2$ and $R3$.

Given that backbone routers cannot keep up with high line rates (*e.g.*, OC-192 or 10 Gbps), routers typically run a variant called Sampled NetFlow that uses a simple stage of uniform packet sampling to ensure that the processor as well as the memory resources in the router are not overwhelmed because of the huge volume of traffic. Typical sampling rates vary from 0.01 to 0.001 depending on the capacity of the link.

At each of these interfaces, the line-card CPU computes individual flow records from the sampled packet stream. Each flow record is created by aggregating all packets that share some common fields in the header, called the *flow key*. While NetFlow allows multiple flow definitions, the most commonly used flow key is the TCP 6-tuple comprising of the source and destination IP addresses, source and destination ports, the protocol field and finally the interface number. For each flow, NetFlow maintains the total number of packets, total number of bytes, different TCP flags observed and other such statistics. It also records the timestamps of the first and last packets observed for that flow.

Each flow record is kept in the flow cache either until the TCP connection closes (either FIN or RST) or until an inactivity timer expires (no packets for the flow for greater than threshold time) or an activity period is over (maximum amount of time for which a flow remains in a flow-cache). Typical value of the activity period is 30 minutes while inactivity timers are often set for 15 seconds. Once the flow expires, the flow record is exported to a flow collector that aggregates flow records obtained from different routers and performs several post-processing functions such as identifying duplicate flows, sampling flows further to control reporting bandwidth (*e.g.*, using usage-based sampling [28]) and so on.

B. Prerequisites

We first describe the two basic assumptions our approach relies on: (i) Accurate time synchronization, and (ii) FIFO packet ordering.

Time synchronization. We require accurate time synchronization between the two measurement end-points. This is a fundamental requirement for *any* architecture that wishes to enable accurate *one-way* delay measurements. We could

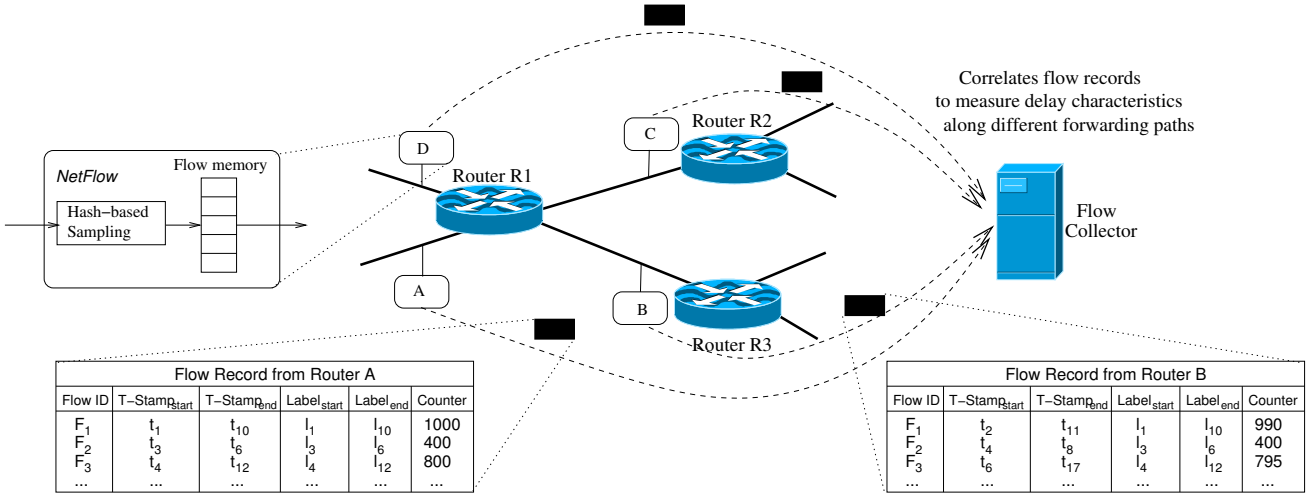


Fig. 1. Overview of our Consistent NetFlow architecture. The main change compared to the current NetFlow architecture is the hash-based sampling stage shown in the NetFlow process. The flow collector aggregates flow records from individual routers and extracts a time-series of delay samples from which per-flow latency measurements are recorded.

leverage the Global Positioning System (GPS) to synchronize clocks at different measurement points; see *e.g.*, [8] where the authors resort to GPS-based clocks to characterize delays within routers. Recent advances (*e.g.*, [29]) in precise time synchronization can achieve a median error of 30μseconds. Router vendors are also increasingly deploying hardware protocols such as IEEE 1588 [6] to synchronize routers within μsecond precision.

There are potential issues of this assumption such as GPS-receiver failure, clock drift, and so forth. These issues exist for any architecture that relies on these techniques. While we do not thoroughly investigate the issues, we conduct basic experiments about how many delay samples are lost due to clock uncertainty; see Section III-D for delay computation method and Section IV-B for experimental results. We leave a thorough study on the impact of these failures as part of our future work.

Packet forwarding order. Another assumption is that the stream of packets follows a serial order (FIFO) between the monitoring points. This is mostly true for latency measurements on a per-hop basis that we are primarily interested in. However, some routers can employ fair queuing mechanisms, and thus packets may pass through separate forwarding paths within a router. In such cases, we require that different forwarding paths be treated differently. A simplest way to achieve this is to annotate each flow with an additional class label that dictates which class the flow belongs to. Within a class, it is reasonable to assume that all packets will follow a FIFO order between the end-points.

C. Consistent NetFlow

We now explain our approach to achieve the goal of retrofitting delay measurements into the NetFlow architecture described before. In particular, we are interested in obtaining delay characteristics of the forwarding path between two interfaces within a router, or a link across routers (from the egress interface of one router to the ingress of the other). Typically, because NetFlow is turned on in the ingress direction, we will only be able to measure the sum of both of these delays. Without loss of generality, let us consider measuring the delay between NetFlow instances A and B in Figure 1; this will essentially measure forwarding path latencies in $R1$ from input port A to output port connected to $R3$, and the link delay between $R1$ and $R3$. We start with the observation that NetFlow instances A and B already store the timestamps of the first and last packets for the set of sampled flows. The *key idea* is that, if both A and B sample the same first and last packets (p_0 and p_n) for some flow recorded by both (in their set of sampled flows), then we can compute the delay observed by these packets p_0 and p_n from the timestamps recorded at both A and B. Of course, as we mentioned in Section III-B, both A and B are time-synchronized for the measurements to be accurate.

One problem with the original NetFlow architecture is that given that both A and B sample packets completely independently, the number of commonly recorded flows is small, and even among the commonly recorded flows, the first and last packets may be different since both routers are seeing streams that are independently sampled. This would mean that the timestamps recorded at both ends are not consistent and hence any delays inferred from these timestamps would, with very high probability, be erroneous. In order to address this problem, we propose a *Consistent NetFlow* (CNF) architecture, that is essentially NetFlow with the additional modification that routers use *hash-based sampling*. Thus, in CNF, routers independently select the same packet set and thus timestamps are consistent across NetFlow instances.

Hash-based sampling. Each router calculates a hash over some set of packet fields that are invariant over the packet path, and the packet is selected for reporting if the hash fall into a given range. When all measuring routers use the same hash function, input fields, and selection range, their selection is consistent in the sense that each packet is selected either at each measurement point or none. Hash-based sampling was

proposed in [7], and has now been standardized in the IETF [30], [31].

With a suitably strong hash function, the average sampling rate can be determined as the size of the selection range divided by the size of the set of possible hash values. Note that the standards are flexible over what fields are used as input to the hash. An input that uses only invariant fields from the flow key—*e.g.*, source and destination IP address and TCP/UDP port numbers—would result in selection of all packets in a subset of flows. Such sampling is often referred to as *flow sampling* [32]. Including hash input from the packet payload too would result in selection that looks statistically similar to uniform random sampling except that packets are selected consistently between different routers (unless otherwise mentioned, we refer to this form of sampling henceforth as just *packet sampling*).

