One-way Loss Measurements From IPFIX Records

Fabio Ricciato
Univ. del Salento and FTW
Email: ricciato@ftw.at

Felix Strohmeier, Peter Dorfinger
Salzburg Research Forschungsgesellschaft
Email: felix.strohmeier@salzburgresearch.at

Angelo Coluccia
Univ. del Salento
Email: angelo.coluccia@unisalento.it

***Abstract*—In this work we describe a methodology to estimate one-way packet loss from IPFIX or NetFlow flow records collected at two monitoring points. The proposed method does not require tight synchronization between the two monitoring points, nor it relies upon external routing information. It can run online or offline, and can work on legacy IPFIX/NetFlow traces which were not collected for the specific purpose of loss estimation. In this preliminary work we describe the estimation procedure and present early validation results from a real testbed.**

I. Introduction

One-way packet loss between two network sections, A and B, is defined by the fraction of packets which travel from A towards B but are lost along the path, i.e., fail to be delivered to point B. Together with delay, one-way loss is a key quality indicator of the service offered by the network between the two reference points. Monitoring the actual level of packet loss is important for network operation and troubleshooting. Continuous online monitoring of packet loss rates can be used to promptly detect and report on sudden increase or slow drifts, pointing to network problems like congestion, route failure and others. In some cases, offline loss estimation is also of interest, e.g. for the "post-mortem" analysis of network traces, to investigate and drill down the cause of an abnormal event.

Many network operators nowadays have deployed and maintain legacy monitoring infrastructures collecting NetFlow [1] or IPFIX [2] flow records. It is therefore natural to ask whether and how this *legacy* infrastructure and/or pre-existing flow data — which may have been collected for other purposes than loss estimation — can be exploited to measure one-way loss. In this paper we propose a method to accomplish that.

Besides the operational utility of measuring the packet loss level in time — typically in time bins of fixed length, e.g. 1 minute — our work has a more curiosity-driven motivation: we aim at developing a method to measure one-way loss *on a per-flow basis*. Our long-term goal is to use it as a tool for studyng the correlations between the loss level and other structural characteristics of real Internet traffic, e.g. flow size and/or duration. To this end, we need to define "flows" in a way that allows accurate and non-ambiguous measurement of per-flow one-way loss. This requirement is not fulfilled by raw IPFIX/NetFlow "flow records", since there is no 1:1 association between records collected at different monitoring points. Therefore, we introduce new entities that we call *superflows* — to avoid confusion with the term "flows", used here to refer to IPFIX/NetFlow records — for which one-way packet loss can be unambiguously measured.

Related Work: Our method was directly inspired by the methodology proposed by Friedl *et al.* in [3]. Therein, the authors proposed a method for online loss estimation based on the notion of *expired flows* — the latter play a role that is logically similar to our *superflows*. The solution proposed in [3] requires the deployment of an *ad-hoc* measurement infrastructure with coordinated flow meters at the two monitoring points, while instead *we seek to develop an approach that can work with legacy flow data as produced by exisiting IPFIX/NetFlow meters*. More in details, the method in [3] assumes that the packets-to-flow mapping at the two monitoring points is tightly aligned, based only on expiration timeouts set exactly to the same value at both points. Instead, in the IPFIX/NetFlow scenario the two meters work independently. First, they might use different values for the active/inactive timeouts. Second, they can terminate flows based on protocol events (e.g. TCP FIN flag observed) or router cache flushing, and these can not be assumed aligned at the two monitoring points (consider e.g. that a TCP FIN packet might be lost across the path between the two points). Therefore, we aggregate the IPFIX/NetFlow flows into larger entities, superflows, designed to be aligned.

In another relevant work Gu *et al.* [4] address the problem of loss estimation in fixed time-bins from sampled IPFIX records through a single link. The focus is orthogonal to ours: while they study on the accuracy achievable in presence of packet sampling, assuming fully synchronized flow meters and a single-link path between the two monitoring points, we consider an orthogonal case without packet sampling, in a multi-link environment, without synchronization and possibly without external routing information. Furthermore, the approach in [4] does not deliver loss measurements per-(super)flow, that is one of our goals.

