Assessing the Quality of Packet-Level Traces Collected on Internet Backbone Links

Behrooz Sangchoolie[1], Mazdak Rajabi Nasab[1], Tomas Olovsson[1], Wolfgang John[2]

[1] Chalmers University of Technology, Department of Computer Science and Engineering, SE-412 96 Gothenburg, Sweden
behrooz.sangchoolie@chalmers.se, mazdak@student.chalmers.se, tomas.olovsson@chalmers.se
[2] Ericsson Research, Kista, Sweden
wolfgang.john@ericsson.com

Abstract. The quality of captured traffic plays an important role for decisions made by systems like intrusion detection/prevention systems (IDS/IPS) and firewalls. As these systems monitor network traffic to find malicious activities, a missing packet might lead to an incorrect decision. In this paper, we analyze the quality of packet-level traces collected on Internet backbone links using different generations of DAG cards[1]. This is accomplished by inferring dropped packets introduced by the data collection system with help of the intrinsic structural properties inherently provided by TCP traffic flows. We employ two metrics which we believe can detect all kinds of missing packets: i) packets with ACK numbers greater than the expected ACK, indicating that the communicating parties acknowledge a packet not present in the trace; and ii) packets with data beyond the receiver's window size, which with a high probability, indicates that the packet advertising the correct window size was not recorded. These heuristics have been applied to three large datasets collected with different hardware and in different environments.
We also introduce *flowstat*, a tool developed for this purpose which is capable of analyzing both captured traces and real-time traffic. After assessing more than 400 traces (75M bidirectional flows), we conclude that at least 0.08% of the flows have missing packets, a surprisingly large number that can affect the quality of analysis performed by firewalls and intrusion detection/prevention systems. The paper concludes with an investigation and discussion of the spatial and temporal aspects of the experienced packet losses and possible reasons behind missing data in traces.

Keywords: Traffic measurement, measurement errors, packet drop, intrusion detection/prevention system, firewall

[1] This work was supported in part by Swedish University Computer Network (SUNET).

1 Introduction

Firewalls and intrusion detection/prevention systems monitor network traffic for malicious activities. These systems make decisions based on the observed traffic. For instance if an attack's signature is a sequence of two packets, with the first one containing pattern X followed by the second one containing pattern Y, missing either of these packets by intrusion detection/prevention systems might result in a false negative decision. For this reason, it is vital that they have the ability to capture all packets on network. This is one of the motivations for network traffic analysis.

Even though simulating the behavior of different Internet protocols is possible using well-known simulators, it is mostly believed that genuine Internet traffic analysis is more advantageous. Plenty of traffic measurement research has been performed so far; some focused on backbone traffic [1] [2] [3] while others aimed more at edge links [4] [5]. In spite the advantages of analyzing genuine Internet traffic, operational limitations always create challenges. As an example, special purpose hardware and software are required in order to capture traffic from high speed backbone links. Manufacturers of high speed measurement cards claim that these cards can capture Internet traffic with 100% accuracy [6].

The community generally relies on the manufacturer's claim; therefore not much research has been done to verify trace quality. Plenty of literature focuses on statistical properties of Internet traffic where missing a couple of packets do not have any consequences. Others aim at systems that require all transmitted packets to be present, such as intrusion detection systems. In order to detect all kinds of malicious traffic, an intrusion detection/prevention system should be able to capture all the transmitted packets.

In this paper we have analyzed the quality of packet-level traces with respect to missing packets. The data has been collected on 10Gbit/s Internet backbone links within the MonNet[2] project [1]. The result of our work can be used to find out whether systems such as firewalls and intrusion detection/prevention systems are able to capture all transmitted packets for malicious traffic detection. This can then be used to verify the correctness of their behavior.

1.1 Related Work and Contributions

To the best of our knowledge, there have been very little investigations showing that measurement cards are capable of capturing all the ongoing packets of their capturing link. Even NSS labs' *Attack Leakage* test [7] only evaluates deep packet inspection capability of *EndaceProbes*, a server hardware product capable of running high speed packet analysis using DAG cards. In this test the accuracy and performance of IPS/IDS devices are evaluated. Devices are tested against a test traffic that contains known number of attack vectors. The load is increased to a point where the device under the test starts to miss detection of attack vectors. They also show that

[2] MonNet project aimed to provide better understanding of Internet traffic characteristics based on passive measurement of backbone links.

EndaceProbes can capture 100% of packets at 10 Gbit/s although they have not tested DAG cards on other server solutions to see if they drop any packets for example as a result of resource limitations.

