

Analysis of Traffic Traces for Stateful Applications

Javier Verdú[‡] Jorge García[‡] Mario Nemirovsky[§] Mateo Valero[‡]

[‡] Departament de Arquitectura de Computadors
Universitat Politècnica de Catalunya
Jordi Girona 1-3, Edifici D-6,
Campus Nord, 08034 Barcelona, Spain
{jverdu, jorge, mateo}@ac.upc.es

[§] Tidal Networks Inc.
697 River Oaks Pkwy
San Jose, CA 95134, USA
mario@tidalnetworks.net

Abstract

Traffic traces are important for many kinds of network research. For reasons of confidentiality, however, publicly available traffic traces are normally anonymized. This stripping of information discards vital properties of the captured traffic, rendering the traces unusable for certain network studies.

Stateful networking applications are an emerging class of applications in the Network Processors area. The packet processing procedure for this class of applications differs significantly from that for stateless applications; thus, different sets of traffic properties are required for representative packet trace analysis.

In this paper we study the differences in results between analysis based on sanitized traces and real traces processed by Snort, as an example of a stateful networking application. Our results demonstrate that there is, in fact, no significant difference. In addition, we analyze the impact of flow locality on the memory workload generated by Snort processing.

1. Introduction

In recent years the exponential growth of Internet and fiber optic technologies has been even higher than the exponential growth predicted for integrated circuits by Moore's law. This increase involves broader network bandwidth through the links. In addition, more complex applications are being applied to provide a better intelligence to the network packet processing.

Networking applications can be classified according to the packet processing procedure: in *stateless applications*, packets are processed relying on tables that change infrequently (e.g. IP forwarding, packet classification), whereas in *stateful applications*, the state tables suffer much more frequent changes as they are modified by the processing of the packets themselves (e.g. intrusion detection, monitoring).

A current trend in router design is the use of Network Processors, high-performance, programmable devices designed to efficiently execute networking applications [2]. One of the problems that appears in the design of Network Processors is the lack of representative Internet traffic traces. Traffic traces from several representative links are pub-

licly available in diverse sites (e.g. NLANR [9] or CAIDA [3]). However, due to confidentiality reasons, the IP addresses of the packets of public traffic traces must be modified through a specific method, called sanitization process, which modifies the Internet IP address distribution [5]. A new approach to transform and anonymize packet traces is proposed by Pang et. al. [11] in order to improve this transformation. Sanitized traces can be useful for some study purposes (for example, sanitization does not modify temporal locality properties). However, when the packet processing depends on IP address distribution, such as IP forwarding, using sanitized traces can lead to erroneous results. The question that comes out is whether the sanitized traces can be used as representative traffic in studies of stateful networking applications.

In this paper we study the differences between the use of sanitized and real traffic traces processed by Snort, as an example of stateful networking application, focusing on memory access characteristics. The main result we obtain is that sanitized traffic traces does not have significant differences in the memory access behavior. Additionally, we analyze the some traffic aggregation characteristics related to the temporal and spatial locality of the flow. Finally, we study the flow localities impact on the behavior of memory workload generated by Snort.

The remainder of this paper is organized as follows. Section 2 we explain the benchmark selection and the methodology used for obtaining the results to analyze. In Section 3 results are studied and discussed. Finally we summarize with conclusions in Section 4.

2. Experimental Methodology

In this section we describe experimental methodology used in the paper.

2.1. Measurement

There is a reduced number of benchmarks suites for Network Processors (NetBench[7], Commbench[15], NP Forum[4], NpBench[6]). However only one benchmark of these suites

(Snort[14]) belongs to stateful application category. In this paper we use Snort as a representative application of this category.

Snort is a Network Intrusion Detection System capable of real-time traffic monitoring, packet logging and detecting unauthorized use of or attacks on a network [1]. The packet processing can be divided into three stages: rules matching of packet header, pattern matching of packet payload and preprocessing of the whole packet. However, the statefulness of the application resides on determined preprocessing engines. We configure Snort in order to enable these preprocessors.