II. Reference Scenario and Assumptions

The input data to the process are two sets of flow records (IPFIX or NetFlow) collected at two monitoring points A and B, as sketched in Fig. 1. The goal is to measure the loss rate of packets in the path from A to B during a generic observation interval. We assume for simplicity that the datasets from the two monitoring point are gathered to a common repository, and therefore can be processed centrally.

The clocks of the two monitored sections are not assumed to be tightly synchronized. If the mis-alignement of the timestamps between the two datasets is large, a preliminary

978-1-4577-0457-4/11/$26.00 ©2011 IEEE

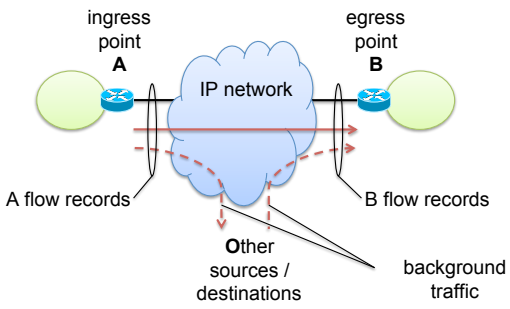

Fig. 1. Reference scenario.

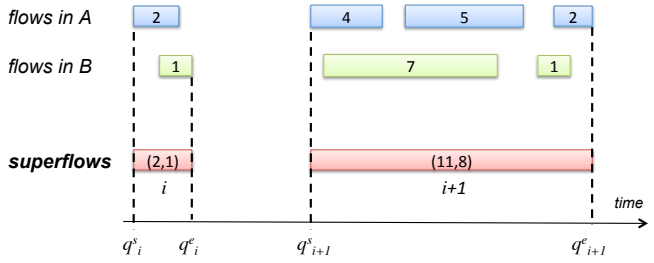

Fig. 2. Flows and superflows for a given 5-tuple. Each superflow i is labelled with the pair (n_i, m_i).

alignement procedure (see §III-C) can be adopted to reduce the timing error down to sub-second.

The flows traversing section A that are routed towards section B will be collectively referred as *foreground traffic*. All the other flows constitute the *background traffic*. The proposed method is designed to work also without external routing information, i.e. when no *a priori* knowledge is available about which of the flows observed in A are routed towards B. In this case, foreground flows are extracted directly from the dataset: only the flows that are observed at both monitoring points are taken as foreground. This approach however introduces an additional source of error related to *disappearing flows*: by this term we identify flows observed at the ingress point A with n packets and routed towards B — as such, they are legitimate foreground flows — but for which all n packets are lost before reaching point B. Therefore, no flow record is reported in B about such flows, and therefore they will not be classified as foreground in our method unless external routing information is provided. The implications of disappearing flows and a possible workaround are discussed in more details in §IV.

For the basic algorithm version considered in this article we assume that only *direction* information is available, at least for foreground flows: if a flow is seen in both datasets A and B, enough information is available to discriminate whether the flow traversed the path from A to B or viceversa. In most practical settings this assumption is not critical, i.e. when the measurement points are located at stub networks. Moreover, for TCP flows the direction information can be easily derived from the flag fields. As part of our future work we intend to develop refinements of the proposed method to cope with the case that direction information is not available.

A key assumption is that the input flow datasets are complete, not sampled. In principle, the proposed method can be applied without modifications to sampled data, but only in case of flow-based sampling (based on IP 5-tuple hashing) and with the condition that both meters use the same hashing scheme.

III. Flow Records Processing

A. Notation

We consider the subset of flow records matching a specific *key*, i.e. a particular value for the IP 5-tuple (src/dst addresses and ports plus protocol number), on both measurement points A (ingress) and B (egress). For the generic flow k in this set, we denote by l_k^A and l_k^B the number of packets observed at each monitoring point. We consider the following points in time, according to an ideal reference clock:

- $t_k^{s,A}$ and $t_k^{e,A}$ are respectively the start and end times of flow k observed at point A, i.e. the instant when the first and last packet traversed the ingress monitoring.
- $t_k^{s,B}$ and $t_k^{e,B}$ are respectively the start and end times of flow k observed at point B, i.e. the instant when the first and last packet traversed the egress monitoring.

Each of the above instants is associated to a timestamp, produced by the flow meter's clock at the relevant monitoring point and included in the flow data. We use the symbol 'p_x^y' to denote the timestamp corresponding to the *true* instant 't_x^y': for example, $p_k^{s,B}$ denotes the start timestamp recorded at the ingress monitor point corresponding to $t_k^{s,B}$.

