

Realistic Passive Packet Loss Measurement for High-Speed Networks

Aleš Friedl¹, Sven Ubik¹, Alexandros Kapravelos², Michalis Polychronakis², and Evangelos P. Markatos²

¹ CESNET, Czech Republic {afriedl, ubik}@cesnet.cz

² FORTH-ICS, Greece {kapravel, mikepo, markatos}@ics.forth.gr

Abstract. Realistic and accurate packet loss measurement of production traffic has been challenging, since the frequently-used active monitoring approaches using probe packets cannot capture the packet loss experienced by the traffic of *individual* user applications. In this paper, we present a new approach for the accurate measurement of the packet loss rate faced by actual production traffic based on passive network monitoring. In contrast to previous work, our method is able to pinpoint the packet loss rate experienced by the *individual* traffic flows of concurrently running applications. Experimental results suggest that our approach measures packet loss with 100% accuracy for network speeds as high as 12 Gbit/s, while traditional ICMP-based approaches were usually much less accurate. We also report experiences from a real-world deployment of our method in several 10 Gbit/s links of European research networks, where it has been successfully operational for several months.

1 Introduction

Packet loss is an important performance characteristic of network traffic, crucial for applications including long-range data transfers, video and audio transmission, as well as distributed and GRID computing. Unfortunately, most of the existing tools report only *network link* packet loss rate and cannot measure the actual packet loss experienced by the traffic of *individual* applications. Most of the existing techniques are based on active network monitoring, which involves the injection of probe packets into the network for measuring how many of them eventually reach their final destination [2, 10, 11]. Although these approaches approximate the overall packet loss of a link, they inherently cannot measure the packet loss faced by the traffic of individual applications. To make matters worse, for accurately approximating bursty and volatile packet loss events, active monitoring methods need to inject a large number of packets, increasing their intrusiveness in the network, and possibly perturbing the dynamics of the system. When using a small number of probe packets to avoid a high level of intrusiveness, such methods need to run for a long period, and then are only able to approximate packet loss rates that remain constant for a long duration—a highly unlikely case in real networks.

In contrast to active monitoring approaches, in this paper we describe a real-time end-to-end packet loss measurement method for high-speed networks based on distributed *passive* network monitoring. The main benefit of the proposed approach is the accurate measurement of the *actual* packet loss faced by user traffic, both in terms of

loss magnitude, as well as the identification of the individual traffic flows that were affected. Such fine-grained per-application packet loss measurement is important in case different applications on the same network path exhibit different degrees of packet loss, e.g., due to the deployment of differentiated services and service level agreements (SLAs), rate-limiting devices, or load-balancing configurations.

We presented a prototype version of a passive packet loss estimation method in our previous work [9]. In this paper, we describe a significantly enhanced version, called `PcktLoss`, that measures packet loss with higher precision, can detect even very short packet loss events, and has been proved to work reliably for multi-Gigabit traffic in a real-world deployment at the GÉANT2 network, which interconnects most of the national research and education networks (NRENs) in Europe.

2 Related Work

`Ping` is one of the most popular tools for inferring basic network characteristics, such as round-trip time and packet loss. `Ping` sends ICMP probe packets to a target host at fixed intervals, and reports loss when the response packets are not received within a specified time period. Although `ping` has been used as a first-cut mechanism for link packet loss estimation, its applicability has recently started to get limited because several routers and firewalls drop or rate-limit ICMP packets, which introduces *artificial* packet loss that undermines the accuracy of the measurement. Instead of using ICMP packets, `zing` [2] and `Badabing` [11] estimate end-to-end packet loss in one direction between two cooperative end hosts by sending UDP packets at pre-specified time intervals. `Sting` [10] overcomes the limitation of requiring two cooperative hosts by measuring the link loss rate from a client to any TCP-based server on the Internet based on the loss recovery algorithms of the TCP protocol.

Benko and Veres have proposed a TCP packet loss measurement approach based on monitoring sequence numbers in TCP packets [4]. Our approach uses a completely different estimation approach, independent from the L4 protocol specification, and thus can be universally applied to both TCP and UDP connections. Ohta and Miyazaki [8] have explored a passive monitoring technique for packet loss estimation relying on hash-based packet identification. Their work is similar to our approach, but ours differs in that it matches packets to flows and compares flows with each other for computing the packet loss, while theirs hashes the packet's payload and correlates them. Our approach is more lightweight and thus can be performed on-line, while Ohta and Miyazaki's technique needs to stop monitoring for computing the packet loss.