While in theory, both sampling mechanisms can be used to ensure that same flows are recorded, hash-based packet sampling is often preferred because flow sampling can lead to bursts while packet sampling results in smoother selection of packets. This is because the arrival of packets within flows is bursty; with flow sampling the CPU will need to process all packets in a burst of a selected flow. In contrast, packet sampling, assuming a strong hash function, dilutes these bursts. Another strong reason is that established network management tasks, such as traffic matrix estimation, etc., already use Sampled NetFlow. Use of flow sampling instead, potentially requires re-architecting these traditional management tasks, making it harder for deployment. Finally, packet sampling preferentially selects large flows for which our goal of latency measurement may be even more critical.

If there are resource limitations in the measurement reporting infrastructure, then it may become necessary to sample flow records themselves, whether or not packet sampling has been employed. In principle, consistent sampling could be applied to the flow records based on invariant fields in the flow key. However, at this time, no support exists for consistent flow export in the IETF IP Flow Information Export (IPFIX) protocol [33]. Note however that an additional level of sampling in the flow export process can be thought of as similar to reducing the sampling rate for the packet selection itself (that in turn reduces the number of records created). For simplicity, therefore, in our architecture, we assume that all sampled flows are transmitted to the flow collector.

D. Flow Correlation

Expired flow records are transmitted by the NetFlow process on each of the routers to a centralized flow collector as shown in Figure 1. Flow records obtained at any downstream NetFlow instance may contain flows that would have potentially taken different paths at the upstream router. For example, flow records at A will contain flow records that would have traversed both the NetFlow instances at B as well as C. Similarly, the flow records at B will contain flows that may have traversed through the A instance or the D instance. Thus, to obtain the properties of the path between a pair of NetFlow instances, the first step that needs to be performed is to obtain the *intersection set* of the flow records obtained from the endpoints. Thus by considering all the flow records with flow keys present in both A's and B's flow records, we can easily obtain the base set of flow records that are representative of the stream of packets going from A to B.

The second step is to associate flow record measured at different points. Since packet selection is performed by hashing, flow records present at the downstream instance B will have corresponding records at A. In an idealized operation with no packet loss, flow records at each instance would be in a one-one correspondence, and so pairwise association could be performed after sorting records at each instance using the start timestamp. However, this idealized picture can fail to hold in practice for a number of reasons such as (i) packet loss, (ii) differences in application of flow timeouts due to slight difference in inter-packet times, (iii) different cache expiry events due to heavy loads, and (iv) uncertainties in timestamps due to propagation delay and clock synchronization issues. Because of these factors, the demarcation of stream of packet into flows may not be performed identically at the two instances, and so flow timestamps may not refer to the same packets in some cases. This implies we use additional measures to associate flow records between NetFlow instances, and eliminate inconsistencies by ensuring that the timestamps refer to the same packet. We now describe two approaches to achieve this—one based on packet labeling, and the second one based on detailed examination of timing properties.

Mapping packet label to a timestamp. In the first approach, we require that NetFlow report a label (hash of invariant fields of a packet) of the first and last packet in addition to a timestamp. Thus for the purposes of delay measurement, we sort flow records at each NetFlow instance A and B based on timestamp, then associate timestamps only if the corresponding packet labels are identical. This method is still subject to the effects of label collision. However, collisions between flows with widely different timestamps are easily identified at the corresponding (large) delay values excluded; see Section IV-B. Furthermore, we are only concerned with collisions between packets *within* a flow. In any case, the collision rate can be made as small as desired by choosing a large enough packet label.

Timing checks to eliminate inconsistencies. The packet labeling approach, although conceptually straightforward, is not positioned so squarely on the standardization and deployment path as hash-based sampling. Thus, other mechanisms to correlate packet timestamps need to be considered. To this end, we now outline a second mechanism that utilizes timestamps to identify inconsistencies, without requiring changes to the current NetFlow architecture. Here we analyze a packet stream with a single key, since flows with different keys at sender and receiver are not matched. The sender exports a set of flow records i with fields $(\tau_{si,1}, \tau_{si,2}, n_{si}, b_{si})$ being the initial, final packet timestamps number of packets and bytes respectively. The receiver side flows have corresponding fields $(\tau_{ri,1}, \tau_{ri,2}, n_{ri}, b_{ri})$. But note the same index i on sender and receiver side is not assumed to originate in the same set of packets.

We now consider four different quantities e_1^-, e_1^+, e_2, e_3 that

represent timing uncertainty:

- The propagation delay lies in the interval $[e_1^-, e_1^+]$.
- The packet processing latency is less than e_2.
- The clock uncertainty is less than e_3.

Let T_{in} and T_{act} denote the inactive and active flow timeouts respectively. We consider the following criteria (C1)–(C5) applied to matching sender flow i and receiver flow j. In practice, these are applied as follows. We seek pairs of sender and receiver flows that are time compatible, *i.e.*, they satisfy (C1). Since flows of a given key can be ordered by first packet time, this requires only a windowed search. Each (C1) compatible pair is then examined to check they satisfy (C2)–(C4). If so, they can be passed to the analysis stage and we return to seek another (C1) compatible pair. If the flows fail any of the conditions (C2)–(C5), they are discarded.

(C1) $\tau_{ri,1} \in \tau_{sj,1} + [e_1^- - e_3, e_1^+ + e_2 + e_3]$ and $\tau_{ri,2} \in \tau_{sj,2} + [e_1^- - e_3, e_1^+ + e_2 + e_3]$. This condition says the first and last packet times of receiver flow i are compatible with first and last packet times of sender flow j, given the uncertainties of propagation, queuing, and clock synchronization.

(C2) $n_{ri} = n_{sj}$ and $b_{ri} = b_{sj}$. Equality of bytes and packets is a necessary condition for the flows to be reporting on the same set of packets.

(C3) $\tau_{ri,2} - e_1^- + e_3 < \tau_{sj,1} + T_{act}$. The upper bound for latest possible compatible sending time of the last flow packet seen at the receiver $\tau_{ri,2} - e_1^- + e_3$, should fall within the active timeout since the first sender packet.

(C4) $\tau_{ri,2} - e_1^- + e_3 < \tau_{sj,2} + T_{in}$. This condition is similar to (C3) except that it rules out sender packets being excluded from the flow due to inactive timeout since the last sender packet.

(C5) If a pair of sender and receiver flows (of a particular key) has been discarded, discard all subsequent sender and receiver flows (of the same key) until the separation between successive flows exceeds T_{in} on both sender and receiver sides. This is used because after discard of a pair of flows for violating any of conditions (C2)–(C4), the first times of subsequent flows may refer to different packet on the sender and receiver sides. The condition ensures discard of such "out of sync" flows.

The strictness of timing checks relative to packet labeling depends on the timing parameters $e_1^{\pm}, e_2, e_3$. In Section IV-B, using timing parameters matched to our experimental setup, we validate the efficacy of timing checks by comparing with packet labeling, and quantify the relative number of discarded flows. However, for implementational simplicity, the evaluation of estimation accuracy in Section IV uses only packet labeling.

E. Foundations of delay correlation

Once pairs of flow records are associated, we have sender and receiver timestamps from both its first and last packets (but for just one packet in the case of a single packet flow). We discuss the key foundation of our approach to estimate per-flow latency characteristics of the particular segment using these packet timestamps.

The central premise behind our approach is that when two packets traverse a link closely separated in time, then the queuing delays that they experience are positively correlated. We then draw the conclusion that it makes sense to estimate the delay of a packet that we cannot measure directly, from the delays of closely separated packets that we can measure directly.

At an intuitive level, packets that experience the same queuing busy periods should tend to have positively correlated delays. There are a number of analytical results that confirm this behavior for classes of Markovian queuing models, such M/G/1 or G/M/1. For our purposes, there are informative results for models of this nature. In the simplest cases, the lag n autocorrelation of the packet queuing delay, γ_n, obeys

$$\gamma_n = 1 - n(1-\rho)^2 + O(1-\rho)^3 \tag{1}$$

where $\rho \leq 1$ is the offered load; see [34]. This formula shows that under heavy traffic conditions (ρ close to 1), noticeable correlations persist at lag n for n up to $1/(1-\rho)^2$, *e.g.*, up to $n = 100$ for a 90% load.

