HAL open science

Inline Data Integrity Signals for Passive Measurement

Brian Trammell, David Gugelmann, Nevil Brownlee

To cite this version:

Brian Trammell, David Gugelmann, Nevil Brownlee. Inline Data Integrity Signals for Passive Measurement. 6th International Workshop on Traffic Monitoring and Analysis (TMA), Apr 2014, London, United Kingdom. pp.15-25, 10.1007/978-3-642-54999-1_2 . hal-01396468

HAL Id: hal-01396468

https://hal.science/hal-01396468v1

Submitted on 14 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Inline Data Integrity Signals for Passive Measurement

Brian Trammell[1], David Gugelmann[1], and Nevil Brownlee[2]

[1] Communication Systems Group, ETH Zurich, Switzerland
[2] Dept. of Computer Science, University of Auckland, New Zealand

Abstract. In passive network measurement, the quality of an observed traffic stream is obviously crucial to the quality of the results. Some sources of error (e.g., packet loss at a capture device) are well understood, others less so. In this work, we describe the inline data integrity measurement provided by the QoF TCP-aware flow meter. By instrumenting the data structures QoF uses for detecting lost and retransmitted TCP segments, we can provide an in-band, per-flow estimate of observation loss: segments which were received by the recipient but not observed by the flow meter. We evaluate this mechanism against controlled, induced error, and apply it to two data sets used in previous work.

1 Introduction and Related Work

Network measurement is the practice of examining traffic data in order to deduce or derive information about the properties of the measured network. This traffic can either be generated for the purposes of measurement (in active measurement), or can be generated by the network's users (in passive measurement). In all cases, the measurement infrastructure itself injects some error or bias into the measurement, such that interpretation of the results must take care to account for this error. This fact is too often ignored in the network measurement literature.

Measurement studies can use control traffic with known parameters to provide ground truth, whether part of the protocol for active measurements, or injected alongside the measured traffic for passive measurements. In all cases, the data produced are a function of the actual traffic measured, the mismatch between the assumed and actual behavior of the network, and inaccuracies injected by design or implementation flaws in measurement tools.

Even the simplest of active measurement approaches, the venerable `ping` utility, is not immune from this conflict between the ideal and the real. For example, Pelsser et al [1] recently found that two-way delay measured across a link by `ping` can have significant dependencies on the flow identifier, an artifact of in-path load balancing.

It is harder to control for these errors in passive measurement than active measurement, and the mismatch between real and assumed behavior of the measurement tools plays a greater role. For example, Hofstede et al [2] characterized

inaccuracy in timing and TCP flags in NetFlow data in three commercial flow meters. Cunha et al [3] detailed timing errors in publicly available Abilene and GEANT flow traces injected by Juniper's J-Flow. An author of the present work [4] even found timing error inherent in the design of Cisco's Netflow Version 9 protocol. Error injected by passive measurement design, implementation, and deployment issues was treated systematically by Kögel [5], who advocated exporter profiling to correct for these errors post-measurement.

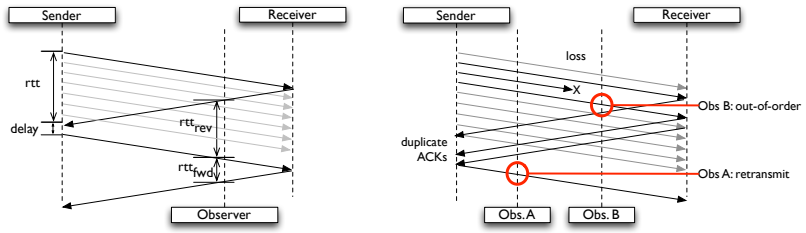
In this paper, we consider the QoF (pronounced "quaff") TCP-aware flow meter, which leverages the flexibility of the IPFIX [6] protocol to export per-flow statistics relevant to TCP performance. QoF was developed to allow per-flow observation of TCP flows, for operational troubleshooting as well as for research purposes. As such, it requires precise timing for the estimation of TCP round-trip time (RTT) as well as very low or no *observation loss* – packets which were delivered, but not observed. Because QoF is a purely passive flow meter, a self-validation approach using injected traffic is not an option. Therefore, we sought to observe signals present in the data in order to increase our confidence about the suitability of the data for the measurement techniques employed by QoF.