We denote by Γ a deterministic upper bound to the one-way packet delay through the path $A \rightarrow B$. In practice, the value of Γ will be set to a conservative value, e.g. $\Gamma = 1$ sec (note that typical values for one-way delay through the internet rarely exceed 100-200 ms). We denote by Δ the maximum absolute value of the timestamp error, i.e. $|p_i - t_i| \leq \Delta$, $\forall i$. Our algorithms requires only a loose synchronization between the meter's clocks at the two monitoring points, e.g. via NTP. If the traces are severely mis-aligned, a preliminary timing realignement procedure can be applied to reduce the maximum clock error Δ down to a value comparable with the maximum one-way delay, as explained below. Based on that, we assume conservative value $\Delta = 1$ sec. This value is also in agreement with the typical clock error for NetFlow timestamp reported recently in the literature (see e.g. [5]).

B. Construction of superflows

At the heart of the proposed method there is a procedure to aggregate flow records that match the same key (i.e., 5-tuple) and are in temporal proximity into virtual entities called "superflows". Each superflow includes records *from both monitoring points* A and B. By construction, superflows with the same key must have a minimum temporal spacing $p_{k+1}^s - p_k^e \geq D$ with $D \stackrel{\text{def}}{=} 2\Delta + \Gamma$: this ensures that when a new superflow starts, all packets from the previous superflow (with same key) that are not lost along the path have been

received at the egress point[1]. In other words, no packets from the previous superflow can be found still "on the fly" along the path. A graphical representation of the mapping between flows and superflows is given in Fig. 2.

The algorithm that maps flows to superflows for a generic key (5-tuple) takes as input two sets of flow records matching the given key, collected at the ingress and egress monitoring points, A and B respectively. These two dataset are merged into a single collection of K flow records indexed in $k = 1, \ldots, K$. Before describing the algorithm we introduce some notation. For each input flow k, we denote by p_k^s and p_k^e respectively the start and end timestamps. Furthermore, we associate to flow k the vector $\boldsymbol{u}_k \stackrel{\text{def}}{=} (l_k^A, l_k^B)$ constructed as follows: if the flow record k is from the meter in A, the component l_k^A is set to the number of packets seen in A, while $l_k^B = 0$; conversely, if the flow record k is from the meter in B, the component l_k^B is set to the number of packets seen in B, while $l_k^A = 0$. We further assume that, for the given key, the flow records are *ordered by increasing start timestamp* p_k^s

The output is a set of *superflows* for the given key. Each superflow i is associated to a pair of integers, n_i and m_i, counting the number of packets observed respectively at the ingress and egress monitoring points, with $n_i \geq 1$ and $0 \leq m_i \leq n_i$. For a more compact notation, we introduce the vector $\boldsymbol{c}_i \stackrel{\text{def}}{=} (n_i, m_i)$. Each superflow i is also associated to start and end timestamps q_i^s and q_i^e. With this notation, the algorithm that maps flows to superflow can be described by the following pseudocode. Our implementation of the algorithm in `awk` required less than a dozen lines of code.

Algorithm III.1: FLOWS2SUPERFLOWS(D)

$k \leftarrow 1$; $i \leftarrow 1$
repeat
- $q_i^s = p_k^s$; $q_{tmp} = p_k^e$
- $\boldsymbol{c}_i \leftarrow \boldsymbol{u}_k$
- **while** $(p_{k+1}^s \leq q_{tmp} + D) \wedge (k \leq K)$
  - **do** $k = k + 1$
- $q_i^{tmp} \leftarrow \max(q_{tmp}, p_k^e)$
- $\boldsymbol{c}_i \leftarrow \boldsymbol{c}_i + \boldsymbol{u}_k$
- $q_i^e \leftarrow q_{tmp}$
- $i \leftarrow i + 1$