Although this result applies to a very simple model, there are two reasons to suggest that correlations in packet delays at a queue whose input is typical Internet traffic will be even greater. First, under more general conditions than (1) was derived, γ_n is a convex function of n, and hence the correlations in practice decay more slowly than the dominant linear behavior in n of (1). Second, Internet traffic exhibits burstier arrivals than the simple models [35]. So, for a given load, packets that queue are more likely to do so behind other queuing packets, and consequently waiting time autocorrelation will be greater than in the Markovian case.

The power of this idea for delay measurement is as follows. While NetFlow records do not report timestamps for each packet in a flow, they do provide them for the first and last packets. Thus, a stream of NetFlow records in turn provides a stream of packet-level timestamps, which, when compared at either end of a link, provides a time series of packet delays. Appealing to the correlation property, we can enhance the packet delay estimates of a particular flow, with directly measured delays from nearby packets from potentially other flows, in particular, arriving between the first and last packets of the flow under study.

F. Latency estimation

We consider several estimators in increasing degree of accuracy by exploiting delay correlation property discussed in Section III-E. We first discuss a simple estimator followed by our proposed approach.

Endpoint Estimator. The Endpoint method is the naive estimator that averages the pair $\Delta_{EP} = \{d_1, d_2\}$ of the delays of the first and last packets of the flow:

$$D_{EP} = \frac{d_1 + d_2}{2}$$

Multiflow Estimator. Although the Endpoint estimator is exact for flows having two packets, it is expected to have limited accuracy in general precisely because it is based only on two samples. We propose the Multiflow estimator based on

the autocorrelation description above. We first construct the matching streams τ_{si} and τ_{ri}, $i = 1, 2, \ldots$ of matching sender and receiver timestamps of all first and last packets of sampled flows. The Multiflow estimator does not distinguish between first and last packet timestamps. Now consider a particular flow under study with sender timestamps $\tau_{si,1}$ and $\tau_{si,2}$ and receiver timestamps $\tau_{ri,1}$ and $\tau_{ri,2}$ where $i, 1$ and $i, 2$ are the indices of the first and last packets of the flow. We form a delay estimate by averaging the delay over all packets with sender timestamps between the sender timestamps of the flow. Specifically, for $r_1 < r_2$ let $S(r_1, r_2) = \{i : r_1 \leq \tau_{si} \leq r_2\}$. The set of packet delays associated with the flow is

$$\Delta_{MF} = \{\tau_{ri} - \tau_{si} : \ i \in S(\tau_{si,1}, \tau_{si,2})\}$$

and the Multiflow estimator is

$$D_{MF} = \frac{\sum_{\delta \in \Delta_{MF}} \delta}{|\Delta_{MF}|}$$

where δ is a delay sample in the set Δ_{MF}.

Hybrid estimator. The hybrid method attempts to blend the best of the Endpoint and Multiflow methods. In our experiments in Section IV-C we shall find that the Endpoint method tends to be more accurate for smaller flows; indeed in the special case of a two packet flow without loss, it is exact. In the hybrid approach we apply a threshold in the number of packets per flow, using the Endpoint method for smaller flows and the Multiflow method for larger flows.

G. Variance and its Estimation

In measuring delay, it is useful to produce not just a single summary statistic, the mean, but also a measure of the variability of the distribution of individual delays about that value. Thus, we ask two questions: (i) given the expected correlations between delay measurements, how well is the variability of a single measurement predicted? (ii) how does the variability of set of samples Δ relate to that of the packets in the flow under study itself?

To answer the first question, if the delay samples $\delta \in \Delta$ were i.i.d random variables, we would form the unbiased estimate of the population variance as

$$\sigma^2 = \frac{\sum_{\delta \in \Delta} (\delta - D)^2}{|\Delta| - 1}$$

where $D = (|\Delta|)^{-1} \sum_{\delta \in D} \delta$ is the sample mean. In the presence of positive correlations between the δ, namely $\mathsf{E}[\delta\delta'] > \mathsf{E}[\delta]\mathsf{E}[\delta']$ for δ, δ' distinct elements of Δ, then $\mathsf{E}[\sigma^2] > \mathsf{Var}(\delta)$. Thus we would tend to overestimate the variance, which can be regarded as conservative for performance applications. We will answer the second question through experiments in Section IV. For the moment we make the observation that for the Endpoint estimator, the quantity σ^2 is expected to be extremely noisy, being based on only two data points; this is indeed the case reported in Section IV.

H. Interpolation of Packet Delays

Given our motivating intuition concerning correlation of packet delays, the correlation between the delay of arbitrary packet in a flow, and the endpoint delay of another flow should be strongest if we could choose the endpoint to have closest possible sender time to the packet in question. Unfortunately, we cannot construct an estimator this way, since we do not know the sender times of individual packets. On the other hand, for a performance study, choosing to estimate with a close-by packet indicates the limits of accuracy of any method, such as our Multiflow estimator, which uses delays of other flows as estimates.

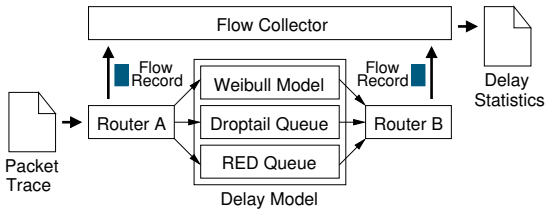


Fig. 2. Simulation environment.

To this end, we use linear interpolation between endpoint delays in order to construct a delay value for any time, then compare it with individual packet delays (not just endpoints) of arbitrary flows. Let $d_i = \tau_{ri} - \tau_{si}, \ i = 1, 2, \ldots, n$ denote the set of flow endpoint packet delays, ordered with increasing send time τ_{si}. For any time x between τ_1 and τ_m, let $i_+(r)$ and $i_-(r)$ label the closest delay values in the future and past:

$$i_+(x) = \min\{j : \tau_j > x\} \text{ and } i_-(x) = \max\{j : \tau_j < x\}$$

Then the interpolated delay at an arbitrary time x between τ_1 and τ_m is

$$d_{\text{int}}(x) = d_{i_-(x)} + (x - \tau_{si_-(x)}) \frac{d_{i_+(x)} - d_{i_-(x)}}{\tau_{si_+(x)} - \tau_{si_-(x)}}$$

For a lossless flow of m packets, having sets of $\{\tau_{si} : i = 1, \ldots, m\}$ and $\{\tau_{ri} : i = 1, \ldots, m\}$ of packet send and receive times, we construct the set of interpolated delay values for interior packets ($S_{\text{in}} = \{d_{\text{int}}(\tau_{si}) : \ i = 2, \ldots, m-1\}$), and actual endpoint delays ($S_{\text{end}} = \{\tau_{r1} - \tau_{s1}, \tau_{rm} - \tau_{sm}\}$):

$$\Delta_{\text{int}} = S_{\text{end}} \cup S_{\text{in}}$$

which we compare with the actual packet delay values $\Delta_0 = \{\tau_{ri} - \tau_{si} : \ i = 1, \ldots, m\}$.

IV. Evaluation

In this section, we evaluate the efficacy of our Multiflow estimator in obtaining accurate per-flow statistics using two different datasets—real backbone packet header traces with different synthetic queueing models and real router traces with synthetic workloads. We also consider the impact of several variables such as packet loss rate, packet sampling rate, etc. on the accuracy of our estimates. Before we describe the results, we first outline our experimental setup.

A. Experimental Setup

There are three main components in our experimental setup—packet header trace, delay model, and flow collection tool as shown in Figure 2. A full experimental realization

Link: OC-3, Duration: 305 secs.