Our major concern is to analyze the impact of IP addresses on the performance of the application. The IP dependency is reflected in the cost of updating state related to each processed packet. The states are kept in SplayTree structures. A SplayTree [12] is a self adjusting form of a binary search tree. The tree is autobalanced every time it is accessed, involving that the last accessed node is the new root of the tree. Therefore, the upper nodes of the tree are the most recent accessed nodes. According to the binary search, the sorting of nodes depends on the values of source/destination ports and IP addresses associated to the flow. Hence, the method of node rotation involves different distributions depending on the sorting values, although this difference is not significative according to the cost of node searching. Instead, when memory has to be released, the selection of the victim node is determined by the lower nodes distribution. Therefore, the selection of a node in order to be released is IP address dependent. The influence of releasing a different node is reflected on an extra workload due to inserting the node again. However, if the node is not released, no extra work is needed.

The Snort's binary code was instrumented using the ATOM tool [13]. In order to delimit the processing of packets, checkpoints were placed at the beginning and at the end of the packet processing. The instrumented code records the number of memory accesses performed by each packet. At the end of the traffic trace processing, a list including the total number of memory accesses per packet was generated.

RT	SN1	SN2	SN3
192.161.35.48	10.0.0.1	10.0.0.1	147.83.53.35
145.32.148.23	10.0.0.2	10.0.1.1	158.109.7.38
192.161.35.48	10.0.0.1	10.0.0.1	147.83.53.35
137.239.42.12	10.0.0.3	10.1.1.1	146.19.5.248
195.232.62.17	10.0.0.4	11.1.1.1	65.167.55.24
131.25.105.36	10.0.0.5	11.1.1.2	211.243.8.86

Table 1. Examples of Sanitization Methods

2.2. Traffic Traces

In order to assess the impact of the IP address distribution on memory accesses during the packet processing, we compare a real traffic trace versus several sanitized versions of the same trace. The sanitization methods we used are summarized in Table 1. In SN1, addresses were obtained by using a counter that increments the last octet of the IP address every time a new address is found. In SN2, the method is the same as SN1, but changing the incremented octet. Finally, in SN3 we use random IP addresses.

The packet trace we used were taken from a OC-3 link (155 Mbps) that connects the Scientific Ring of Catalonia [10] to RedIRIS [8], the Spanish national research network. The Scientific Ring of Catalonia is composed of around 40 institutions (research centres and other organizations) connected through roughly 25 access points.

We also used a number of sanitized traces from other links with different bandwidth (see Table 2), obtained from the NLANR site [9].

Label	Site	Link	Mbps
ANL	Argonne National Laboratory	OC-3	155
MRA	Merit Abilene	OC-12	620
IPLS	Abilene IPLS router instrumentation	OC-48	2480

Table 2. Traffic Traces

2.3. Comparison Methodology

In order to assess the influence of different sanitization methods on the memory access characteristics, we compare the number of accesses realized for packets of different traces.

However, as we mentioned in Section 2.1, the behavior of SplayTree structures may involve spurious large differences, due to the workload generated in shifted packets of the same flow. Therefore, in order to be able to cover these situations, we also compare the number of memory accesses per flow. The same methodology as before is used for showing the impact of the difference.

Additionally, for studying traffic aggregation properties related to the flow localities, we perform the following measurements. In order to measure the *flow temporal locality*, the expected value of the distance between two packets of the same flow is calculated. Additionally, the *flow spatial locality* is determined by the flow variation degree between two packets of the same flow. The flow variation degree ranges from 0 to 1. 0 means that every packet belongs to the same flow and 1 means that every packet belongs to a different flow.