until $k = K$

C. Preliminary Timing Alignement

This preliminary procedure can be applied when the two input datasets were produced by flow meters that were not online synchronized, not even with a basic protocol like NTP. The idea is to identify a set of "landmark" flow records, pick their start timestamps from each monitoring point, and estimate the offset and skew error components via linear regression — an approach conceptually similar to the offline synchronization method first proposed in [6] (see also the discussion in [7]). Landmark flows must be selected among those that did not experience any packet loss along the path, i.e. *with the same number of packets* at both ingress and egress monitoring points ($l_k^A = l_k^B$). This ensures without ambiguity that the start timestamps at both monitoring points refer to the transit of the same (initial) packet, which in turns guarantees that a strict ordering relationship exist between the (unknown) transit times. Loss-free landmark flows will be more likely found among flows of small size, particularly single-packet flows. For the generic landmark flow j we denote by p_j^A and p_j^B the *start timestamps* at the two monitoring points. Without loss of generality, we take the clock t^A at the ingress point A as the reference. The absolute clock error ϵ at point B (relative to A) is commonly modeled by two main components: fixed offset u and skew v, therefore $\epsilon \stackrel{\text{def}}{=} t^B - t^A = u + v \cdot t^A$. When considering the timestamps, one must consider also the truncation error, typically in the order of 1 ms for IPFIX data — for NetFlow data, the truncation error within a single trace can be reduced to below 100 ms, see [5]. The skew error can be estimated by the slope of the regression line across the set of datapoints $\{p_j^A, p_j^B\}$. After estimating the skew error, the timestamps of *all* flow records in B (not only the landmarks) are adjusted to compensate for it. Finally, in order to compensate for the initial offset, a constant shift is added so as to impose that the minimum difference between the timestamps at the two monitoring points is exactly zero.

This procedure leaves a residual error comparable with the (unknown) minimum one-way delay between the two interfaces — which was absorbed into the clock offset compensation — plus the truncation error. Altogether, we assume that the absolute residual timing error between the two datasets does not exceed $\Delta = 1$ sec.

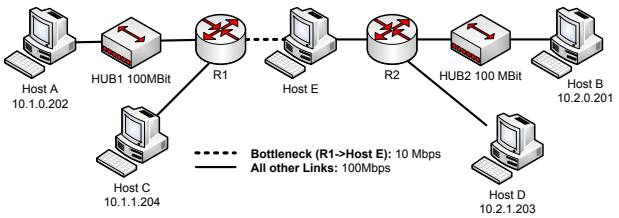

Fig. 3. Testbed topology.

IV. Loss Metrics

We denote by $F_{n,m}$ (with m, n integers such that $n \geq 1$ and $0 \leq m \leq n$) the number of foreground superflows with n packets observed in A (ingress) and m packets observed in B (egress). The value of n will be referred to as the *size* of the flow. Superflows with $m = 0$ are observed only on A, and are called *disappearing superflows*. We further denote by $N \stackrel{\text{def}}{=} \sum_{n \geq 1} \sum_{m=0}^{n} n F_{n,m}$ the total number of foreground

[1]The condition $p_{k+1}^s - p_k^e \geq 2\Delta + \Gamma$ is sufficient to ensure that the spacing between the "true" start/end instants is larger than the maximum one-way delay, i.e. $t_{k+1}^s - t_k^e \geq \Gamma$, under the assumption that the absolute error on each timestamp is bounded by Δ, i.e. $|p_k - t_k| \leq \Delta$.

packets seen in A (which are routed towards B) and by $M \stackrel{\text{def}}{=} \sum_{n\geq 1}\sum_{m=0}^{n} mF_{n,m}$ the total number of foreground packets seen in B. We also define $S \stackrel{\text{def}}{=} \sum_{n\geq 1} F_{n,0}$ the total number of packets belonging to disappearing superflows.

The goal is to measure the total fraction L of packets lost in the path $A \rightarrow B$, i.e. the *Empirical Loss Ratio* defined as:

$$L_0 \stackrel{\text{def}}{=} 1 - \frac{M}{N} = 1 - \frac{\sum_{n\geq 1}\sum_{m=0}^{n} mF_{n,m}}{\sum_{n\geq 1}\sum_{m=0}^{n} nF_{n,m}}. \quad (1)$$