Among the available tools, *tcpanaly* [8] is one of the oldest. Given a trace file, *tcpanaly* verifies whether the observed TCP implementations follow the TCP standard. It can also produce detailed statistics for TCP connections. One weakness of *tcpanaly* is that it performs a two-pass analysis on the trace file. The two-pass analysis results in a more time-consuming assessment and is not suitable for real-time analysis. *Tcpanaly* also assumes that an ACK will always be sent back on the arrival of any out-of-sequence data [8] which is a big limitation since not all end systems follow this behavior. *Tstat* [5] is another relevant tool developed especially for statistical analysis of TCP/IP traffic. *Tstat* uses the libpcap library and is capable of calculating more than 80 different performance statistics. However *tstat* does not take missing packets into account within its comprehensive TCP logs; therefore it cannot be used to analyze packet drops.

The main goal of this paper is to analyze the quality (in term of completeness) of collected traffic traces. In this way, we can investigate whether systems such as firewalls have access to all the transmitted packets to detect suspicious activities. Specifically, this paper contributes with the following:

- The development and implementation of a method to analyze the quality of data with respect to missing packets in data captured in traffic measurement campaigns.
- An investigation of the spatial and temporal aspects of the experienced missing packets in traces and plausible reasons behind missing data.
- A tool called *flowstat*, designed for this purpose, which can operate on saved traces and on real-time traffic. *Flowstat* is written in standard C and is open to the research community for further development[3].

The remainder of this paper is organized as follows. In section 2, we describe the data collection hardware and data sets which we used in our analysis. In section 3 we present the reasons for missing packets in collected data. Section 4 describes our methodology to detect missing packets. We present the overview of *flowstat* in section 5 and in section 6 we present the results of our analysis. We conclude with a summary of major findings and suggestions for further research in section 7.

2 Data Collection Hardware and Data Sets

Three different data collection campaigns of PoS HDLC traffic have been performed. The first campaign was conducted during 2006 on an OC192 link inside the GigaSUNET [9] network. The second campaign was conducted with the same hardware but at a new physical location in the updated OptoSUNET [10] network. In the third campaign, the physical location remained unchanged while the infrastructure was slightly altered. Moreover, new system hardware including a new generation of DAG cards was used. The hardware used consisted of high-end systems at the time of

[3] The tool is available at http://sourceforge.net/projects/flowstat/

purchase. For simplification, we name campaign (2) and (3) OptoSUNET1 and OptoSUNET2, respectively. In GigaSUNET and OptoSUNET1, two measurement nodes were used, one for each traffic direction and each equipped with one DAG card. In OptoSUNET2, one system was used with two data collection cards (Table 1).

The DAG cards were configured with a buffer reserved from the system's main memory in order to deal with traffic bursts. For the hardware used in GigaSUNET and OptoSUNET1, Endace recommends a buffer size of at least 32MB for OC-12 links, thus we conservatively chose a large buffer of 512 MB for our OC-192 links. In OptoSUNET2 we used DAG Tools ver. 4.2.0 without changing the default configuration which used a buffer of 256 MB. The optical splitters were changed between GigaSUNET and OptoSUNET1, but remained the same between OptoSUNET1 and OptoSUNET2. Since the signal strength was quite high, splitters with a 90/10 ratio turned out to be sufficient for the sensitivity of the measurement cards in all campaigns [11].

All traces have a duration of 20 minutes or longer. Packets with IP checksum errors are preserved in our traces although packets with link-level CRC errors were automatically discarded by the cards. In GigaSUNET, the DAG cards were configured to capture only the first 120 bytes, and in OptoSUNET1 and OptoSUNET2, only 160 bytes. Given that the average packet size on the links was 687 bytes and that 44% of all frames were smaller than 120 bytes and 160 bytes, respectively (and thus not truncated by the DAG card), we calculated the average packet size to be stored on disk to 88 and 110 bytes, respectively.

This means that even at a maximum link utilization of 10 Gbps, only about 200 MByte/s had to be transferred to disk. However, due to heavy over-provisioning of the links, in reality the nodes rarely needed to store more than 35 MByte/s (280 Mbps) on disk during the measurement campaigns. Disk and processor performance should therefore never have been an issue.