We instrumented QoF's data structures for signaling detected gaps in the input data in order to report on observation loss. The literature on the effects of packet sampling treats the problem of known random observation loss on the fidelity of flow data [7, 8]; a key contribution of the present work, in essence, is the export of information about the *unknown*, not necessarily random sampling attributable to a measurement infrastructure, and export of this information along with measured data, so that it can be used in subsequent analysis. While this information is often available in the logs of flow meters and analysis tools, QoF's approach of providing information on loss *in-band* greatly increases its usefulness.

In this paper, we briefly introduce QoF and the techniques it uses for passive TCP performance measurement, as background for understanding the data integrity checking approach it uses, in section 2. We then detail the methodology for observation loss and timestamp frequency variance measurement in section 3, and apply it to several data sets and network observation points in 5. We note that the tool is particularly applicable to the verification of packet and flow data quality in research applications.

2 Background

QoF is a fork of the CERT Network Situational Awareness Group's YAF (Yet Another Flowmeter) [9]. Our fork of the code removes all payload inspection and export facilities from YAF to increase performance and reduce end-user privacy risk, and replaces the packet capture code with WAND's libtrace [10] to add flexibility and accelerate capture with general-purpose network interfaces. The overall goal is to produce a flow meter operating on unsampled packet header observations which is useful for both performance research and operational purposes. Performance is an explicit design goal, so we avoid techniques



(a) Correction of RTT (b) Indications of transport loss

Fig. 1. Effects of observer placement on TCP observations

which marginally improve accuracy at the cost of significant analysis or state requirements at the metering process.

Additional functionality in QoF includes passive RTT and transport loss event estimation. As round-trip time (RTT) is a basic parameter of all TCP connections, passive measurement of RTT is a fairly well studied area. QoF uses an approach based on previous work in the area [11, 12].

As in figure 1(a), RTT is estimated by adding the interval between seeing a sequence number and the corresponding ACK RTT_{fwd} to the interval between seeing the sequence number in the reverse direction and the corresponding ACK RTT_{rev}. Since many flows are mainly unidirectional, QoF also uses intervals between a TCP timestamp value on a pure ACK and its corresponding echo as well. In contrast to the method described in [13], timestamp echoes are considered only if they would decrease or keep equal the estimate for the component on which they were measured; this simple heuristic serves to minimize *delay*, especially that due to an application not having data to send.

QoF estimates transport loss – packets that did not arrive at the receiver – by observing the sequence of TCP sequence numbers. QoF tracks the initial sequence number as well as the next-expected sequence number in each direction of a flow. Since gaps in the sequence number space should be relatively rare – occurring transiently and refilled within an RTT in most cases – the module tracks gaps in a *gap stack* per flow direction.

Each unseen range in sequence number space is kept in descending order of sequence number. When a segment is observed that begins beyond the next expected sequence number, an out-of-order segment is signaled and the resulting gap is pushed onto the gap stack. Conversely, when a segment is observed that does not advance the sequence number, it is compared with the gaps in the gap stack, most recent gap first. If covering a gap, the gap is filled; otherwise, a retransmission is signaled.

If a loss occurs in the path after the observation point (Observer A in figure 1(b)), the observer will see the retransmitted segment(s) up to one RTT after the loss occurred. If the loss occurs in the path before the observation point (Observer B), the observer will see a gap in sequence number space immediately after the

loss, but not observe a retransmission; the size of the gap in the sequence number space is an indication of how many segments were lost. QoF handles spurious retransmissions in part by grouping transport loss indications into RTT-sized windows.