3. Evaluation and Results

3.1. Memory Access Measurements

Before memory access measurements were taken, the application was warmed up with the processing of 100,000 packets. At this point, we started registering the number of memory accesses per packet. We used 500,000 packets as a representative length of the evaluation period. Analyses were also performed with other number of packets and no significant variations were observed. In the case of memory accesses comparisons generated by flows, we have detected all the packets that constitute the flow, restricting our analysis to TCP traffic. Thus flows always started with a SYN TCP segment opening the connection packet and include the last closing packets. We used 1,000 complete flows as a representa-

tive evaluation length using the same verifying methodology.

Figure 1 shows an example of the results obtained both for the original and for a sanitized packet trace. We observe that for most packets we obtain similar number of memory accesses, although for some packets there are huge differences. Nevertheless, the number of packets that generate large workloads is very similar, though these workloads are associated to the processing of different packets. For example, the Figure 1 shows that both traces have two packets that generate a long number of memory accesses. However, there are two huge differences, because of the second packet that generates this important workload is not the same in both traces.

In this section we show the impact of the memory access differences. As we mentioned in Section 2.3, we measure the number of memory accesses of every trace. The difference between the workload generated by a packet of the original trace and the sanitized traces is calculated. This difference is converted into a percentage in order to measure the impact from the point of view of the workload generated by the packet of the original trace.

Figure 2 plots the ranges of memory access dif-

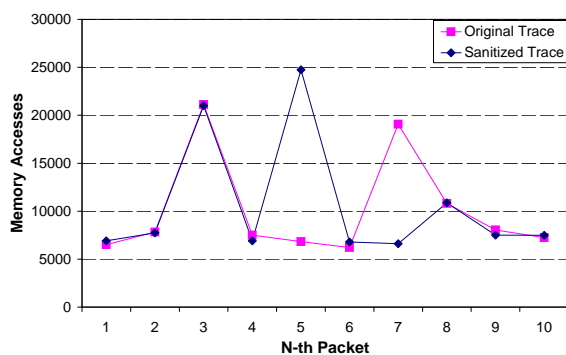
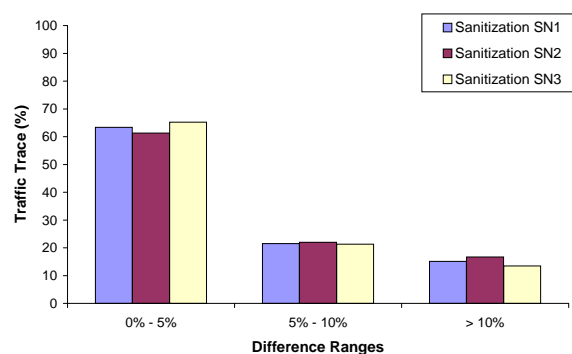
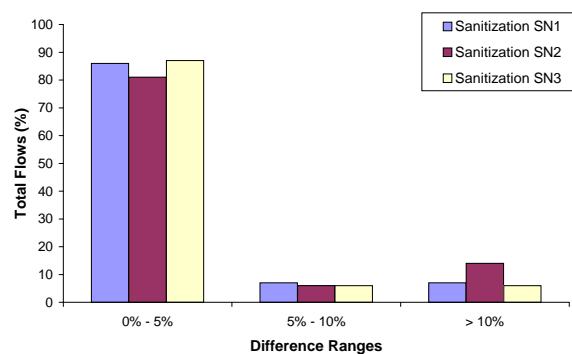


Figure 1. Example of workloads generated by the packet



(a) Packet based comparison



(b) Flow based comparison

Figure 2. Differences of memory accesses against real traffic trace

ferences of the sanitized traces against the real traffic trace. In the left graph we can see that about 65% of the packets show a lower difference than 5% of the memory accesses done by the real trace, which correspond to the packets with a very similar behavior. Around 20% of the packets show a difference between 5% and 10%, which is associated to the freeing memory that belongs to nodes that may vary depending of the tree structure dis-