Table 1. Collection systems technical specifications

	GigaSUNET and OptoSUNET1	**OptoSUNET2**
Motherboard	Tyan K8SR	Dell PowerEdge R710
CPU	Two 2 GHz 64-bit AMD Opteron	Two Intel Xeon E5620 2.4 GHz Quad-core Hyper-Threaded
Memory	2 GB (1 GB per CPU)	16 GB DDR3 (shared among CPUs)
Bus dedicated to DAG cards	133 MHz 64-bit PCI-X	Dual PCIe x8
Disk controller	Dual-channel Ultra-320 SCSI	PERC 6/I RAID controller
Disks	Six SCSI RAID-0 (software)	Six SATA RAID-0 (hardware)
Tested sustained disk write throughput	410 MByte/s (for each DAG card)	520 Mbyte/s (shared between DAG cards)
Measurement card (DAG)	DAG6.2SE	DAG8.1SX
Number of cards per system	1	2

A longer discussion of possible limitations in data collection campaigns can be found in [11]. After frame truncation, traces were *de-sensitized* and *sanitized* [11] [12]. The de-sensitization consisted of two phases; first the payload of each packet was removed using CAIDA's coralReef [13] crl_to_dag utility and then IP addresses were anonymized using prefix-preserving Crypto-Pan [14]. Also as a result of the prefix preserving nature of the anonymization, neighbor addresses will also be neighboring after anonymization. Sanitization checks were also applied before and after each de-sensitization step to verify the correctness of trace pre-processing.

3 Reasons for Missing Packets in Collected Network Data

According to Paxson [8], there are four types of measurement errors that affect the quality of data; drops, additions, resequencing and timing. In this paper, our main focus is to investigate and quantify the amount of packets dropped by the measurement nodes or missed before being captured. Based on the mentioned measurement errors and the data collection hardware used, we identify four sources of measurement errors:

- Errors that are introduced by the DAG card, possibly as a result of frame truncation, insufficient buffer space, PCI bus limitations and losses between the DAG card and memory.
- Errors that occur if the measurement nodes do not accept all packets communicated by the end nodes, like packets with link-level CRC errors.
- Errors introduced in trace pre-processing, e.g. de-sensitization.
- Missing packets that did not appear on the measured link due to alternative routes between the communicating nodes. These packets might be missed due to routing policies along the end-to-end path, such as load balancing.

Any packet drop due to insufficient buffer space, PCI bus limitations and losses between the DAG card and memory is supposed to be reported by DAG cards [11]. Moreover, however unlikely, it is possible that buggy or faulty sanitization tools introduce errors like packet drops into the traces.

All SUNET generations follow a per-flow routing policy. For this reason when a flow's packet is observed on the link we expect to observe all of that flow's packets on the same link. However, there is no guarantee for this assumption. As routing happens on a packet-per-packet basis, it is possible that a flow's packets are routed through different links.

Since TCP packets account for more than 90% of our captured packets, we made the decision to apply two TCP based metrics on the traces in order to discover missing packets. In the first metric, described in sub-section 4.1, we keep track of packets to see if any packet that is not recorded by the measurement node is acknowledged by the end systems. The second metric, explained at sub-section 4.2, assesses whether any end-points transmit data beyond the allowed TCP window size, which is a possible indication of missing packets.

4 Methodology

We have focused on two metrics that we believe are sufficient for detecting all types of missing packets in TCP flows. Only bidirectional flows that have finished their three-way handshake are evaluated. For simplicity throughout the remaining of this paper, we use the term "bidirectional flows" instead of "bidirectional flows with observed three-way handshake". In this section the used metrics are explained.

4.1 Metric One[M1]: End Systems Acknowledge a Packet not Present in the Trace

In this metric, *flowstat* keeps track of the next expected ACK number, *expack*, in each direction (Fig. 1). After a successful three-way handshake, *flowstat* keeps track of the packets and compares each ACK with the data it has observed up to that point. *Flowstat* compares each packet's ACK number with its corresponding *expack*. A packet with an ACK number greater than *expack* denotes one or more missing packets (Fig. 2). *Flowstat* is able to deal with out of order packets.

As soon as a measurement error is detected, the erroneous flow will no longer be analyzed by *flowstat*, i.e. *flowstat* currently counts the number of flows with at least one missing packets and not the total number of missing packets in the flow. We decided on this approach because it is not possible to know exactly how many packets were actually missed when there is an ACK greater than *expack*. Moreover an M1 measurement error will also most probably lead to more M1 measurement errors. If desired, it is easy to modify *flowstat* to update *expack* with the value found in the packet raising the error in order to approximate the amount of erroneous packets.