These two features complement each other, and assume the observation of every packet in each flow in order to work properly: transport loss requires the whole sequence of sequence numbers, and RTT estimates on missing ACKs and timestamp echos would be erroneously long. We therefore sought indications in the measured data that this assumption holds.

3 Observation Loss Methodology

In contrast to transport loss, *observation loss* occurs when a packet arrives at the receiver but is not seen by the observation point. We use the gap stack to measure observation loss as well as transport loss. Gaps which are not filled before they are pushed off the stack by newer gaps, or that are still in the gap stack when a flow completes, represent portions of the sequence number space that we presume the TCP receiver saw, since progress through sequence number space continued, but which were not observed by QoF. QoF exports information about observation loss for each direction of each TCP flow.

Packet capture devices and libraries often make available information about how many packets were dropped – partially observed or inferred but not captured – on a per-interface basis. However, this is not the only type of observation loss. For example, packet capture based on packet replication at a switch will fail to observe packets the switch fails to forward down the span port; our method will count this as observation loss as well.

Though this measurement is admittedly rather obvious, the innovation of counting observation loss on a per-flow basis as opposed to a per-interface basis also allows certain analyses to proceed even in the face of unobserved packets, and to investigate any dependencies on packet properties in observation loss. Though we can only caluclate observation loss for TCP flows, we assume that most observation loss processes are transport protocol independent, and can therefore extrapolate observation loss at a given time on a given path for coincident non-TCP traffic.

We note that observation loss measurement using a gap stack is subject to overcounting in three specific conditions. First, the gap stack has a fixed size per flow, so particularly "frothy" flows – those with many reorderings – may have observation loss overcounted, if a gap falls off the end of the stack before it would have been filled. QoF's compiled-in default gapstack size is 8, chosen empirically through analysis a set of test traces. Overcounting due to froth can be reduced by increasing this, at the expense of memory efficiency.

Second, if a gap would be filled after the flow is expired by idle timeout or the observation of a FIN or RST segment, that gap will be counted as observation loss as well. We consider the performance of this structure to be worth the potential overcounting; future work involves additional tweaks to the algorithm to

minimize overcounting while maintaining performance. We explore this further in section 5.2.

Additionally, QoF's decoder drops any packet it cannot succeessfully decode, and is paranoid about what it accepts, a legacy of its heritage as a security monitoring tool. This can lead to observation loss caused by QoF itself. QoF also uses a fixed capture length per packet, allowing it to disregard octets it will never use, thereby improving its performance. However, packets where the layer 2, IP, and TCP headers together are longer than the compile-time default of 96 octets, or read from capture files with a shorter snaplen, will lead to observation loss due to such decode failures. This is possible especially in cases of excessive IPv6 encapsulation and/or extension headers. As with the gapstack size, observation loss on long packets can be reduced by increasing the compile-time constant, at the expense of I/O performance.

4 Evaluation

To evaluate the utility of per-flow loss counting to detect impaired measurement, we added detuning functionality to QoF to model packet loss and packet delay in the measurement path[3]. Loss and delay due to residence in a bottleneck queue are modeled using a leaky bucket with a specified size and drain rate. Additionally, we model non-queue-related loss with a test of a specified loss probability against a linearly smoothed uniform random die, and non-queue-related delay using a linearly smoothed uniform delay generator. Since QoF is built under the assumption that packet timestamps are monotonic strictly not decreasing, all delay imparted by the detune module is clamped to a minimum value in order to avoid reordering packets.

We then tested against a publicly available trace from the WIDE MAWI Working Group Traffic Archive[4], taken from a 150Mb/s transpacific link over a three hour period on 30 March 2012, containing 386.2 million packets. The peak flow concurrency [14] is 307.6 thousand concurrent flows (with 63s idle and 300s active timeouts). Reflecting the fact that the trace is taken from a backbone, 72.6% of observed flows are one-way due to asymmetric routing[5]. We assume, but cannot confirm, that actual observation loss in the MAWI traces is negligble.