Queue	Trace	#Flows	#Packets	Loss rate	Data rate
Droptail	WISC-D1	0.214M	3.99M	0.00%	124.98 Mbps
	WISC-D2	0.236M	4.38M	0.20%	137.67 Mbps
	WISC-D3	0.256M	4.70M	7.08%	146.58 Mbps
RED	WISC-R1	0.214M	3.98M	0.01%	124.58 Mbps
	WISC-R2	0.236M	4.38M	0.12%	137.22 Mbps
	WISC-R3	0.257M	4.69M	4.59%	146.19 Mbps

TABLE I
STATISTICS OF REAL ROUTER TRACES WITH SYNTHETIC WORKLOADS.

of CNF architecture would require correlated packet traces with per-packet timestamps taken at two different routers. Although some previous work has looked at packet latencies across routers (*e.g.* Papagiannaki *et al.* [8]), no traces meeting these requirements are publicly available, as far as we are aware. For this reason we adopted an alternative two-pronged approach. Firstly, we used a general-purpose packet header trace obtained from a single monitoring point, and applied simulated different delay distributions to it. Secondly, we used a trace from the authors of [36] arising from delay measurement of synthetically generated traffic traversing a *real* router. In each case, the flow collector allows us to take packet header traces and form flow records just as NetFlow would. Each of the components are explained next in detail.

Traces. We use two different datasets. Our first dataset is two real backbone packet header traces. Both traces contain no payload information, and all IP addresses are anonymized. The CHIC trace is collected by a monitor located at Equinix in Chicago, IL, contains traffic for 60 seconds from 5:00 PM, April 30th, 2008 on an OC-192 backbone link published by CAIDA [37]. This trace has about 1 million flows and 13 million packets in 60 seconds. The IPLS trace published by NLANR [38] contains traffic between Indianapolis and Kansas City through an OC-48 link (2.5 Gbps) collected on August 14th, 2002 at 9:20AM in the Abilene network. There are approximately 0.8 million flows and 17 million packets in this trace.

Our second dataset is collected by the authors of [36] in their evaluation of a new active probing tool. The Harpoon traffic generator [39] produces synthetic self-similar traffic over a dumbbell topology with an OC-3 bottleneck link using heavy-tailed sources and controls the offered load (*e.g.*, packet losses and delays) on the link with different Harpoon configurations. Even though the traffic sources are synthetic, packet stream generated by them experiences *real* router forwarding paths, queueing and other behavior, and thus are quite realistic. Ingress and egress packets are separately collected to two different traces. The router employed either Droptail or RED queueing policy during collection of these traces. The dataset consists of six different traces, with basic statistics summarized in Table I. Together, we call them WISC traces.

Flow collection tool. We used an open-source NetFlow platform called YAF [40] for our evaluation. We modified YAF to support hash-based packet sampling and flow sampling. In addition, we also extended YAF to report the hash label for the first and last packets of a flow for comparing the efficacy of our timing-based checks for packet association.

The tool also needs to perform packet association for finding individual packet delay when real router traces with synthetic workloads are fed into the tool. The idea of the packet association is basically borrowed from Section III-D. From five conditions in the section, we only used (C1) condition along with packet header fields (*e.g.*, IPs, port numbers, sequence number, and so on). While the condition is only applied to first and last packets of a flow, in this case we applied the condition for every packet because we basically need to know delays of all packets for the evaluation. Through the association for each and every packet, we successfully obtained delays of all packets except a few unassociated packets and lost packets for WISC traces.

Delay models. As shown in Figure 2, we have implemented three different delay models. The first is a Weibull distribution that Papagiannaki *et al.* have empirically found to model packet delays in real backbone routers [8]. The other two models are based on simulating queueing dynamics according to the Droptail and RED queue management [41] strategies. Note that these synthetic queueing delay models are only applied for CHIC and IPLS traces, not WISC traces. For the Weibull distribution, we set the scale and shape parameters to 0.0001 and 0.6 (as recommended in [8]) for both CHIC and IPLS traces. We model packet losses as a uniform distribution for this delay model. For the queueing models, we control the packet delays by configuring queue length and drain rate. We fix the drain rate (or equivalently per-byte processing time defined as the reciprocal of drain rate) in terms of bytes per second. By fixing the one parameter, drain rate, Droptail and RED automatically control both the delay as well as the loss distribution.

While Droptail requires no further parameters, RED needs configuring the queue weight (w_q), minimum and maximum thresholds (min_{th} and max_{th}), and maximum drop probability (max_p). Following the guidelines in [41], we chose a queue size of 10,000, $min_{th} = 4,000$ and $max_{th} = 9,000$ for the CHIC trace. For IPLS trace, we chose a queue size of 3,000, $min_{th} = 1,000$ and $max_{th} = 2,500$. We use $w_q = 0.002$ and $max_p = \frac{1}{50}$ for both traces.

B. Associating flow records

We first compare the efficacy of the timing-based approach by running the constraint checks (outlined in Section III-D) over the CHIC trace. We set $e_1^- = 10ms$ and $e_1^+ = 20ms$. We set e_2 by multiplying maximum queue size by average packet processing time (which varies from $40ms$ to $47.6ms$ respectively). We set the clock uncertainty parameter e_3 to $1ms$ to ensure worst case. We set $T_{in} = 10s$ and $T_{act} = 30s$. Although the default active timeout in NetFlow is 30 minutes, we use this lower value because the trace is only 60 seconds long.

Table II shows the proportion of flows surviving after filtering using the timing checks described in Section III-D. The results are cumulative, i.e., the column C3 shows the survival rate after checks C1, C2 and C3 have been applied. The last column represents the percentage of flows obtained using label-based consistency checks (described in Section III-D).

(Pkt. sampling rate =0.005, Total flows = 40,564)

Pkt. Loss	C1	C2	C3	C4	Pkt. Label
0.00%	100.00%	100.00%	99.99%	99.99%	100.00%
0.98%	98.77%	98.44%	98.43%	98.43%	99.24%
2.11%	97.21%	96.57%	96.56%	96.56%	98.20%
2.91%	96.24%	95.38%	95.37%	95.37%	97.64%
3.92%	94.88%	93.85%	93.84%	93.84%	96.65%
4.93%	93.73%	92.42%	92.41%	92.41%	95.85%

TABLE II
SURVIVAL RATES AFTER TIMING CHECKS (CUMULATIVE) AND USING PACKET LABELS.

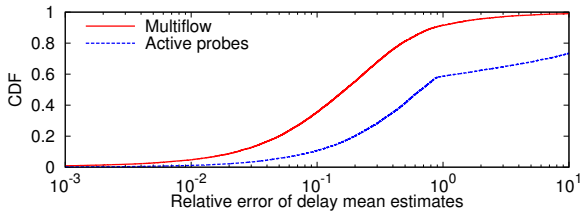


Fig. 3. Comparison with active probes. Droptail delay model on CHIC trace is applied. Packet sampling rate = 0.01.

For each flow, we include the flow if either label of the first or last packets matches between the flows at the sender and receiver. If both fail to match, which can happen for example when both packets are lost, the flow is discarded. Thus, at no packet loss, the packet label approach includes all flows, while at about 0.98% loss, we lose approximately 0.76% of flows because the packet labels do not match. The table also indicates that C3 and C4 result in obtaining the same set of flows since there were no instances of C4 in the trace. We found in all cases, that the set of flows obtained by applying the constraints are always a proper subset of those obtained using the packet labels. In summary, we can observe that just applying C1 alone, we already filter most of the inconsistencies out. The other checks result in eliminating only a small number of flows.

In our subsequent experiments, we use the packet label approach to flow record association, primarily due to its simplicity. As mentioned in Section III-D, label collision is a potential source of error in this approach. Now collision between temporally separate labels is manifested as an abnormally large delay value. Hence, to mitigate the effect of label collision, we remove large outlier delay values; in our experiments we remove the top 1% of delays.

C. Estimator accuracy

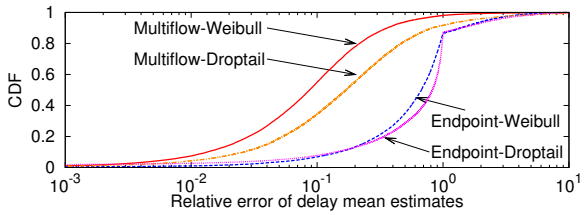
As an initial baseline, we compare the Multiflow estimator against an average delay derived from active packet probing. We follow this with an analysis of the errors in delay estimation for our proposed and existing passive measurement methods. We simulate both packet as well as flow sampling approaches. For simplicity, we also assume no packet loss for these cases; we discuss packet loss variation later in Section IV-D.