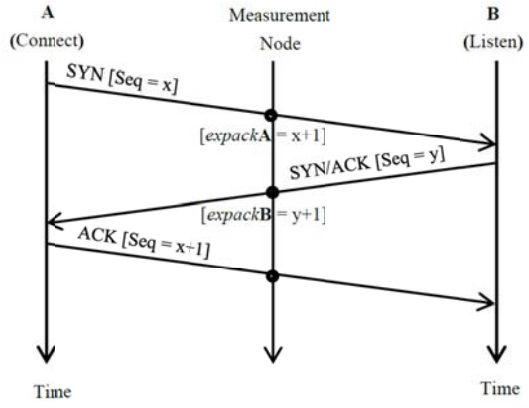

Fig. 1. The initial *expack* for each direction (measurement node's view of the traffic)

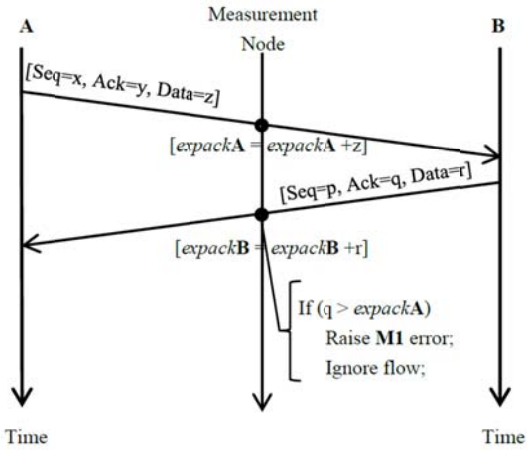

Fig. 2. When to raise an M1 measurement error (measurement node's view of the traffic)

Despite the fact that M1 is a powerful metric, it is unable to detect missing zero-length packets such as ACK packets carrying no data, keep alive packets, etc. Since TCP is using a cumulative ACK system, keeping track of the packet lengths, as explained in M1, will not assist us in discovering missing zero-length packets. In our next proposed metric, the *TCP window* field will be used to equip *flowstat* with yet another way of detecting missing packets including zero-length packets.

4.2 Metric Two[M2]: Data Beyond the Advertised Window

Inspired by Vern Paxson [8], we also implement another powerful check that can be used to detect missing packets. M2 evaluates whether a sender has sent data beyond the receiver's earlier advertised window size. Nonstandard or faulty TCP implementations may cause this behavior. Feng Qian et al. [15] believe that apart from erroneous TCP implementations, the underutilization of the congestion window is another reason for observing data beyond the receiver's window size.

However, if both communicating parties are following the TCP standard, it can be assumed with a high probability that the packet advertising the correct window size has not been observed by the measurement node (Fig. 3).

Since there are other reasons apart from packet drops involved in observing data beyond the advertised window size such as different implementation problems of TCP/IP stacks, M2 is not as accurate as M1. On the other hand, unlike M1, M2 can detect missing zero-length packets. Since we used different TCP properties to detect missing packets corresponding to M1 and M2, none of them are necessarily a superset of the other. Therefore *flowstat* reports both these metrics in order to detect missing packets in TCP flows.

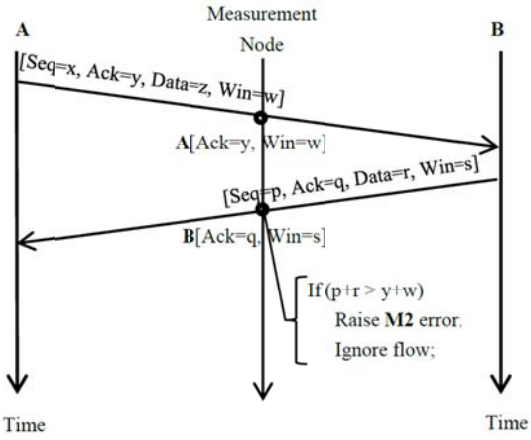

Fig. 3. When to raise an M2 measurement error (measurement node's view of the traffic)

5 Flowstat Overview

In this section, an overview of *flowstat* is given. *Flowstat* keeps a connection table for all active TCP connections and stores necessary information for sessions in each direction. Only flows with an observed 3-way handshake are evaluated. *Flowstat* also recognizes the value of TCP window scale option from the first two packets of each flow's three-way handshake and uses it to calculate the correct advertised window size for each direction to be used in M2.

In our traces, we found packets with TCP set in the IP protocol field but without enough data to build a TCP header [16]. *Flowstat* recognizes these malformed packets and reports them as *packet size errors*, see Table 2. For all other TCP packets, *flowstat* looks up the corresponding active flow from its connection table. If a SYN packet is captured and no associated entry in the connection table is found, a new entry will be created; whereas a non-SYN packet with no corresponding active flow will be ignored. If there is an entry in the connection table that corresponds to a newly captured packet, *flowstat* will update the entry's necessary fields.

Each time an entry is updated, M1 and M2 checks are performed and in case an error is detected, the erroneous flow counter is incremented and the flow's entry is removed from the connection table. Apart from M1 and M2 errors which can lead to the removal of an entry from the connection table, RST and FIN packets also result in removing the corresponding entry. These policies make sure that the size of the connection table does not grow infinitely.