This evaluation shows, unsurprisingly, that there is a strong relationship between the proportion of flows with reported observation loss and the induced packet drop rate. The function of this relationship is determined by the type of loss induced. For leaky bucket loss with induced packet loss rate varying from 0.00289 to 0.138 and a queue length of 100ms, this function is linear ($slope =$

[3] As these features are only useful in the context of this evaluation, they are only enabled in QoF when the `--enable-detune` flag is given to QoF's build configuration script.

[4] `http://mawi.wide.ad.jp/mawi/`

[5] While QoF is a biflow meter, integrity checking does not require observation of both sides of the flow to work properly, so this evaluation treats each uniflow in a biflow separately.

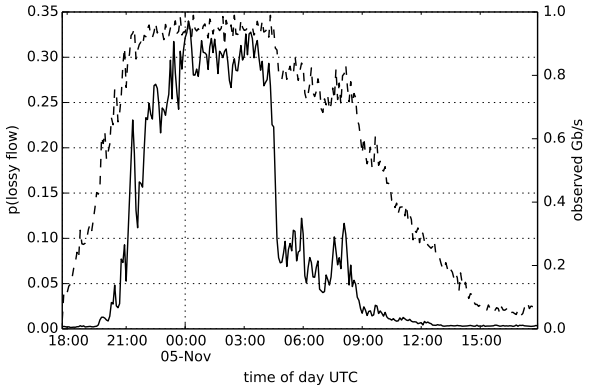


Fig. 2. Daily seasonality of flow loss (solid line) and observed data rate (dashed) on a typical day at Auckland

0.136, $intercept = 3.44 \times 10^{-4}$, $corr = 0.996$). This technique is sensitive even to very low packet loss rates; dropping only 246 packets from 386.2 million led to 70 flows being detected with observation loss.

The dominance of short flows in Internet traffic [14] means that packet loss will lead to unobserved flows, as well. In this evaluation, for each flow with measured observation loss, about four flows went unobserved.

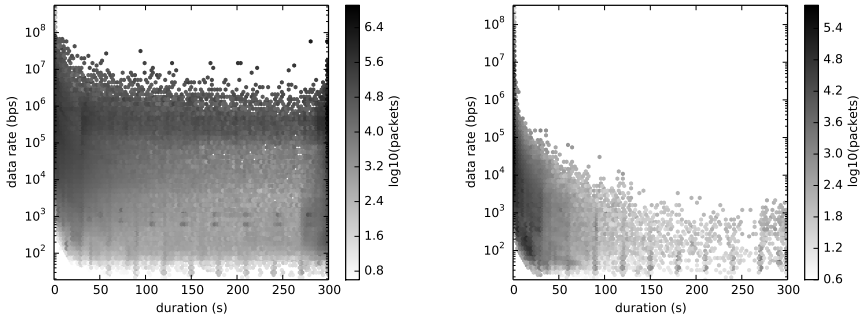
The queue modeled by leaky bucket loss also induces error in packet timings, which changes QoF's estimation of round trip times: the median of the minimum estimated RTT of flows in the MAWI data set is 101ms; this shifts to 110ms using an induced drop rate of 0.302 (i.e., on the order of the loss seen in the Auckland data set treated in section 5.1 below). Since loss due to queueing or buffering (e.g., at a span port or on a capture device) is the dominant type of loss we assume, we note that this indicates that observing loss in a data set also indicates that the timings should not be trusted in further analysis.

5 Findings

Here we apply QoF's per-flow loss reporting features to ongoing data collection efforts on various research networks which have been used in recent publications. The goal here is to verify that the data source used in these works has appropriate observation loss characteristics for the analysis done.