Comparison with active probes. We conduct 1-in-n active probing, which entails injecting a probe packet for every n background packets, and computing the average latency over a measurement interval, 60 seconds in our case. To conduct fair comparison with Multiflow estimator, the total number of active probes is roughly set to the total number of delay samples that Multiflow estimator uses. Since per-flow measurements are not explicitly provided by active probes, the aggregate latency is assigned to average latencies of all sampled flows. We believe that this is a right way to use active probes since there is no guideline about how to use active probes for per-flow latencies. We also conducted experiments with different active probe injection rates, but there was not much difference because the ensemble average is not very sensitive to the injection rate.

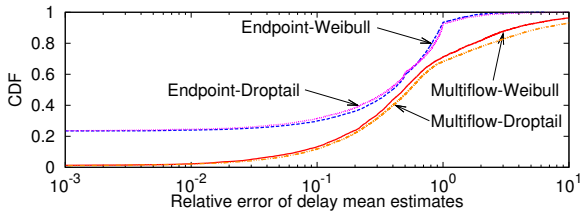
Figure 3 shows the comparison results between Multiflow estimator and active probes. As we expected, a single aggregate latency value cannot reflect flow-level latency measurements correctly, thus yielding high relative errors. For instance, about 41% flows have higher than 100% relative errors in case of active probes while Multiflow estimator has only 8% flows with those high errors. From a different angle, median relative error of Multiflow is about 17% but that of active probe is approximately 67%, which is four times higher than Multiflow's error. Deeper investigation revealed that while average latency measured by the active probing is about $3.5ms$, per-flow latencies spanned from $125\mu s$ at 10 percentile, to $3ms$ at 50 percentile, to $8.4ms$ at 90 percentile. Diverse flow durations (0.53, 20, and 51 seconds at 10, 50, and 90 percentiles, respectively) also cause an increase in the error of active probing for per-flow latency measurements. Therefore, the comparison results illustrate that our Multiflow estimator can keep track of variability in per-flow latencies better than active probes. We also observed same trend in case of flow sampling, and thus, omit showing the picture for brevity.

Accuracy depending on delay models and sampling methods. In Figure 4(a), we plot the CDF of the relative error of mean delay estimates for the Multiflow and Endpoint estimators. We show the curves for both Weibull distribution as well as Droptail queueing model (RED is exactly the same since there is no packet loss). We can observe that the Multiflow method obtains a median relative error of 10% and an 80th percentile relative error of about 20% for the Weibull delay model. The accuracy is slightly lower for the Droptail queueing model. The Endpoint estimator however is significantly inaccurate (a median error of 50%) compared to the Multiflow estimator for both distributions.

Further, we compare the accuracy of Multiflow with that of Endpoint using WISC traces. Packet sampling probability is set to 1%. We investigate estimation accuracy of all sampled flows as shown in Figure 5. Note that unlike experiments using CHIC trace, since we have no control of setting parameters like packet loss over WISC traces, the experiments were performed in the presence of packet losses. In the figure, the same estimators show similar accuracy for different queueing mechanisms because RED and Droptail have roughly the same packet loss rate. While Multiflow estimator achieves about a median relative error of 9%, Endpoint estimator achieves about a median relative error of 28% under both queueing models, which depicts that Multiflow is three times more accurate than



(a) Packet sampling



(b) Flow sampling

Fig. 4. Accuracy of estimators using Weibull distribution model on CHIC trace for flow and packet sampling. Flow and packet sampling rate = 0.01.

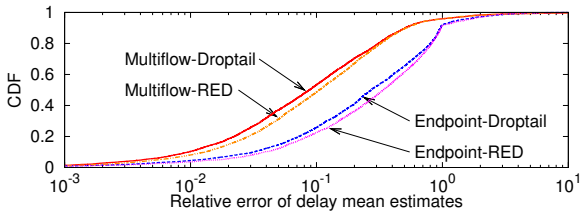
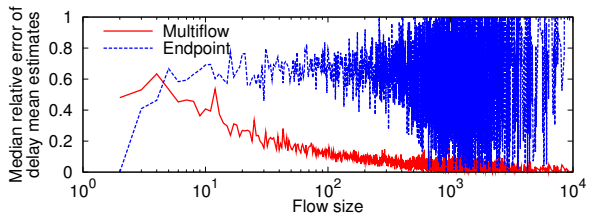
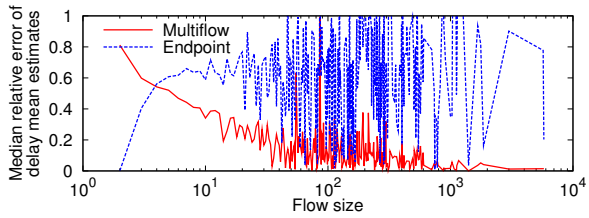


Fig. 5. CDFs of relative error of delay mean estimates using WISC-D2 and -R2 traces. Packet sampling rate = 0.01.



(a) Packet sampling



(b) Flow sampling

Fig. 6. Median relative error of delay mean estimates depending on flow size. Weibull distribution is applied to CHIC trace.

Endpoint. Both Multiflow and Endpoint estimators perform slightly better in estimation accuracy using WISC trace than CHIC trace. The difference is because true per-flow latencies in WISC trace is roughly an order of magnitude higher than those in CHIC trace and large denominator (*i.e.*, true per-flow latency) can yield small relative error.

Figure 4(b) shows a similar comparison between Endpoint and Multiflow for flow sampling. Curiously, Endpoint appears to perform better than Multiflow. We observe the similar trend in WISC traces, but omit the exact picture for brevity. The fundamental difference between flow sampling and packet sampling is the fact that flow sampling uniformly samples flows while packet sampling is known to be biased towards heavy-hitters, as large flows get sampled more often. So, in order to understand this deeper, we study the relationship between the estimator error and flow sizes (number of packets in a flow).

We plot in Figure 6 the variation of median relative error of mean delay estimates for different flow sizes. In other words, each point represents the median relative error among all flows that are of a particular size as defined by the x-axis. From the figures, we can clearly observe that the relative error of the Multiflow estimator decreases as the flow size increases for both flow- as well as packet-sampling. For flows of size greater than 200, the median relative error is less than 10% for packet sampling. In contrast, the error suffered by the Endpoint estimator increases as flow size increases significantly. More importantly, it becomes erratic indicating that it is not a reliable predictor of flow latency estimates.

For both types of sampling, somewhat curiously, we can observe from Figures 6(a) and 6(b) that the Endpoint performs better than the Multiflow estimator for flows up to a size of 3-4 (or so) and then the accuracy decreases. While the figures are for Weibull distribution, we have observed similar patterns for RED and Droptail. Further, similar trends in the experiments using WISC traces have been observed as well. We believe this is because typically for small flows, an estimate from two actual delay samples is better than approximating using all the intermediate packets. Thus, a hybrid estimator that combines the two approaches will provide strictly better accuracy than either of the two as we have discussed in Section III-F.

Now, we answer the question about why the Endpoint estimator appeared better than the Multiflow estimator in Figure 4(b). For smaller flow sizes (< 4 packets), the Endpoint estimator was more accurate than Multiflow. The fact that flow sampling contains a large number of small flows, meant that the overall distribution was heavily skewed towards these small flows. For flow sampling, the number of small flows is much larger than that using packet sampling (11,342 out of 11,721 for flow sampling as compared to only 5,436 small flows out of 16,178 for packet sampling). Thus, the distribution is dominated by the small flows for flow sampling.

Accuracy with respect to flow duration. We explore if there exists the correlation between flow duration and accuracy of our estimators. To understand that, we group flows at the unit of second and find median relative error of each group. Figure 7 shows the analysis results.