3 Architecture

Over the past few years, we have been witnessing an increasing deployment of passive network monitoring sensors all over Europe. In this paper, we propose to capitalize on the proliferation of passive monitoring sensors and use them to perform accurate per-application packet loss measurements. Our approach is quite simple: assuming a network path equipped with two passive monitoring sensors at its endpoints, as shown in Fig. 1, measuring packet loss is just a matter of subtraction: by subtracting the number

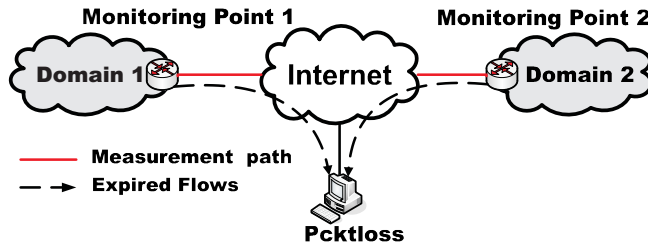


Fig. 1. Overall architecture of PcktLoss.

of packets arrived at the destination from the number of packets that were originally sent, one can find exactly how many packets were lost in the network.

Unfortunately, our algorithm is a little more complicated than what we have simplistically described. Indeed, the timing details of the subtraction are crucial for the correct calculation of the loss rate. A prematurely computed subtraction may report as lost all packets that have left their source but have not yet reached their destination. To accurately define the timing of the subtraction, we base our method on the concept of *expired flows*. A flow is defined as the set of IP packets with the same L4 protocol, source and destination IP address, and source and destination port (also known as a 5-tuple). A flow is considered *expired* if no packet has arrived within a specified timeout (30 sec in our experiments). This differs from the traditional Netflow or IPFIX flow records, which also report long-running flows. In case of TCP, a flow can also be considered expired when the connection is explicitly closed, i.e., when an RST or FIN packet is seen.

To calculate per-application packet loss, our algorithm periodically retrieves the expired flows from the two passive monitoring sensors at the endpoints of the network path. Each record includes a flow identifier (the 5-tuple), the number of packets and transferred bytes, as well as the TTL and timestamp of the first packet of the flow. If the same expired flow is reported from both sensors, but with a different number of packets, then this is an indication that the flow lost some packets, and the actual packet loss rate can be computed from the difference of the reported number of packets. Flows with only one packet captured are ignored, since they will not be matched if their packet is lost. This limitation derives from the fact that we are not always sure if this traffic is routed through our observation points. Therefore we cannot decide if the packet was lost or avoided all other observation points.

4 Experimental Evaluation

4.1 Comparison with Active Monitoring

Our experiments aim to explore the accuracy of PcktLoss compared to ping, probably the most widely used packet loss measurement tool based on active monitoring, as well as verify that our method measures the actual packet loss of existing traffic without deviations. Our experimental environment consists of two PCs, a “sender” and a “receiver,” also acting as passive monitoring sensors for PcktLoss. The traffic between

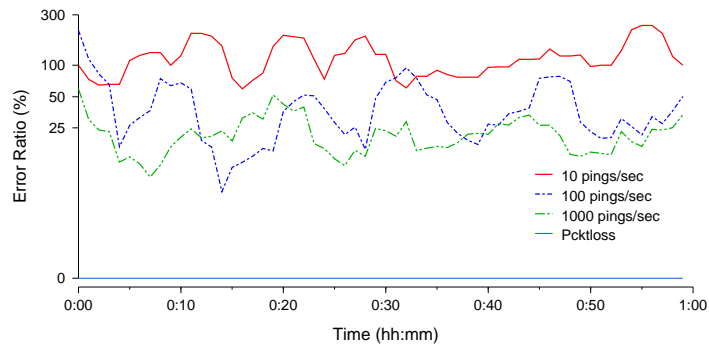


Fig. 2. Measurement error for ping and PcktLoss when introducing a constant loss rate of 0.1%. PcktLoss reports the actual loss rate without deviations.

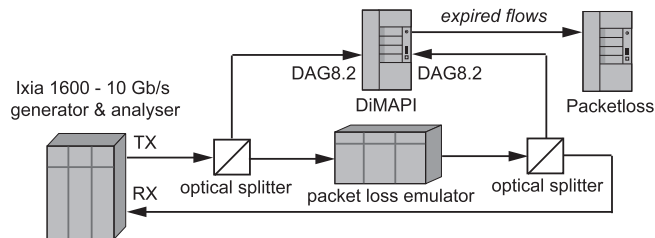


Fig. 3. Experimental environment for performance testing.

the two sensors is transparently forwarded through a third PC that introduces artificial packet loss at a controlled rate using `netem` [6]. We generated UDP traffic with `mgen` [3], which uses explicit sequence numbers and logging at both ends to calculate the actual number of lost packets. We used 1 Mbit/s traffic with 1KB packets to prevent the passive monitors from dropping packets due to excessive load, since both sensors used commodity Ethernet interfaces. Each run lasted for one hour.