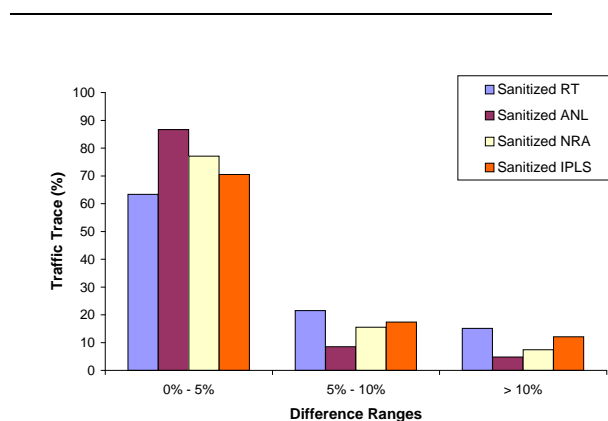
tribution (Section 2.1). Finally, around 15% of the packets show larger differences. These differences are due to the fact that in one trace memory needs to be released, whereas in the other trace memory is still available. In the right graph we can see a similar behavior, but the differences are reduced, because a significant percentage of the packets that generate the huge differences above commented are collected into the flow that they are associated. Additionally, both graphs show there are no significant differences between traces generated with different sanitizing methods.

The same comparison has been applied to traces from different bandwidth links (see Table 2). The Figure 3 shows that the variations are different between traces from other links, because their original traffic has been sanitized with diverse methods. Due to this, the IP address distribution is more similar to the compared traces. Additionally, we can see that the increment of the bandwidth has no impact on the results.

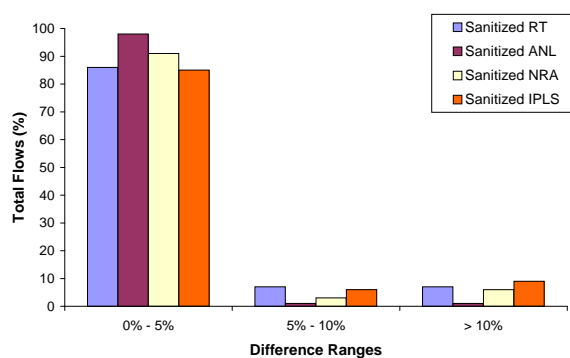
3.2. Traffic Aggregation

As we mentioned in Section 2.2, the flow temporal locality is defined by the distance between two packets of the same flow. The metric system used for measuring the distance is the number of packets. The average distances are shown in the left graph of the Figure 4. The results involves than the flow temporal locality is reduced with the growth of traffic bandwidth, which is the same repercussion cache temporal locality suffers.

On the other hand, flow spatial locality is determined by the flow variation degree between two packets of the same flow. The right graph of Figure 4 shows the increment of the flow variation according to the increment of traffic bandwidth. In this case, the increments are not completely proportional to the growth of bandwidth due to intrinsic characteristics of the traffic. However we can see that a significant number of different flows are found along processing of the packets between two packets of the same flow. The impact of these results involves that, due to the behavior of SplayTree structures, node related to the flow will be reallo-



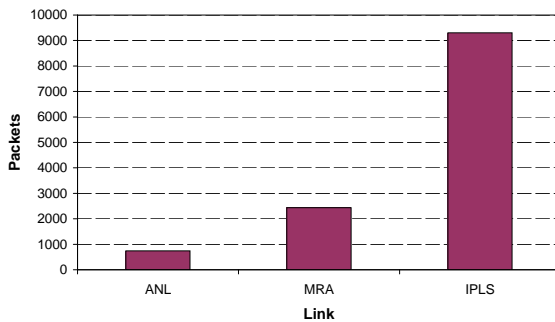
(a) Packet based comparison



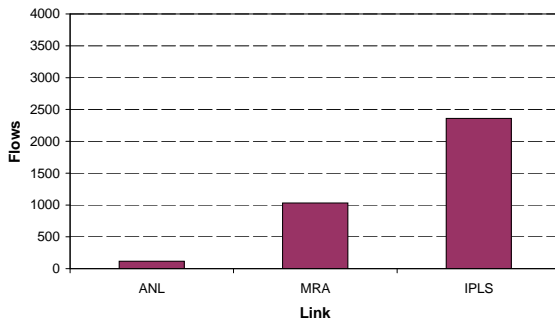
(b) Flow based comparison

Figure 3. Differences of memory accesses against original traffic traces of different bandwidths

cated to lower levels of the tree. Thus, the flow spatial locality is reduced with the growth of traffic bandwidth. In addition, there is a higher probability that the next time that appears a packet of a flow, the node related had been released.