First, from the figure we observe that Multiflow estimator achieves much smaller median relative error than Endpoint across all the flow duration, and even the errors by Multiflow estimator become smaller as flow duration increases. This is mainly because packet sampling samples elephant flows more and the elephant flows tend to last long. Second, while we omit exact picture for flow sampling, we observed that median

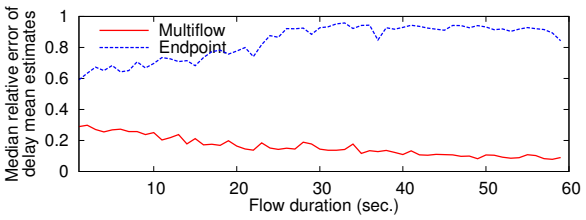
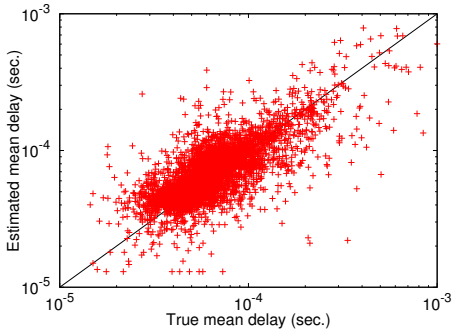
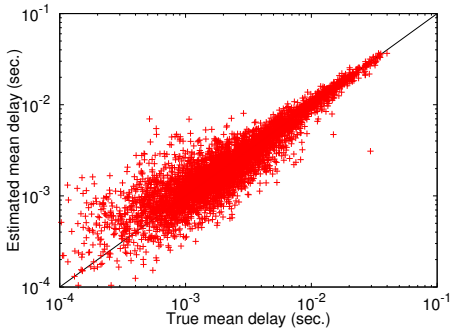


Fig. 7. Median relative error of delay mean estimates depending on flow duration using Droptail queueing model and packet sampling.



(a) IPLS



(b) WISC-D1

Fig. 8. Scatter plot showing the true as well as estimated delay spread across all flows.

relative errors by Multiflow estimator across almost all flow duration groups is higher than 50%. This is because in case of flow sampling there is no strong correlation between flow size and flow duration. Thus, the accuracy of Multiflow estimator is more related to the size of a flow, not the duration of the flow. These observations conform the observations made above.

Unbiasedness and delay-spread. To illustrate the unbiased nature of our Multiflow estimator, we show two scatter plots in Figure 8; one from IPLS trace and the other from WISC-D1 trace. The plots show the true mean delay on the x-axis and the estimated mean delay on the y-axis for each and every flow. Two main conclusions can be drawn from the plots: First, for IPLS trace, when all the flows are considered, there is quite a bit of spread in the true latency characteristics of the flows ranging all the way from 10^{-5} to 10^{-3} seconds, although most of the delays lie between 2×10^{-5} and 10^{-4}. In the case of WISC-D1 trace, we also find that the true latency characteristics of the flows spans over two orders of magnitude. These, in some sense, motivate why we need estimators such as ours; if all average delays were the same, per-flow estimators would not have been required. Second, the Multiflow estimator appears to have two-sided errors from both scatter plots empirically indicating its unbiased nature although it is not strictly conclusive. This nature could be important to obtain accurate aggregate latency, if necessary.

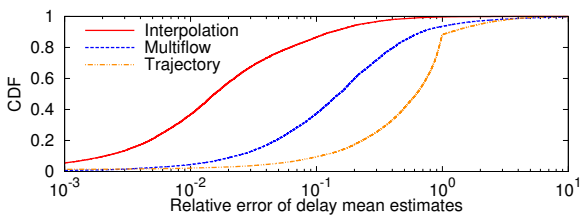


Fig. 9. Relative error for interpolated delay and Multiflow delay estimator for packet sampling. RED queueing model with no packet loss is used for CHIC trace.

Comparison with Interpolation and Trajectory sampling. In this experiment, we compare our Multiflow estimator with the *hypothetical* interpolation approach and a prior approach based on Trajectory sampling [7]. In Trajectory sampling, packets are consistently hash-sampled across routers, and reports on sampled packet include a (distinct) hash label with which to associate reports on the same packet. The association from label to flow is accomplished using an augmented report that includes both, that, to save reporting bandwidth, needs to be reported only once in the network, *e.g.*, at an ingress router. To obtain per-flow latency estimates in Trajectory sampling, we group together packet samples associated with the flow key and compute the packet average latency.

We observe from Figure 9 that the relative errors obtained using our Multiflow estimator are much higher than those obtained via interpolation for the CHIC trace. The results from the figure indicate that Multiflow has better accuracy than the Trajectory sampling approach, but there is still plenty of room for improvement. We could likely achieve the accuracy of the *unrealistic, hypothetical* interpolated delay estimator if information about packet inter-arrival can be obtained efficiently. Specifically, the horizontal distance between the curves is nearly an order of magnitude: an error of a given likelihood is about 8 times as large for Multiflow as for the hypothetical interpolation approach. The horizontal distance from Multiflow to Trajectory sampling is a little more than another half an order of magnitude, meaning an error of a given frequency is about 4 times larger for Trajectory sampling as for Multiflow. For example, the median relative error (0.5 on the vertical axis) is about 0.015 for Interpolation, 0.15 for Multiflow, but about 0.62 for Trajectory sampling. For the IPLS trace (omitted for the brevity), the curves are in the same order but tighter, with errors of a given likelihood being a factor of 2 greater for Multiflow as for Interpolation, with Trajectory sampling another factor of about 3 greater.

D. Sampling and loss rate variation

We have two variables that control the effective number of sampled packets—packet sampling rate and loss rate. Since the Endpoint estimator does not function as well as Multiflow, we

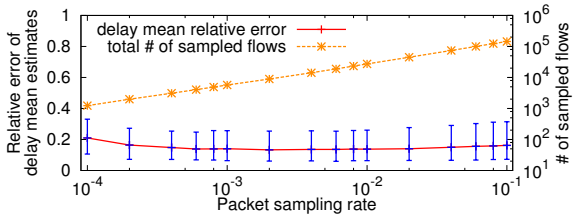


Fig. 10. Dependency of delay mean estimates on packet sampling rate. Each curve represents median with error bars indicating 25th and 75th percentile where flow size $>$ 100 packets for IPLS trace.

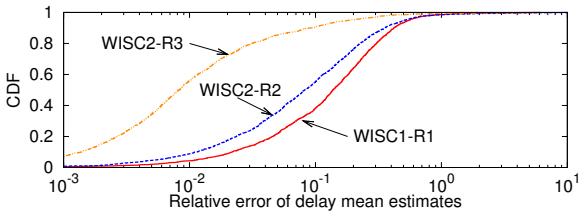


Fig. 12. CDFs of relative error of delay mean estimates by Multiflow estimator with different traces using WISC traces. Flow size $>$ 100.

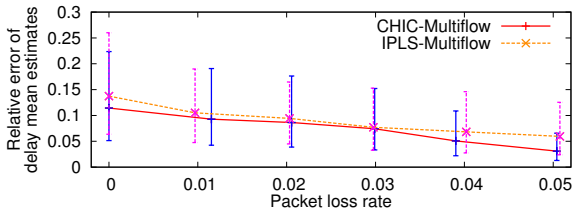


Fig. 11. Relationship between delay mean estimates and packet loss rate. The curves show median and the error bars indicate 25th and 75th percentiles. Flow size $>$ 100 packets. RED queue model is used.

only show the results for Multiflow. We also only consider packet sampling henceforth in the interest of space.

Impact of sampling rate. For understanding the relationship between estimation accuracy and packet sampling rate, we ran the experiments varying packet sampling rate from 0.0001 to 0.1 for both CHIC and IPLS traces using RED queue model. We only show the results of IPLS trace in this paper because the results of CHIC trace follow a similar trend of those of IPLS trace. We configured per-byte processing time to produce no packet loss in the experiments.