Note that L includes counts of disappearing superflows. As such, it can be measured exactly only if one is able to classify disappearing flows into foreground traffic, e.g. from external routing information. If routing information is not available, only the superflows observed in both A and B are classified as foreground, therefore the disappearing flows (which are not observed in B due to the loss of all packets) can not be distinguished from background traffic. This introduces an additional source of error, as part of the foreground traffic is left out of the computation of the loss ratio. Without countermeasures, we would then measure the following metric:

$$L_{dis} = 1 - \frac{\sum_{n\geq 1}\sum_{m=1}^{n} mF_{n,m}}{\sum_{n\geq 1}\sum_{m=1}^{n} nF_{n,m}} = 1 - \frac{M-S}{N-S}. \quad (2)$$

It can be easily seen that, due to the presence of disappearing flows, this metric systematically *underestimates* the actual loss level, i.e. $M \leq N$ and $S \geq 0$ implies that $L_{dis} < L_0$. The simplest way to mitigate the problem is to ignore small superflows with size below a fixed threshold, i.e. $n \leq \phi$: in fact, in practice only flows of very small size, and especially single-packet flows, have non-negligible probability of being completely lost (disappearing). Therefore, filtering out the small superflows implies the removal of most (if not all) disappearing flows from the computation.

This leads to the following class of loss metrics, with threshold parameter ϕ:

$$L_\phi = 1 - \frac{\sum_{n\geq \phi+1}\sum_{m=1}^{n} (n-m)F_{n,m}}{\sum_{n\geq \phi+1}\sum_{m=1}^{n} nF_{n,m}}. \quad (3)$$

This approach was adopted in [3], wherein single-packet flows where discarded ($\phi = 1$).

Given the above framework, it is easy to define loss metrics for specific classes of superflows. For instance, the mean loss probability experienced by superflows of the size in the interval $n_1 \leq n \leq n_2$ is defined simply by

$$L(n_1, n_2) = 1 - \frac{\sum_{n=n_1}^{n_2}\sum_{m=1}^{n} (n-m)F_{n,m}}{\sum_{n=n_1}^{n_2}\sum_{m=1}^{n} nF_{n,m}}. \quad (4)$$

Comments on small-superflow filtering. A side advantage of ignoring small superflows for the computation of the loss rate is that the number of elements to be handled by the estimation algorithms is reduced. Moreover, filtering away very small superflows automatically eliminates most of the so-called "unwanted traffic", i.e. traffic originated by unproductive (and often illegitimate) activities like scanning and flooding — see e.g. [8], [9] and references therein. Such traffic might experience a loss rate considerably higher than the rest of legitimate traffic: for example, in a previous study of traffic from a real network [9] we found that high-rate packet bursts originated by sequential scanning were causing micro-congestion events, i.e. very short periods (subsecond) of high loss. This caused roughly 80% of packets involved in the scanning burst to be lost at some bottleneck link, but the impact on the remaining (legitimate) traffic was negligible (since the duration of scanning bursts was very short). In this case, averaging together in a single metric the loss rates of the two traffic components, unwanted and legitimate traffic, would give a distorted view of the packet loss experienced by the latter. Since in practice most of unwanted traffic is mapped to very small flows, often made of a single packet, filtering very small superflows would implicitly filter out the impact of such traffic in the measurement.

Time binning and superflow weighting. In practical settings, the loss metrics is computed in time bins of e.g. 1 minute, so as to obtain timeseries that can be inspected or processed to identify short-term anomalies and/or long-term drifts. If a superflow is not entirely contained within one timebin, we weight its contribution to each timebin proportionally to the share of its total duration falling in that bin. This is equivalent to assume a constant packet (and loss) rate throughout the whole superflow lifetime. A similar approach was applied in [4] to weight IPFIX flows. In our testbed (introduced below), the maximum superflow duration is around 100 seconds, therefore any individual superflow can span two, maximum three timebins. In this condition, superflow weighting works very well. The option of filtering out very long superflows spanning many timebins might be considered when applying our method to real datasets.

V. Testbed validation

A. Testbed Setting

The proposed estimation procedure was validated with flow data gathered in a testbed. Testbed experiments allow to approximate real traffic conditions, while retaining full control over the test environment and full knowledge of the reference "ground truth". The testbed topology is depicted in Figure 3: all involved hosts run Linux, while the routers are from Cisco. The central Host E runs a link emulator tool, NetEM [10], configured to impose a random delay in the range $[45, 55]$ msec to the packets flowing in each direction, resulting in an average RTT of 100 msec.