Flowstat also takes advantage of two timers to detect inactive flows and remove them from the connection table. Choosing a proper value for these timers is a tradeoff between longer execution time and the ability to evaluate more flows. The first timer waits for the completion of the uncompleted three-way handshakes for 30 seconds before removing these entries from the connection table. We also used 60 and 120 seconds as the value for this timer, but number of newly added bidirectional flows

was negligible. The second timer triggers the removal of entries for which no related packet has been observed in a 120-second period. Claffy et al. in [17] show that 64 seconds is a good flow timeout. However, we choose a conservative large timeout in order to make sure that *flowstat* detects all possible missing packets.

In order to validate the correctness of *flowstat* and the proposed metrics, two types of tests have been performed on *flowstat* to test its ability to detect missing packets. In the first test, errors were injected manually into random captured files i.e. a number of packets were intentionally removed from these files. *Flowstat* could correctly detect all these missing packets. In the second test, five sample trace files including around one million bidirectional flows were examined by *flowstat* for missing packets. We then randomly selected 10% of the flows with missing packets and manually verified that they correctly were detected as being erroneous.

6 Data Analysis

In this section, we analyze the results of running *flowstat* on the captured traces. After evaluating more than 400 captured traces, the total number of detected measurement errors is shown in Table 2. As Table 2 illustrates, there are a great number of unidirectional flows. This might to some degree be due to the fact that SYN attacks and SYN scans are counted as unidirectional flows. SUNETs network layout and routing policies also introduce a fair amount of asymmetrical routing [18] (e.g. hot-potato routing), and many flows are indeed unidirectional.

Table 2 also shows that the total number of M2 measurement errors is almost three times as large as M1. This might be due to the fact that not just missing packets, but also nonstandard and faulty TCP implementations as well as underutilization of the congestion window and implementation problems of TCP/IP stacks cause us to observe data beyond the receiver's window size, which in turn result in M2 measurement errors.

Table 2. Total number of M1 and M2 measurement errors in different SUNET generations. The values inside parenthesis refer to the number of analyzed traces.

SUNET generation	Total number of packets	Unidirectional flows	Bidirectional flows	Flows with M1 errors	Flows with M2 errors	Packet size error
GigaSUNET(240)	21,136,187,688	62,808,014	54,724,892	44,549	118,177	66,122
OptoSUNET1 (163)	28,446,722,927	192,709,261	19,488,316	9,335	39,102	887,384
OptoSUNET2 (4)	1,838,475,707	495,312,512	1,127,806	4,324	1,235	31,675
Total(407)	51,421,386,322	750,829,787	75,341,014	58,208	158,514	985,181

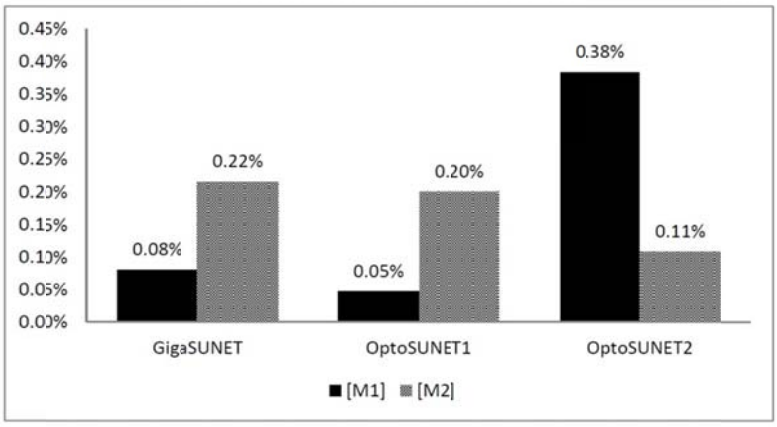

Fig. 4. Percentage of M1 and M2 measurement errors in bidirectional flows

Fig. 4 shows the percentage of M1 and M2 measurement errors in bidirectional flows. It can be seen that there is only a small difference in the detected percentage of measurement errors between GigaSUNET and OptoSUNET1, while OptoSUNET2 is considerably different. However, only four trace files have been analyzed from OptoSUNET2 and these traces have been captured on the same day and during a three-hour time period which makes it difficult to draw any accurate conclusions. Moreover, the dissimilar behavior of OptoSUENT2 might be due to new routing policies or the new hardware, which could be faulty or, more likely, not able to cope with the load.