We show the influence of packet sampling rate on the estimator accuracy of large flows in Figure 10. We can observe that while relative errors decrease as the packet sampling rate increases, they stabilize after a sampling rate of about 10^{-3} for the IPLS trace and 10^{-2} for the CHIC trace (not included in the figure). While we do not show a similar graph for small flows (of size $\leq$ 100), we note that there is one major difference from the corresponding large flows' graph. We observe that as packet sampling rate increases, the relative errors for small flows also increase. We found this trend in both traces. While not as pronounced as packet sampling, we also observed this in flow sampling (omitted for brevity). We believe this phenomenon could be because of the fact that increasing packet sampling rate leads to an increase in delay samples. While we expect larger number of delay samples to be beneficial in general, the problem is that estimated delays now are aggregated over larger number of samples, which benefits larger flows but reduces smaller flows' accuracy. This observation is similar to that observed in Figure 4(b) where the Endpoint estimator was better than the Multiflow estimator.

Impact of packet loss rate. Higher loss rate typically reduces the effective number of samples from which we can compute the average per-flow latencies. We vary packet loss rate from 0% to 5% for both IPLS and CHIC traces by changing the drain rate of a queue. While 1% loss rate seems a reasonable maximum loss rate, we test up to 5% loss rate for understanding the impact of such a high packet loss rate. While we performed experiments for all three delay models, we mainly analyze results of RED while briefly summarizing other results. We show the RED results in Figure 11. Normally, one would expect that as loss rate increases, the error should increase as lesser number of samples exist. But, contrary to our intuition, in Figure 11, relative error reduces as we increase the loss rate. The explanation for this is as follows. In RED or Droptail queues, loss rates and delay characteristics are inter-twined; high loss rate implies that the queue is almost always full. Thus, the delay distribution is a lot more stable and hence easier to predict even with smaller number of samples (as a result of the loss rate). While our results for a Droptail queue model were similar to RED, the Weibull delay model shows relatively constant error in the latency estimates despite increase of packet loss rate. This is due to the fact that packet losses are independent with delays in our settings.

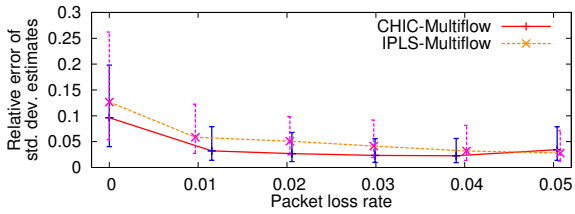
We also show the accuracies of per-flow mean latency estimates obtained by Multiflow estimator using real router traces with synthetic workloads–WISC traces. Since we do not have all traces showing various different packet loss rates, we only focus on three traces having small (0.01%), medium (0.12%), and high (4.59%) packet loss rates, which are named as WISC-R1, -R2 and -R3, respectively, and show CDFs of these traces with focus on flows having 100 or more packets.

Figure 12 illustrates the similar trend shown in Figure 11 that as packet loss rate increases, delay mean estimates get more accurate. Multiflow estimator achieves a median relative error of 0.83% by WISC-R3, 8.20% by WISC-R2, and 13.75% by WISC-R1, showing an order of magnitude in median relative error difference between WISC-R3 and others. From a different angle, 90% of flows in WISC-R3 have less than a relative error of 10%, 55% of flows in WISC-R2 show the trend, and those less relative errors are obtained by 38% of flows in case of WISC-R1, which again supports the observation from Figure 11.

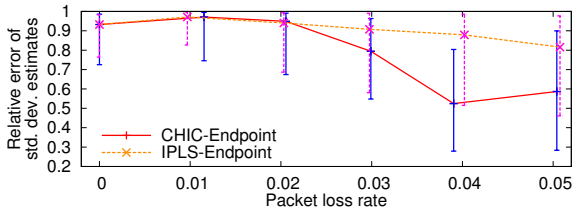
E. Accuracy of standard deviation estimates

We now explore the accuracy of the standard deviation estimator outlined in Section III-G. Just as before, we compare the accuracy of both the Multiflow and Endpoint estimators.

Similar to the case of mean delay estimates, the increase of the packet loss rate reduces the relative error of standard deviation of flow-level latency. In Figure 13(a), we observe that the relative error of Multiflow estimator for large flows



(a) Multiflow



(b) Endpoint

Fig. 13. Relationship between standard deviation of delay estimates and packet loss rate. The curves show median and the error bars indicate 25th and 75th percentiles. Flow size $>$ 100 packets. RED queue model is used.

decreases significantly. The same trend exists for both CHIC as well as IPLS traces. The Endpoint estimator on the other hand exhibits at least 50% error for all packet loss rates as shown in Figure 13(b). In both traces, while Endpoint estimator also shows the similar trend of Multiflow estimator such that as packet loss rate increases, relative error decreases, the median relative errors of Endpoint are over 80% for IPLS and 50% for CHIC. Even in the case of high packet loss rate (say, 4-5%), due to so much variation in estimation accuracy by Endpoint, standard deviation estimates by Endpoint cannot be trusted as well.

One curious observation is that, at around 4%-5% packet loss rate, the relative error of Multiflow on CHIC increases slightly. Upon careful investigation, we have observed that, as link utilization become higher (which happens when we increase the processing time and thus increase the loss rate), the variance of packet delay becomes very small, and the standard deviation of packet latencies of a flow becomes smaller compared to low link utilization case. Since relative error is prone to be more erroneous when the small value needs to be estimated, the relative error of the standard deviation becomes large. For instance, in CHIC trace, true and estimated delay standard deviations of a flow were about 0.016 and 0.015 at 2.11% packet loss rate, of which relative error was 6.25%. However, for the same flow, true and estimated delay standard deviations were about 0.006 and 0.007 at 4.93% packet loss rate leading to a relative error of about 17%.

We also studied the dependence on flow size of the relative error of the standard deviation for Multiflow. Just as before for average delay, as flow size increases, the relative error decreases significantly; for flows greater than size 100, the median error in standard deviation is less than 20%.

Again, under the study using real router traces with synthetic workloads, latency standard deviation becomes more accurate as packet loss rate increases as shown in Figure 14. However, the improvement in accuracy of standard deviation estimates

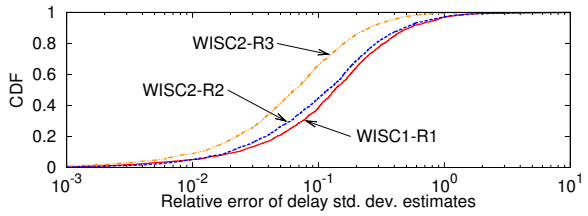


Fig. 14. CDFs of relative error of estimates by Multiflow estimator with different traces. Flow size $>$ 100.

among traces is less than that in mean estimation accuracy. Multiflow estimator obtained a relative error of 13.5% in WISC-R1, a relative error of 11.8% in WISC-R2, and a relative error of 6.5% in WISC-R3 at 50th percentile. We observed this trend in Droptail traces but omitted the graph for brevity.

V. Conclusion

Customers today are frustrated whenever their applications experience performance problems and demand better service from their ISPs. Network operators therefore need sophisticated tools to diagnose these problems whenever they do occur. Unfortunately, routers only provide coarse-grained SNMP counters and NetFlow which does not provide any latency measurements. Approaches that use active probes and tomography provide only aggregate statistics, not per-flow, and are also subject to inaccuracies due to the fundamentally under-constrained nature of the problem.

In this paper, we have proposed a way to retrofit per-flow latency estimates in the NetFlow. We started with the fundamental observation that NetFlow already records two timestamps on a per-flow basis. By harnessing hash-based sampling framework, which has been standardized by IETF, we proposed a Consistent NetFlow architecture that ensures that different routers record the same set of flows. From the different flow records collected from different router interfaces at a flow collector, we have shown the design of an opportunistic estimator that can utilize the background flow delay samples to estimate the per-flow average delay and standard deviation. Using two different datasets–real router traces with synthetic workloads and real backbone traces with synthetic queueing, we have shown that our Multiflow estimator can achieve significantly accurate estimates (about 20% median error for flows of size greater than 100 packets) of per-flow latencies under several realistic scenarios compared to prior approaches.