5.1 Auckland and iatmon

We first applied this methodology to an observation point at the University of Auckland, implemented using a span port on inbound and outbound traffic at the campus border. During the daytime traffic peak at about 11:00 local time (12:00 UTC), we routinely found between 25% and 35% of observed flows reporting



(a) Lossy (b) Lossless

Fig. 3. Packets by mean data rate and duration of flow during a peak hour at Auckland: note that observation loss affects long-duration, high-rate flows disproportionately

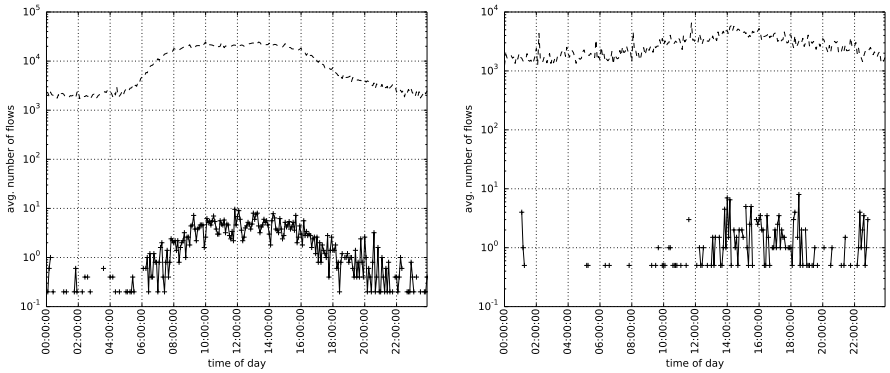
observation loss. On further investigation, we found that a span port originally configured to forward approximately 100 Mb/s of traffic in each direction over a Gigabit Ethernet interface was now being offered approximately 700 Mb/s of traffic in each direction at peak. Since 1400 Mb/s of traffic will not fit on a 1000 Mb/s link, the span port did not forward about a third of the packets to the observation point.

For the last two years the University's network operations group has been working towards upgrading its Internet-facing infrastructure, including the monitoring and measurement infrastructure. Unfortunately, this requires forklift upgrades to firewalls and boundary routers, and progress is slow. Taking a more positive view, we regard this as a chance to investigate the effects of high observation loss on traffic analysis.

In figure 2, we show the evolution of the dropped flow rate over a typical day (Tuesday, November 5, 2013). Flows experiencing observation loss during the peak hour have a significantly different shape from those which escape the span port lossless. Compare the subfigures in figure 3: while simply dropping the lossy flows in an analysis would lead to retaining about two-thirds of the flows, most of the long-duration, high-rate traffic, and therefore most of the packets, would be missing.

In [15], we monitored one-way traffic using `iatmon`, which measures interarrival time for packets from unsolicited trafic, and classifies such flows into ten groups based on their IAT distributions. That paper used data from the UCSD Network Telescope, i.e. header-trace data for only one-way flows.

However, we also run `iatmon` at Auckland using the same observation point as for our QoF work. At Auckland, `iatmon` filters out the one-way flows from the total traffic – one-way traffic accounts for about 3.5% of the observed packets but about 35% of the observed flows. `iatmon's` 'flow group' classification is fairly coarse-grained, relying only on the overall features of the IAT distributions. so



(a) Weekdays (5 min bins) (b) Weekend (5 min bins)

Fig. 4. Average daily seasonality of flows with detected observation loss (solid line) and total number of flows (dashed line) as observed by QoF.

it should not be affected by our high observation loss rate. However, one-way flows with only a few packets – say 10 or fewer – may not be observed, thereby distorting the distribution of flows between groups.

5.2 Horizon Extender

We then applied this methodology to HTTP traces captured at the border router of an university network. Similar to the Auckland setup, packets from the 10 Gb/s inbound and 10 Gb/s outbound link of the main router are forwarded via a single 10 Gb/s span port to the capturing device. A Myricom 10G network adapter in combination with tcpdump is used for traffic capturing. Packets are filtered before being written to disk. The network data analyzed in the following cover IPv4 TCP port 80 traffic of clients in a selected subnetwork of the university that have been recorded during 5 work days and one weekend in late 2012. The total IP-level raw size of the analyzed traces is 1.2 TB of incoming and 43 GB of outgoing traffic, which correspond to 15M TCP streams accounting for 1.2 TB[6] of incoming and 23 GB of outgoing TCP payload data.