The IP addresses were anonymized in the trace files using the same key, so each IP address consistently shows up as the same IP address after anonymization. Therefore it is possible to check if there are specific erroneous IP addresses which are more represented than the others. Fig. 5 shows the erroneous IP addresses and number of times they occur in each SUNET generation.

In the first two SUNET generations, in Fig. 5, the IP addresses have been scattered more for M2 than for M1, and there are some IP addresses that are overrepresented in the erroneous flows, especially in M2. A possible reason can be that there are nodes that either do not follow the proper TCP implementation or that they send packets with link-level CRC errors which are considered as erroneous by the measurement node and are dropped. Moreover, Fig. 5 shows that a lot of overrepresented erroneous IP addresses are adjacent which might indicate that they belong to the same subnets. It is noticeable that the systems responsible for many of the M1 errors are also responsible for M2 errors in both the GigaSUNET and OptoSUNET1 generations. And as mentioned before, the limited number of OptoSUNET2 traces makes it difficult to draw any accurate conclusions about this SUNET generation.

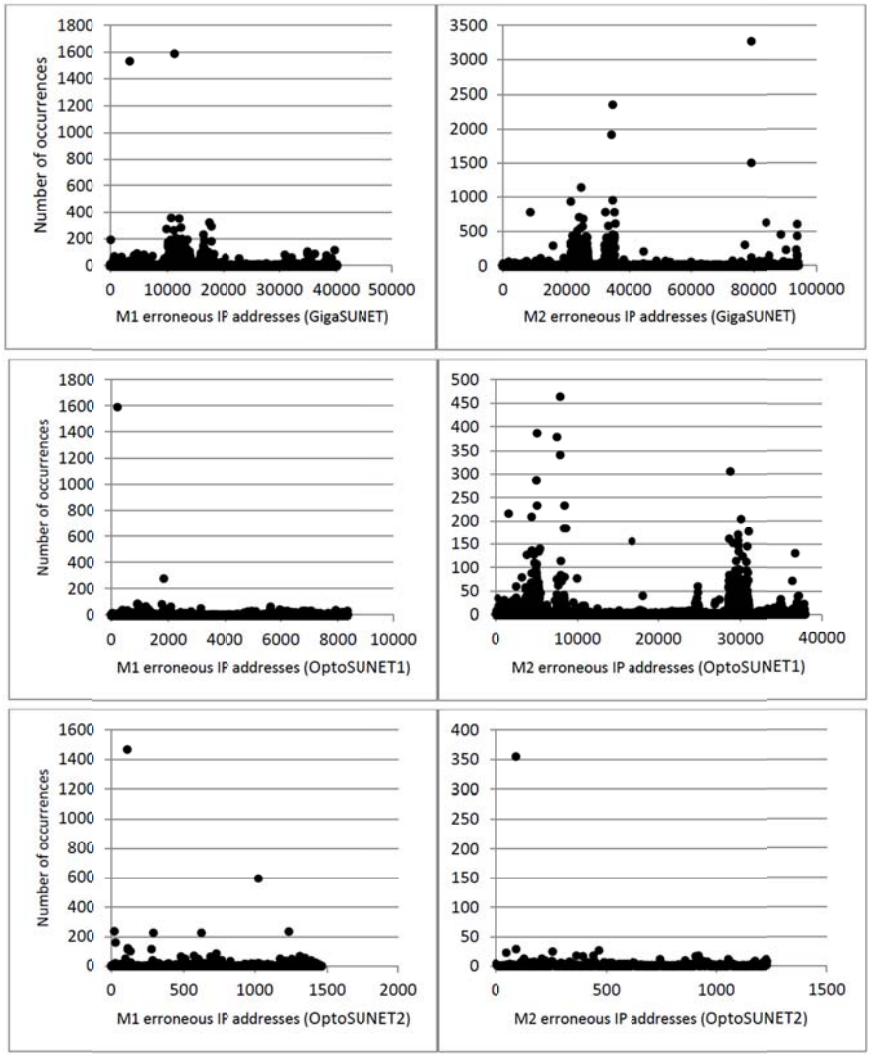

Fig. 5. Number of times an IP address has been seen in erroneous flows. The smallest and the largest values on the x-axis correspond to the smallest and the largest IP addresses respectively.

Table 3, shows the percentage of erroneous IP addresses which were detected more than 10 times along with the percentage of erroneous flows containing these IP addresses. The selected value, 10, is good enough to show that a small group of IP addresses were observed in a lot of erroneous flows. This is also another indication that there might be systems that are not following proper TCP implementations and are responsible for most of the errors. It can also be seen that these small groups of IP addresses are seen in all (100%) of the erroneous flows detected by M2 and M1 in

GigaSUNET and OptoSUNET2 respectively. There are several possible explanations for this behavior:

- The overrepresented end systems send large bursts of traffic and the measurement node's hardware cannot cope with the load.
- The overrepresented end systems send lots of data which leads to more M1 and M2 errors.
- Some packets of the overrepresented end systems are routed a different way.