The flow metering processes on Hosts A and B created the IPFIX flow records by using the YAF [11] tool. In order to proof the robusteness of our scheme in heterogeneous scenario, the flow termination timeouts where set to different values at the two meters: the idle / active timeouts were set respectively to 10 / 180 seconds in A, and 5 / 60 seconds in B.

In parallel, a tool built on JPCAP [12] was used to collect packet dump from the same interfaces, from which the "ground truth" was derived. The two hosts had been synchronized via NTP until a few hours before the experiments. When

running the Timing Alignement procedure described in §III-C we estimated a positive clock skew of $3.4 \cdot 10^{-7}$ (1.2 ms/hour): we did not compensate for it, since for the duration of the experiment (2 hours) such a small value is not influential.

The test network was loaded with a rich traffic mix, blending TCP and UDP connections representative of different applications. To this end, we had to use two traffic generators in parallel: Harpoon [13] for web traffic, D-ITG [14] for voip, streaming and bulk transfers. Moreover, we used nmap [15] to produce high-rate scanning traffic, as representative of the unwanted traffic that is found in the real Internet (see e.g. [9] and references therein): bursts of TCP SYN directed towards different destinations are produced approximately every 10 minutes. Every burst has a duration of 5-6 seconds, and a gross bitrate of 2.5 Mbps and 0.8 Mbps respectively in the high and low load scenarios.

We used the MINER software platform [16] to manage the configuration, coordinated execution and data collection for all tools involved in the experiments across the various hosts. Both, flow- and packet-capturing processes make use of the pcap library (libpcap). Note that packets can be lost during the capturing process when buffers are exceeded. To validate that no packets are missed neither by the flow meter nor by the packet dumping process, we verified for each experiment that the number of packets "dropped by kernel" as reported by libpcap is zero. The superflow construction agorithm described in §III with $D = 3$ seconds was implemented in the `awk` language and applied on the IPFIX flow record files produced by the YAF meters.

B. Testbed Results

We run experiments in two different scenarios with similar traffic mix but different load, tuned to produce different loss levels: 2.5% for Scenario 1 and 30% for Scenario 2. The loss was measured in time bins of 1 minute during 2 hours. The choice of relatively high loss rates — considerably higher than the "normal" packet loss rates expected in healthy real networks — is motivated by the need to verify that the proposed method works well also in extreme adverse conditions: the higher the loss, the larger the difference between the packet-level patterns observed at the ingress and egress points, hence between the two flow-level datasets.

In Fig. 4 we report the temporal profile (in timebins of 1 minute) of the "true" loss measured from the packet dump, along with the loss measured from the superflows, with and without routing information. For the latter, we show the curves obtained with and without filtering of single-packet superflow, i.e. $\phi = 1$ and $\phi = 0$ respectively.

First, we note that if routing information is available — therefore all disappearing flows are correctly taken into account in the computation — the superflows method follows always very tightly the reference measurements from the packet dumps. This is further confirmed by the scatter-plots on the left side of Fig. 5.

Second, Fig. 4 confirms the need for filtering out short flows (at least those made of a single packet) when no routing

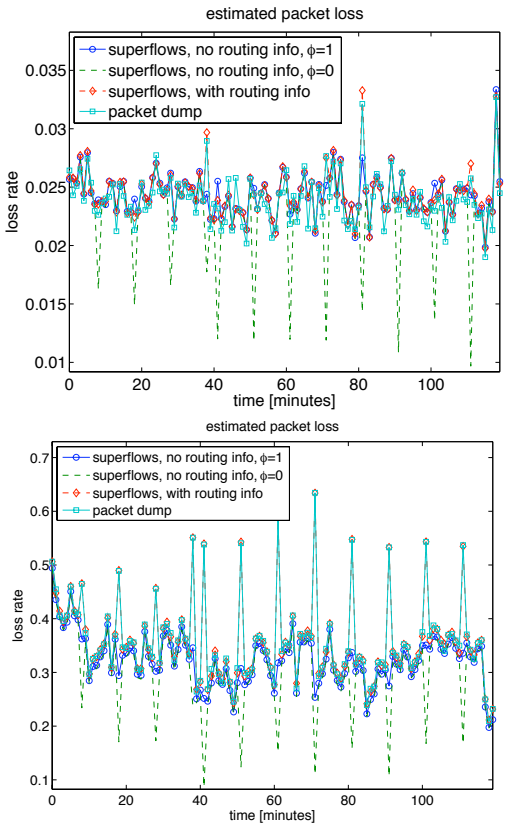

Fig. 4. Loss rate in timebins of 1 min. Upper: Scenario 1, mean loss 2.5%. Lower: Scenario 2, mean loss 30%.