(a) Distance between flow packets



(b) Flow aggregation between flow packets

Figure 4. Traffic properties related to the flow locality

4. Summary

This paper analyzes the differences in results between analysis based on sanitized traces and real traces processed by Snort, as an example of a stateful networking application. The results show that sanitizing packets causes no significant differences in the memory workload behavior generated by packet processing. As future work, we plan to extend the study of the impact of IP address distribu-

tion on other stateful applications, in order to validate sanitized traces to be used for stateful network research.

Additionally, this paper presents a study of the traffic properties that determines the temporal and spatial locality of the flows. We have studied how the flow localities affect the SplayTree structures and on the generated memory workload. The analysis shows that when flow localities are reduced, there is a high probability that the node related to the flow can be placed at the lower levels of the tree or even eliminated.

5. Acknowledgments

This work was supported by the Ministry of Science and Technology of Spain under contract TIC-2001-0995-C02-01, and grant BES-2002-2660 (J. Verdú), and by CEPBA. The authors would like to thank Prof. Solé-Pareta for making available a set of real traffic traces derived from RedIRIS network and Rodolfo Milito for his comments and review inputs of this work.

References

- [1] Jay Beale, James C. Foster, Jeffrey Posluns, and Brian Caswell. *Snort 2.0 Intrusion Detection*. Synpress Publishing Inc., 2003.
- [2] Patrick Crowley, Mark A. Franklin, Haldun Hadimioglu, and Peter Z. Onufryk. *Network Processor Design: Issues and Practices*. Morgan Kaufmann Publishers, San Francisco, CA, 2002.
- [3] Cooperative Association for Internet Data Analysis. <http://www.caida.org>.
- [4] Network Processing Forum. <http://www.npforum.org>.
- [5] Eddie Kohler, Jinyang Li, Vern Paxson, and Scott Shenker. Observed structure of addresses in ip traffic. In *Proceedings of the second ACM SIGCOMM Workshop on Internet measurement workshop*, pages 253–266. ACM Press, 2002.
- [6] Byeong Kil Lee and Lizy Kurian John. Npbench: A benchmark suite for control plane and data plane applications for network processors. In *Proc. of IEEE International Conference on Computer Design: VLSI in Computers & Processors (ICCD)*, pages 226–233, San Jose, CA, October 2003.

- [7] Gokhan Memik, William H. Mangione-Smith, and Wendong Hu. Netbench: A benchmarking suite for network processors. In *IEEE International Conference Computer-Aided Design (ICCAD)*, 2001.
- [8] RedIRIS: Spanish National Research Network. <http://www.rediris.es>.
- [9] National Lab of Applied Network Research. <http://pma.nlanr.net/Traces/>.
- [10] Scientific Ring of Catalonia. <http://www.cesca.es/comunicacions/anella.html>.
- [11] Ruoming Pang and Vern Paxson. A High-Level Programming Environment for Packet Trace Anonymization and Transformation. In *Proceedings of the ACM SIGCOMM Conference*, August 2003.
- [12] Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *Journal of the ACM (JACM)*, 32(3):652–686, 1985.
- [13] Amitabh Srivastava and Alan Eustace. ATOM - A system for building customized program analysis tools. In *Programming Language Design and Implementation (PLDI)*, pages 196–205, June 1994.
- [14] The Open Source Network Intrusion Detection System. <http://www.snort.org>.
- [15] Tilman Wolf and Mark A. Franklin. CommBench - a telecommunications benchmark for network processors. In *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 154–162, Austin, TX, April 2000.