The Myricom driver on the capturing device reported no packet loss due to ring-buffer overflow, but since the 10 Gb/s up- and downstream traffic are mirrored over a single 10 Gb/s span port, packets could have been lost at the router. Here we use QoF to show that this was most likely not the case, and compare results to logs from the Bro IDS [16].

The recorded traces have been used to evaluate *Horizon Extender* [17], an approach to archive HTTP data for data leakage investigations. Because it is impossible to fully reconstruct HTTP request and response streams from TCP

[6] Since HTTP is highly asymmetric, incoming traffic is dominated by large flows, so IP layer overhead is negligible.

streams with packet loss, it was important for the evaluation that packet traces of high quality were available, that is, with as little observation loss as possible.

Figure 4 shows the average number of flows and flows with detected observation loss as observed by QoF. The rate of such flows is on the order of 2×10^{-4} overall.

Analyzing the number of distinct clients and servers for which observation loss was observed in more detail, we found on a single weekday that while we observe on average 5 ± 6 lossy flows per 15 minute time bin, only 3 ± 3 clients and 4 ± 4 servers are affected per time bin. In other words, certain paths between clients and servers are more likely to be affected by observation loss.

We then compared this observation loss rate to the byte loss rate reported by the TCP stream reconstruction engine in Bro, which analyzes complete packet payloads, and assembles them into full streams, as opposed to analyzing just headers as QoF does. We note that Bro reported missing bytes in on the order 10^{-5} of the flows it counted[7]. Upon further investigation, we found potential faults in Bro that would lead to *undercounting* of observation loss, and reported these to the Bro mailing list; a quantification of this undercounting and correction thereof are subjects for future work.

While QoF in this configuration required 29 s to analyze 31 GB of traffic[8], Bro required 730 s in its default configuration and 92 s in bare mode with the connection analyzer only; i.e., QoF is 25 times faster than Bro's default configuration, and still more than three times faster than bare mode, respectively.

In summary, as even a lossy flow rate on the order to 2 per 10000 is acceptable for our analysis of Horizon Extender, we consider our analysis validated by this check. Further, the lack of any evidence of large groups of lossy flows, as with the leaky bucket used in the evaluation, indicates that the span port from which we captured traffic was never significantly overloaded.

6 Conclusions

In this work, we have shown the value of inline data integrity reporting for passive measurement, in the context of the QoF TCP-aware flow metering tool. Presently ongoing work includes the application of QoF to other data sets, and further investigation of other signals available in passively observed TCP traffic, e.g. timestamp frequencies from many TCP stacks, which provide an external timing reference for isolating timing jitter at observation points.

[7] For comparisons to Bro, we set QoF's active timeout to 1 hour (as opposed to the default, 5 minutes) to reduce timeout-based overcounting, based on our model of potential loss overcounting in QoF (see section 3).

[8] To provide the required data rate for this experiment, the packets have been read from a pcap file that was stored on a ramfs.

Future developments in QoF can be followed in source form: QoF is available under the GNU General Public License, and is under active development. The latest version is always available from `http://github.com/britram/qof`[9].

7 Acknowledgments

This work was materially supported by the European Commission though the Seventh Framework Grant Agreement mPlane (FP7-318627); no endorsement of the work by the Commission is implied. Many thanks to Shane Alcock, Bernhard Ager, Jeff Boote, Mirja Kühlewind, and Jinyao Yan for the fruitful discussions during QoF's development; to Emily Sarneso, Chris Inacio, and the CERT Network Situational Awareness Group for further developing YAF, on which QoF is based; and to ITS at the University of Auckland for one of the data sets used in the development of this study.