References

[1] "Sprint unveils SLAs for Internet access, Latency, Packet loss," http://www.crn.com/news/channel-programs/18827630/sprint-unveils-slas-for-internet-access-latency-packet-loss.htm;jsessionid=H9Xzdo9g+LhlF-2Sa89Big**.ecappj02.

[2] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (SNMP)," IETF, RFC 1157, May 1990.

[3] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," in *ACM SIGCOMM*, 2006.

[4] N. Duffield, "Simple network performance tomography," in *USENIX/ACM Internet Measurement Conference*, 2003.

[5] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An Algebraic Approach to Practical and Scalable Overlay Network Monitoring," in *ACM SIGCOMM*, 2004, pp. 55–66.

[6] J. Eidson and K. Lee, "IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems," in *Sensors for Industry Conference, 2002. 2nd ISA/IEEE*, 2002, pp. 98–105.

[7] N. G. Duffield and M. Grossglauser, "Trajectory sampling for direct traffic observation," in *IEEE/ACM Transactions on Networking*, 2000, pp. 280–292.

[8] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot, "Analysis of measured single-hop delay from an operational backbone network," *IEEE JSAC*, vol. 21, no. 6, Aug. 2003.

[9] "Cisco NetFlow," http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html.

[10] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a Better NetFlow," in *ACM SIGCOMM*, 2004, pp. 245–256.

[11] R. R. Kompella and C. Estan, "The power of slicing in internet flow measurement," in *ACM/USENIX IMC*, May 2005.

[12] C. Estan and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice," *ACM Transactions on Computer Systems*, vol. 21, pp. 270–313, 2003.

[13] L. Yuan, C.-N. Chuah, and P. Mohapatra, "ProgME: towards programmable network measurement," in *ACM SIGCOMM*, 2007, pp. 97–108.

[14] A. Ramachandran, S. Seetharaman, N. Feamster, and V. V. Vazirani, "Fast monitoring of traffic subpopulations," in *ACM/USENIX IMC*, 2008.

[15] A. Pasztor and D. Veitch, "A Precision Infrastructure for Active Probing," in *ACM Passive and Active Measurement Workshop*, 2001.

[16] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Accurate and efficient SLA compliance monitoring," in *ACM SIGCOMM*, 2007, pp. 109–120.

[17] J. Mahdavi, V. Paxson, A. Adams, and M. Mathis, "Creating a scalable architecture for internet measurement," in *Proceedings of INET'98*, 1998.

[18] "Test Traffic Measurement Service (TTM)," http://www.ripe.net/data-tools/stats/ttm/.

[19] "Cisco IOS IP SLAs," http://www.cisco.com/en/US/technologies/tk648/tk362/tk920/technologies_white_paper0900aecd8017f8c9.html.

[20] V. Paxson, "Measurements and analysis of end-to-end Internet dynamics," Ph.D. dissertation, University of California Berkeley, 1997.

[21] N. Hohn, D. Veitch, K. Papagiannaki, and C. Diot, "Bridging router performance and queuing theory," in *ACM SIGMETRICS*, 2004.

[22] B. Choi, S. Moon, Z.-L. Zhang, K. Papagiannaki, and C. Diot, "Analysis of point-to-point packet delay in an operational network," in *IEEE INFOCOM*, 2004.

[23] T. Zseby, S. Zander, and G. Carle, "Evaluation of building blocks for passive one-way-delay measurements," in *Proceedings of Passive and Active Measurement Workshop (PAM 2001)*, Amsterdam, The Netherlands, April 23-24 2001.

[24] N. Duffield, A. Gerber, and M. Grossglauser, "Trajectory engine: A backend for trajectory sampling," in *IEEE Network Operations and Management Symposium (NOMS) 2002*, Florence, Italy, April 15-19 2002.

[25] S. Machiraju and D. Veitch, "A measurement-friendly network (MFN) architecture," in *INM '06: Proceedings of the 2006 SIGCOMM workshop on Internet network management*, 2006, pp. 53–58.

[26] T. Qiu, J. Ni, H. Wang, N. Hua, Y. R. Yang, and J. J. Xu, "Packet Doppler: Network Monitoring using Packet Shift Detection," in *ACM CoNEXT*, 2008.

[27] R. R. Kompella, K. Levchenko, A. C. Snoeren, and G. Varghese, "Every MicroSecond Counts: Tracking Fine-grain Latencies Using Lossy Difference Aggregator," in *ACM SIGCOMM*, 2009.

[28] N. Duffield, C. Lund, and M. Thorup, "Charging from sampled network usage," in *IMW*, Nov. 2001.

[29] D. Veitch, S. Babu, and A. Pàsztor, "Robust synchronization of software clocks across the internet," in *ACM/USENIX IMC*, 2004, pp. 219–232.

[30] N. Duffield, B. Claise, D. Chiou, A. Greenberg, M. Grossglauser, and J. Rexford, "A framework for packet selection and reporting," RFC 5474, March 2009.

[31] T. Zseby, M. Molina, N. Duffield, and S. N. andF. Raspall, "Sampling and filtering techniques for ip packet selection," RFC 5475, March 2009.

[32] N. Hohn and D. Veitch, "Inverting sampled traffic," in *Internet Measurement Conference*, Oct. 2003.

[33] IETF, "Ip flow information export (ipfix) charter," http://www.ietf.org/html.charters/ipfix-charter.html, version of 16 December 2008.

[34] J. F. Reynolds, "The covariance structure of queues and related processes–a survey of recent work," *Adv. Appl. Prob.*, vol. 7, pp. 383–415, 1975.

[35] V. Paxson and S. Floyd, "Wide-area traffic: the failure of poisson modeling," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 257–268, 1994.

[36] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Improving accuracy in end-to-end packet loss measurement," in *ACM SIGCOMM*, 2005.

[37] C. Shannon, E. Aben, kc claffy, and D. E. Andersen, "CAIDA Anonymized 2008 Internet Traces Dataset (collection)," http://imdc.datcat.org/collection/1-06BX-2=CAIDA+Anonymized+2008+Internet+Traces+Dataset.

[38] "Abilene-I data set," http://pma.nlanr.net/Traces/Traces/long/ipls/1/IPLS-KSCY-20020814-092000-0.gz.

[39] J. Sommers and P. Barford, "Self-Configuring Network Traffic Generation," in *ACM/USENIX IMC*, 2004.

[40] "YAF: Yet Another Flowmeter," http://tools.netsa.cert.org/yaf/.

[41] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, 1993.

Myungjin Lee is a Ph.D. student of Department of Electrical and Computer Engineering at Purdue University. He previously worked as a research engineer for ETRI, a government-funded R&D institute in Korea specializing in telecommunications. Earlier, he received his M.S. degree from KAIST, Korea, in 2002 and B.E. from Kyungpook National University, Korea, in 2000. His main research interests are in computer networks, with a focus on scalable architectures and algorithms for network measurements and monitoring.

Nick Duffield (M'97, SM'01, F'05) received the Ph.D. degree from the University of London, London, U.K., in 1987. He is a Distinguished Member of Technical Staff and an AT&T Fellow in the Internet and Network Systems Research Center, AT&T Labs-Research, Florham Park, NJ, where he has been since 1995. He previously held post-doctoral and faculty positions in Dublin, Ireland, and Heidelberg, Germany. He is a co-inventor of the Smart Sampling technologies that lie at the heart of AT&Ts scalable Traffic Analysis Service. His current research focuses on measurement and inference of network traffic. Dr. Duffield was Charter Chair of the IETF working group on Packet Sampling. He is an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING.

Ramana Rao Kompella (M'07, ACM M'07) is currently an Assistant Professor in the Department of Computer Sciences at Purdue University. His main research interests include scalable algorithms and architectures for high speed switches and routers. He received his Ph.D degree from UCSD in 2007, M.S from Stanford University in 2001, and B.Tech degree from IIT Bombay in 1999. He received an NSF CAREER award in 2011.