Table 3. Percentage of erroneous IP addresses detected more than 10 times and the percentage of erroneous bidirectional flows containing these IP addresses.

	M1 error		**M2 error**	
	IP addresses	**Flows**	**IP addresses**	**Flows**
GigaSUNET	2.7%	85%	2.6%	100%
OptoSUNET1	2.6%	64%	2.8%	75%
OptoSUNET2	8.3%	100%	1.5%	51%

As mentioned before, all traces have a duration of 20 minutes or longer and have been captured at different times during the day. We have also classified the traces according to the time that they have been captured. The percentage of measurement errors in different time periods is shown in Table 4. The different time periods roughly experienced the same percentages of measurement errors, even though traces which were captured between 12pm and 12am hold slightly more erroneous flows. It is notable that the percentage of measurement errors is larger during the working hours (6am to 6pm) compared to the other periods and grows even more during the evening. This might be due to the fact that traffic patterns and services differ during night time.

Table 4. Percentage of M1 and M2 measurement errors in bidirectional flows captured in different time periods. N/A indicates that no trace file has been analyzed in this time period.

	[12am – 6am)		[6am – 12pm)		[12pm – 18pm)		[18pm – 12am)	
	M1	**M2**	**M1**	**M2**	**M1**	**M2**	**M1**	**M2**
GigaSUNET	0.02%	0.04%	0.01%	0.04%	0.01%	0.05%	0.03%	0.06%
OptoSUNET1	0.007%	0.02%	0.006%	0.03%	0.018%	0.07%	0.015%	0.07%
OptoSUNET2	N/A	N/A	N/A	N/A	0.38%	0.1%	N/A	N/A

7 Conclusions

In this study more than 400 traces were analyzed in order to evaluate the quality of packet-level traces collected from Internet backbone traffic using Endace DAG cards. The quality of captured traffic is important for systems such as firewalls and intrusion detection/prevention systems which make decisions based on the captured network traffic. Missing an attack's signature packet by these systems might result in an incorrect or false negative decision.

We have proposed two TCP-based metrics, M1 and M2 which detect missing packets in TCP flows in the collected traces, and we believe that these metrics are good indicators of the quality of the collected data.

The first metric, M1, detects packets with an ACK number greater than the expected ACK number, something that indicates that the end-nodes have acknowledged packets that are not present in the traces. The reasons for this behavior is either that the measurement system has dropped the packet, or that the packet was actually not present on this link due to routing decisions along the end-to-end route. Even though M1 is a powerful metric, it cannot detect missing zero-length packets i.e. packets containing no data such as keep-alive packets, since ACKs are cumulative in their nature, see sub-section 4.1.

The second metric, M2, deals with observing data beyond the receiver's advertised window size, which corresponds to the situation where the packet advertising the receiver's correct window size has been dropped by the measurement node. M2 measurement errors may also be triggered by implementation problems of TCP/IP stacks or nonstandard TCP implementations sending packets outside the advertised window, possibly with the hope of gaining better performance. As opposed to M1, M2 can also detect missing zero-length packets.

Nearly 0.08 and 0.2 percent of the bidirectional flows were considered as erroneous by M1 and M2, respectively. While there was only minor differences between the results of the GigaSUNET and OptoSUNET1 campaigns, OptoSUNET2 showed slightly different results. After evaluating the erroneous IP addresses, we realized that a small percentage of IP addresses have been observed in many, sometimes even in all, of the erroneous flows. We also showed that the numbers of measurement errors are rather similar regardless of at what time the traces were collected.

The result of our study showed that a considerable number of flows had missing packets. Even though the source of missing packets is not clear, they can affect the correctness of the decisions made by firewalls or intrusion detection/prevention systems. This is especially valid for Internet backbone links where huge amount of network traffic is transmitted in a second.

In order to do the analysis of the traces, we have developed a tool called *flowstat* which is capable of analyzing captured trace files. Depending on the specifications of the computer system and link speed, *flowstat* is also capable of analyzing real-time traffic.

Limitation. There are some limitations which influence our methodology. First, the possibility of asymmetrical routing may cause different packets of a flow to be routed through different links. Second, due to the possible improper implementations of TCP/IP stack from end points, the second metric is not as accurate as the first one.