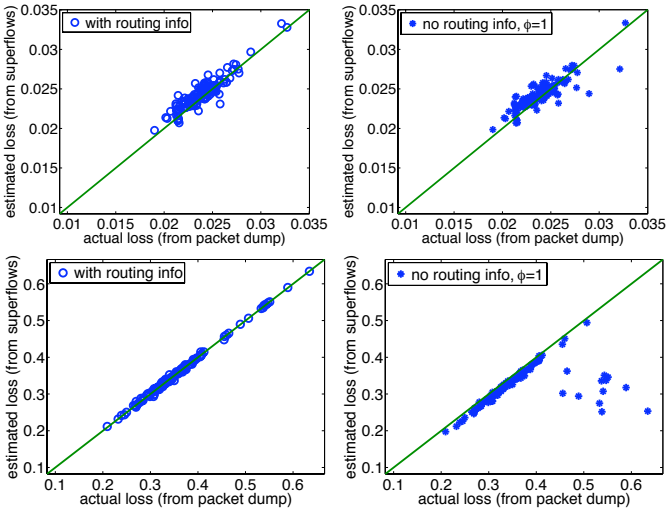

Fig. 5. Loss measured by superflow vs. "ground truth" from packet dump. Top row: Scenario 1, average loss 2.5%. Bottom row: Scenario 2, average loss 30% (35% including scanning traffic).

information is available to the superflows method. In fact, the curve for $\phi = 0$ yields downwards notches corresponding to the timebins containing scanning traffic. Recall that scanning packets are mapped 1:1 into superflows — as they typically target different port and/or IP address — therefore in these

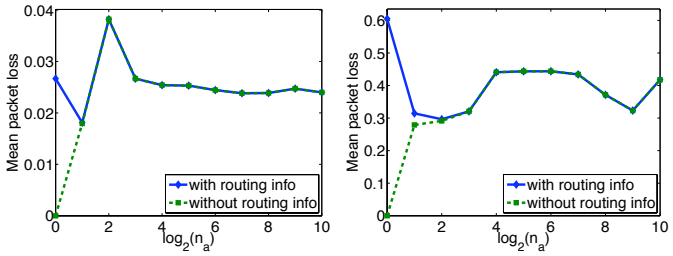

Fig. 6. Average packet loss vs. superflow size (logarithmically binned) in Scenario 1 (left) and Scenario 2 (right).

timebins there is an excess of single-packet flows. Recall from the discussion about (2) that very small superflows induce a negative bias on the loss estimation when no routing information is available, as lost packets are implicitly excluded from the computation. The magnitude of such error depends on the relative frequency of single-packet flows, therefore it should come to no surprise that when scanning bursts are present, hence many single-packet flows are produced, the under-estimation error increases considerably over the remaining timebins, leading to the evident notches. It can be seen that filtering for single-packet flows eliminates completely the problem: the curves for $\phi = 1$ do not yield any notch.

In the high-load scenario the intensity of the scanning bursts (between 2 and 2.5 Mbps) is such to cause scanning packets to be lost with much higher probability than the remaining traffic. This produces "spikes" of higher loss in the total packet loss measured by the packet dump (see bottom graph of Fig. 4). Such spikes are captured by the superflows method with routing information, as all superflows are taken into account in the computation of the loss ratio. Instead, when no routing information is available, the filtering of single-packet superflows ($\phi = 1$) causes the loss estimation to miss these spikes. This is also evident in the scatter-plots on the right side of Fig. 5. As discussed earlier, this might not be an undesirable effect: by discarding very short (super)flows we end up in measuring the loss level "seen" by legitimate traffic, implicitly filtering out the impact of unwanted traffic.