References

1. Pelsser, C., Cittadini, L., Vissicchio, S., Bush, R.: From Paris to Tokyo: On the suitability of ping to Measure Latency. In: Internet Measurement Conference 2013, Barcelona, Spain (October 2013) 125–131
2. Hofstede, R., Drago, I., Sperotto, A., Sadre, R., Pras, A.: Measurement Artifacts in NetFlow Data. In: Passive and Active Measurement (PAM) 2013, Hong Kong SAR, China, Springer-Verlag (LNCS 7799) (March 2013) 1–10
3. Cunha, I., Silveira, F., Oliveira, R., Teixeira, R., Diot, C.: Uncovering artifacts of flow measurement tools. In: PAM. (2009) 187–196
4. Trammell, B., Tellenbach, B., Schatzmann, D., Burkhart, M.: Peeling Away Timing Error in NetFlow Data. In: Passive and Active Measurement (PAM) 2011, Atlanta, Georgia, USA, Springer-Verlag (LNCS 6579) (March 2011) 194–203
5. Kögel, J.: One-way delay measurement based on flow data: Quantification and compensation of errors by exporter profiling. In: ICOIN. (2011) 25–30
6. Claise, B., Trammell, B., Aitken, P.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Flow Information. RFC 7011 (Internet Standard) (September 2013)
7. Brauckhoff, D., Tellenbach, B., Wagner, A., May, M., Lakhina, A.: Impact of Packet Sampling on Anomaly Detection Metrics. In: Internet Measurement Conference 2006, Rio de Janerio, Brazil (October 2006)
8. Zseby, T., Hirsch, T., Claise, B.: Packet Sampling for Flow Accounting: Challenges and Limitations. In: Passive and Active Measurement (PAM) 2008, Cleveland, Ohio, USA, Springer-Verlag (LNCS 4979) (March 2008) 61–71
9. Inacio, C., Trammell, B.: Yaf: Yet another flowmeter. In: Proceedings of the 24th Large Installation System Administration Conference (LISA 2010), San Jose, California, USA, USENIX Association (November 2010) 107–118
10. Alcock, S., Lorier, P., Nelson, R.: Libtrace: a packet capture and analysis library. SIGCOMM Comput. Commun. Rev. **42**(2) (March 2012) 42–48

[9] The code used in writing this paper is in the `albula` branch, while the `master` branch will always contain a stable version of the most recent feature set.

11. Veal, B., Li, K., Lowenthal, D.: New methods for passive estimation of tcp round-trip times. In: Proceedings of the 6th international conference on Passive and Active Network Measurement. Passive and Active Measurement (PAM) 2005, Boston, Mass., USA, Springer-Verlag (LNCS 3431) (2005) 121–134
12. Mellia, M., Meo, M., Muscariello, L., Rossi, D.: Passive analysis of tcp anomalies. Computer Networks **52**(14) (2008) 2663 – 2676
13. Strowes, S.D.: Passively measuring tcp round-trip times. Communications of the ACM **56**(10) (October 2013)
14. Trammell, B., Schatzmann, D.: On Flow Concurrency in the Internet and its Implications for Capacity Sharing. In: Proceedings of the Second ACM CoNext Capacity Sharing Workshop (CSWS), Nice, France (Dec 2012)
15. Brownlee, N.: One-way Traffic Monitoring with iatmon. In: Passive and Active Network Measurement Workshop (PAM), Vienna, Austria, PAM 2012 (Mar 2012)
16. Paxson, V.: Bro: a system for detecting network intruders in real-time. Computer Networks **31** (1999) 2435 – 2463
17. Gugelmann, D., Schatzmann, D., Lenders, V.: Horizon Extender: Long-term Preservation of Data Leakage Evidence in Web Traffic. In: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, Hangzhou, China (2013) 499–504