In our experiment, we introduce a constant packet loss rate of 0.1% to all traffic between the two sensors. Figure 2 presents the measurement error ratio for ping using different probe sending rates, as well as for PcktLoss. The error ratio is calculated based on the packet loss reported by `mgen`. As expected, the lower ping's probe sending rate, the higher its measurement error. Even when using an aggressive rate of 1000 probe packets per second, ping still cannot accurately measure the actual packet loss. In contrast, PcktLoss measures the actual packet loss without errors.

It is possible for a network path to exhibit packet loss only for certain classes of traffic, e.g., due to a traffic shaping policy. In this case, the probe traffic of an active monitoring tool may not face the same packet loss as the production traffic.

Generated rate for both links	Processed packets
10 Gbit/s	100 %
12 Gbit/s	100 %
14 Gbit/s	99.44 %
16 Gbit/s	90.11 %
18 Gbit/s	79.65 %
20 Gbit/s	72.19 %

Table 1. PcktLoss throughput.

Emulated loss rate	# packets dropped by the emulator	# lost packets as reported by PcktLoss
10^{-2}	14000000	14000000
10^{-3}	1400000	1400000
10^{-4}	140000	140000
10^{-5}	14000	14000
10^{-6}	1400	1400
10^{-7}	146	146

Table 2. PcktLoss precision.

4.2 Runtime Performance

We tested the performance of PcktLoss under heavy traffic load in the controlled environment shown in Fig. 3. We used the Ixia 1600 packet generator and analyser to send and receive traffic at a 10 Gbit/s rate. The traffic passes through a custom FPGA-based device that introduces artificial packet loss by selectively dropping packets at a specified rate. The traffic before entering and after leaving the packet loss emulator is diverted through optical splitters to two DAG8.2 monitoring cards installed on a PC running Linux and MAPI, while PcktLoss runs on a different PC. Both PCs are equipped with two quad-core 3 GHz Intel Xeon Woodcrest CPUs.

We configured the packet generator to send 500 UDP and 500 TCP flows using varying packet sizes according to the RFC2544 [5]. The throughput achieved for different traffic rates is presented in Table 1. For speeds up to 12 Gbit/s, PcktLoss processed 100% of the traffic without dropping any packets. Note that the monitoring sensor had to process twice the traffic from both monitoring cards. If each card were installed on a separate PC, it should be possible to monitor full 10 Gb/s of traffic.

In our next set of experiments, we ran a series of tests by setting the packet loss emulator to introduce a loss rate ranging from 10^{-2} to 10^{-7} . On each run, the traffic generator transmitted $1.4 * 10^9$ packets at a speed of 5 Gbit/s. As shown in Table 2, in all cases PcktLoss was able to measure the exact number of lost packets. For a loss rate of 10^{-7} the emulator actually dropped slightly more packets. We doubly verified the precision of the results reported by PcktLoss by comparing them with the actual number of lost packets as reported both by the packet loss generator, as well as by the traffic generator which also receives back the generated traffic.

5 Real-world Deployment

We have installed PcktLoss on several sensors deployed in the GN2 network, which interconnects the National Research and Educational Networks (NRENs) of most European countries. The networks involved in monitoring are CESNET, PIONIER, SWITCH, connected by 10 Gbit/s links, and ACAD, which is connected by a 1 Gbit/s link.

The runtime performance of PcktLoss in this deployment is summarized in Table 3, which presents statistics for one week of continuous monitoring. Traffic load

Monitoring station	max 5-min traf- fic load [Mb/s]	5-min CPU load [%]	packets processed in 1 week	packets dropped in 1 week
SWITCH out	2800	10+10 (2 cores)	$1.62 * 10^{10}$	0
SWITCH in	6800	40+20 (2 cores)	$8.33 * 10^{10}$	0
PIONIER out	240	5	$1.55 * 10^9$	83991
PIONIER in	370	20	$2.00 * 10^9$	5083
ACAD in+out	535	40	$3.30 * 10^9$	0
CESNET in+out	440	90	$1.64 * 10^{10}$	344796
Total			$1.23 * 10^{11}$	433870

Table 3. Passive and active loss measurements and `PcktLoss` performance.