The proposed method can be used to study correlations between loss and other superflow attributes. To illustrate, we report in Fig. 6 the measured loss vs. superflow size. The superflows were grouped by the number of packets n seen at the ingress point A in logarithmically spaced bins $2^b \leq n < 2^{b+1}$, with $b = 0, 1, \ldots$ The average loss in each group (recall eq. (4)) is reported in Fig. 6 for both scenarios, with and without routing information. As discussed above, the loss for single-packet superflows ($b = 0$) can be measured only if routing information is available. Besides that, for all other groups the availability of routing information is almost irrelevant, since the incidence of disappearing flows with multiple packets is very small also in the high-loss Scenario 2. We also notice that in Scenario 2 the loss experienced by single-packet (super)flows is 60%, well above the one experienced by superflows of larger size. In general, differences in the loss level between different classes of superflows can be explained by noting that applications — in our testbed as well as in the real Internet — generate traffic flows (hence, superflows) of different typical size and different temporal patterns (burstiness).

VI. Conclusions and Ongoing Work

In this work we have taken a first step towards developing a tool to measure one-way loss on a per-(super)flow basis from legacy IPFIX/NetFlow data. Our method can serve to investigate the phenomenon of packet loss in the real Internet more in-depth, behind the mere global average, as it allows to dis-aggregate loss measurements for different classes of (super)flows and applications.

On the methodological side, the basic procedure proposed in this article can be refined and evolved along several directions. For one, we are set to develop variants of the basic scheme that do not require directional information. Another important point of further study relates to the impact of sampling on the accuracy achievable by the proposed method.

References

[1] G. Sadasivan, V. Valluri, M. Djernaes, and J. Quittek, "Cisco systems netflow services export version 9," in *RFC 3954*, October 2004.

[2] G. Sadasivan, N. Brownlee, N. Claise, and J. Quittek, "Architecture for ip flow information export," in *RFC 5470*, March 2009.

[3] A. Friedl, S. Ubik, A. Kapravelos, M. Polychronakis, and E. Markatos, "Realistic passive packet loss measurement for high-speed networks," in *2nd Traffic Monitoring and Analysis Workshop (TMA'09)*, 2009.

[4] Y. Gu, L. Breslau, N. Duffield, and S. Sen, "On passive one-way loss measurements using sampled flow statistics," in *IEEE INFOCOM 2009*, Rio de Janeiro, Brasil, April 2009.

[5] B. Trammel, B. Tellenbach, D. Schatzmann, and M. Burkhart, "Peeling away timing error in netflow data," in *PAM 2011*, 2011.

[6] A. Duda, G. Harrus, Y. Haddad, and G. Bernard, "Estimating global time in distributed systems," in *Proc. 7th Int. Conf. on Distributed Computing Systems*, vol. 18, 1987.

[7] B. Poirier, R. Roy, and M. Dagenais, "Accurate offline synchronization of distributed traces using kernel-level events," *SIGOPS Oper. Syst. Rev.*, vol. 44, 2010.

[8] F. Ricciato, "Unwanted traffic in 3G networks," *Computer Communication Review*, vol. 36, no. 2, April 2006.

[9] F. Ricciato, E. Hasenleithner, and P. Romirer-Maierhofer, "Traffic analysis at short time-scales: an empirical case study from a 3g cellular network," *IEEE Transactions on Network and Service Management*, vol. 31, no. 8, pp. 1484–1496, March 2008.

[10] S. Hemminger, "Network Emulation with NetEm," in *Linux Conf Au*, Apr. 2005.

[11] C. M. Inacio and B. Trammell, "Yaf: yet another flowmeter," in *Proc. of LISA'10*, 2010.

[12] K. Fujii, "Jpcap v0.7: a java library for capturing and sending network packets," 2007. [Online]. Available: http://netresearch.ics.uci.edu/kfujii/Jpcap/doc/index.html

[13] J. Sommers, H. Kim, and P. Barford, "Harpoon: a flow-level traffic generator for router and network tests," in *SIGMETRICS '04*, 2004.

[14] A. Botta, A. Dainotti, and A. Pescape, "Multi-protocol and multi-platform traffic generation and measurement," in *INFOCOM 2007 DEMO Session*, Anchorage, Alaska, May 2007.

[15] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure, 2009. [Online]. Available: http://nmap.org/book/

[16] C. Brandauer and T. Fichtel, "Miner - a measurement infrastructure for network research," in *Proc. of TRIDENTCOM'09*, 2009.