Future Work. In this paper we have shown that packet losses are present in all our collected traces, regardless of when, where and with what hardware they have been collected. We have also given some reasons for packet loss, but more work is needed to investigate the sources for errors and to find out why and to what degree they contribute. The systems we have used may or may not be representative for many other data collection campaigns, but the overall conclusion must be that it is worth

investigating the quality of the traces using the *flowstat* tool if 100% accuracy is desired. We would also like to encourage the community to use *flowstat* to check other traces taken in other environments (e.g. using a single link to ensure 100% visibility of inbound and outbound traffic) and compare the results with ours.

As discussed in section 6, a small percentage of IP addresses/hosts experienced a large number of packet drops. It would be interesting to investigate these hosts in more detail, for example to use packet headers to find out what operating system they have. These erroneous flows might as well have other common characteristics which can be checked to find yet other reasons behind the missing packets. *Flowstat* can be improved in a number of ways. *Flowstat*'s default behavior is to remove erroneous flows from the connection table as soon as they are detected by any of the metrics. This prevents the metrics from being applied multiple times on the observed flow.

References

1. Wolfgang John, "Characterization and Classification of Internet Backbone Traffic," Chalmers University of Technology, gothenburg, Sweden, PhD Thesis 0346-718X, 2010.
2. Daniela Brauckhoff, Xenofontas Dimitropoulos, Arno Wagner, and Kavè Salamatian, "Anomaly extraction in backbone networks using association rules," in *IMC '09 Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, New York, NY, USA, 2009, pp. 28 - 34.
3. Chuck Fraleigh et al., "Packet-Level Traffic Measurements from the Sprint IP Backbone," *IEEE Network*, vol. 17, no. 6, pp. 6 - 16, December 2003.
4. Fengjun Shang, "Research on the Link Traffic Measurement System Based on Edge Measurement," in *International Conference on Communications, Circuits and Systems*, Guilin, Guangzi, China, 2006, pp. 1791 - 1795.
5. Marco Mellia, Renato Lo Cigno, and Fabio Neri, "Measuring IP and TCP behavior on edge nodes with Tstat," *Computer Networks*, vol. 47, no. 1, pp. 1-21, January 2005.
6. Endace. (2011, September) Enterprise Network Monitoring Tools, Network Security System, Application Performance Monitoring. [Online]. http://www.endace.com/the-endace-platform.html
7. NSS lab, "Network Intrusion Detection System individual product test result," NSS Labs, Auckland, New Zealand, 2010.
8. Vern Paxson, "Automated packet trace analysis of TCP implementations," in *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, Cannes, France, 1997, pp. 167-179.
9. SUNET. (2011, September) History of the Swedish University Computer Network. [Online]. http://basun.sunet.se/karta/
10. SUNET. (2011, September) The Swedish University Computer Network OptoSUNET. [Online]. http://basun.sunet.se/aktuellt/optosunetbroschyr_eng.pdf
11. Wolfgang John, Sven Tafvelin, and Tomas Olovsson, "Passive Internet Measurement Overview and Guidelines Based on Experiences," *Computer Communications*, vol. 33, no. 5, March 2010.
12. Wofgang John and Sven Tafvelin, "Analysis of Internet Backbone Traffic and Header

Anomalies Observed," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, San Diego, 2007, pp. 111 - 116.

13. Ken Keys et al., "The Architecture of CoralReef:An Internet Traffic Monitoring Software Suite," in *a workshop in passive and active measurement*, Amsterdam, The Netherlands, 2001.
14. Jinliang Fan, Jun Jim Xu, Mostafa H. Ammar, and Sue Bok Moon, "Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme," in *ICNP: proceedings of the 10th IEEE International Conference on Network Protocols*, Washington, DC, USA, 2002, pp. 280-289.
15. Feng Qian et al., "TCP revisited: a fresh look at TCP in the wild," in *Proceeding of the 9th ACM SIGCOMM conference on Internet measurement conference*, Chicago, Illinois, 2009, pp. 76-89.
16. Tomas Olovsson and Wolfgang John, "Detection of malicious traffic on backbone links via packet header analysis," *Campus-Wide Information Systems*, vol. 25, no. 5, pp. 342 - 358, 2008.
17. K.C. Claffy, Hans Werner Braun, and George C. Polyzos, "A parameterizable methodology for Internet traffic flow profiling," *Selected areas in communications*, vol. 13, no. 8, pp. 1481 - 1494, October 1995.
18. Wolfgang John, Maurizio Dusi, and K. C. Claffy, "Estimating routing symmetry on single links by passive flow measurements," in *IWCMC '10 Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, Caen, France, 2010, pp. 473 - 478.