refers to the maximum load among all 5-minute intervals over the week. The indicated CPU load was measured during the same interval. The monitoring cards on the two most loaded links did not drop any packets and the CPUs were not fully utilized, which demonstrates the scalability of our approach. There were occasional packet drops on three of the sensors due to known configuration shortcomings: the CESNET sensor has much slower memory, while the DAG cards in PIONIER use the PCI-X bus, which cannot transfer traffic bursts without dropping packets. It should be noted that `PcktLoss` was just one of three concurrently running passive monitoring applications on the same sensor. Each sensor also hosted ABW [12] to monitor short-term traffic load and distribution into protocols, and Burst [13] to quantify traffic burstiness. Particularly ABW is quite CPU-intensive, since it performs header-based classification for all packets and payload searching for selected packets.

Overall, `PcktLoss` reported 2,737,177 lost packets (out of which 433,870 were dropped by the monitoring cards due to overload), corresponding to an actual packet loss rate of $2.22 * 10^{-5}$. Most packet loss events occurred during short periods, whereas most of the time the packet loss rate was minimal. During the same measurement period, we also used the active monitoring tool Hades [1, 7] to measure the packet loss rate between the same pairs of networks. Hades estimates the packet loss rate of a path by sending a burst of 9 packets with 30 ms offset every minute. In contrast to `PcktLoss`, Hades reported only 245 lost packets.

6 Conclusion

We presented the design and implementation of `PcktLoss`, a novel method for the accurate measurement of the packet loss faced by user traffic. Based on passive network monitoring, `PcktLoss` can measure the packet loss ratio of *individual* traffic flows, allowing to pinpoint loss events for specific classes of traffic. Our experimental evaluation and real-world deployment have shown that `PcktLoss` can precisely measure the packet loss rate even when monitoring multi-Gigabit traffic speeds.

In our future work, we plan to explore how to conveniently integrate checks for packet drops in the packet capturing cards for eliminating any reported packet loss due

to temporary overload. We also plan to explore how to efficiently monitor the packet loss rate in the presence of IP fragmentation halfway into the monitored network path.

Acknowledgments

This work was supported in part by the IST project LOBSTER funded by the European Union under Contract No. 004336. The work of Alexandros Kapravelos, Michalis Polychronakis and Evangelos Markatos was also supported by the GSRT project Cyberscope funded by the Greek Secretariat for Research and Technology under Contract No. PENED 03ED440. Alexandros Kapravelos, Michalis Polychronakis and Evangelos Markatos are also with the University of Crete.

References

1. Hades active delay evaluation system. <http://www-win.rrze.uni-erlangen.de/ippm/hades.html.en>.
2. Andrew Adamns, Jamshid Mahdavi, Matthew Mathis, and Vern Paxson. Creating a scalable architecture for internet measurement. In *Proceedings of INET*, 1998.
3. B. Adamson. The MGEN Toolset. <http://pf.itd.nrl.navy.mil/mgen>.
4. Peter Benko and Andreas Veres. A Passive Method for Estimating End-to-End TCP Packet Loss. In *Proceedings of IEEE Globecom*, pages 2609–2613, 2002.
5. S. Bradner and J. McQuaid. Benchmarking Methodology for Network Interconnect Devices. RFC 2544 (Informational), March 1999. <http://www.ietf.org/rfc/rfc2544.txt>.
6. S. Hemminger. Network Emulation with NetEm. In *Proceedings of Linux Conf Au*, 2005.
7. T. Holleczeck. Statistical analysis of IP performance metric in international research and educational networks (diploma thesis), 2008.
8. S. Ohta and T. Miyazaki. Passive packet loss monitoring that employs the hash-based identification technique. In *Ninth IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2005.
9. A. Papadogiannakis, A. Kapravelos, M. Polychronakis, E. P. Markatos, and A. Ciuffoletti. Passive end-to-end packet loss estimation for grid traffic monitoring. In *Proceedings of the CoreGRID Integration Workshop*, pages 79–93, 2006.
10. Stefan Savage. Sting: A TCP-based network measurement tool. In *USENIX Symposium on Internet Technologies and Systems (USITS)*, 1999.
11. Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Improving accuracy in end-to-end packet loss measurement. In *Proceedings of the ACM SIGCOMM '05*, pages 157–168.
12. S. Ubik, V. Smotlacha, S. Trocha, S. Leinen, V. Jeliaskov, A. Friedl, and G. Kramer. Report on passive monitoring pilot, September 2008. Deliverable MS.3.7.5 GN2 Project.
13. Sven Ubik, Aleš Friedl, and Stanislav Hotmar. Quantification of traffic burstiness with mapi middleware. September 2008.