Measuring the Processing Performance of NetSniff

Julie-Anne Bussiere[*], Jason But
Centre for Advanced Internet Architectures. Technical Report 050823A
Swinburne University of Technology
Melbourne, Australia
julie-anne.bussiere@laposte.net, jbut@swin.edu.au

***Abstract*- NetSniff is an IP traffic analysis tool currently used in low traffic scenarios. Before deployment under higher traffic scenarios, it is important to perform a study into the processing and live traffic capture performance of NetSniff. In this technical report we subject NetSniff to a series of processing performance evaluations in an attempt to determine the limitations of NetSniff with regard to packet processing rates on different hardware platforms and configurations.**

***Keywords*- NetSniff, processing performance, Hardware**

I. Introduction

NetSniff is a multi-network-layered real-time traffic capture and analysis tool developed as part of the ICE^3 project being run out of the Center for Advanced Internet Architectures (CAIA). The NetSniff tool is currently deployed in low-bandwidth and low-traffic scenarios. To gather more useful information, we would like to deploy it within networks where the number of aggregate users is higher. Our motivation and goals have been previously highlighted [1]. In this report we investigate the raw processing performance of NetSniff when analysing traffic from a tcpdump capture file on disk. This anlaysis excludes the effects of live capture and produces results which are entirely dependent on the packet processing and analysis rate that the NetSniff implementation can maintain.

II. Data Collection

A RULE based multiple virtual host testbed was constructed [1] which consisted of three machines running FreeBSD, see Fig. 1. The configuration of these machines is also descibed here [1]. Once configured, we can start generating traffic between the jails hosts and server, and recording it to a series of tcpdump files. We created 20 jail hosts, with the `at` unix command, each jail host was configured to start a preconfigured series of TCP applications at a predefined time. Each jail was launched with a lag of a few seconds. All traffic was recorded on the bridge machine using **tcpdump**, since NetSniff requires the entire packet payload to function, tcpdump is executed with the `-s 0` option.

A series of tcpdump files are made, each with a different number of jails generating concurrent traffic flows, using multiple jails enable us to have concurrent flows generated by different hosts with different IP addresses. Processing performance is analysed by running NetSniff using the tcpdump capture files rather than under a live capture scenario.

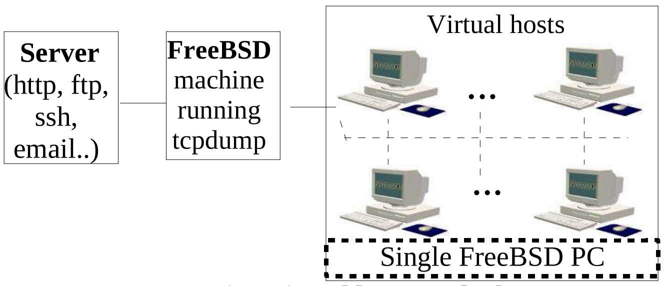

***Fig.1*Virtual hosts testbed**

Tcpdump capture files were recorded consisting of concurrent flows from 1, 5, 10, 15 and 20 different virtual jailed hosts. From these tcpdump files, we can create other traffic files using the **tcpslice** [2], **tcprewrite** [3] and **mergecap** [4] tools. **Tcpslice** can be used to select a portions of a tcpdump file (eg. Extract packets between two specified timestamps). **Tcprewrite** is part of the **tcpreplay** tool, it can modify tcpdump files by, for example, changing IP addresses. **mergecap** is part of **ethereal** tool and can join several tcpdump files with concurrent timestamps. We use these tools to generate tcpdump files consisting of flows from more than 20 different jailed hosts, in our case capture files with flows from 30, 40 and 60 different hosts. We also use these tools to extract three minute segments from each of the original tcpdump file for analysis.

For example, to generate a tcpdump containing flows from 40 unique hosts, we can:

- Duplicate the tcpdump file for 20 unique hosts, using tcprewrite to change the IP addresses.
- Use mergecap to merge the original tcpdump file with the (modified) duplicated one.

In this way we obtain a file with 40 different IP addresses. However, in the 40 hosts file, each packet has its "twin" sent at the exact same timestamp. This does not represent perfectly realistic traffic behavior, but it allows us to examine the processing performance of NetSniff under higher traffic conditions. The following table lists both the size and the number of packets of each of the generated 3 minute tcpdump files we used in the following experiments. The table also indicates the corresponding average bitrate and packets per second (PPS) of the tcpdump files (as calculated using the number of bytes and packets in each file and a three minute duration). These figures can be used to relate results back to measurements that are more typical when referring to network traffic.

[*] Julie-Anne Bussiere performed this work while a visiting research assistant at CAIA in 2005

Number of hosts	*Size (MB)*	*Number of packets*	*Capture rate (Mb/s)*	*Capture PPS*
1	54.9	64535	2.44	358.53
5	271	318761	12.04	1770.89
10	477.4	560611	21.22	3114.51
15	641	751429	28.49	4174.61
20	853.6	1000054	37.94	5555.86
30	1282.2	1502858	56.99	8349.21
40	1707.2	2000108	75.88	11111.71
60	2564	3005716	113.96	16698.42

Table 1 **Traffic files characteristics**

III. Processing performance results

This section contains the results we witnessed when processing the generated tcpdump traffic capture files.

A. Evaluating the "fake" tcpdump file.

We generated some of our tcpdump files, namely those consisting of 30, 40 and 60 different hosts, by duplicating other captured traces. We would like to determine how much of an affect this duplication will have on our results as opposed to actually capturing traffic from a certain number of hosts. We cannot compare against "real" traffic traces with these number of hosts since we did not generate these files, we can however duplicate the tcpdump files consisting of 5 and 10 unique hosts to generate "fake" tcpdump files of 10 and 20 hosts respectively.

When comparing the "real" 10 host tcpdump file with the "fake" 10 host tcpdump file we notice some interesting results:

- The duplicated tcpdump file takes more user time (about 15%) to be processed.
- The observed peak process size for the duplicated trace was smaller, 4572KB as compared to 11988KB. The original unduplicated file (5 hosts) recorded a peak process size of 4224KB. Only slightly smaller than the duplicated file.

Similar results are observed when comparing the two 20 host tcpdump files. In this case the duplicated file has an increased processing user time of 33% while the peak process size remains correlated to the peak process size while processing the unduplicated file.

Most of this increased processing time can be explained by the increased size of the duplicated trace file as compared to the captured trace file of the same size, indeed in both tested cases there is approximately 13% more data to be processed in the duplicated files. This would indicate that the processing rate (MB/s or PPS) would be relatively correct since the increased user time would be offset by the larger tcpdump file size. Further, we must be careful when considering the issue of process size with the duplicated files as the results may not be truly indicative.

B. Test Platforms

Each tcpdump file was processed by NetSniff on four different machines, each with differing configurations (CPU clock speed and RAM). All machine were running the FreeBSD v5.3 Operating System. The following table outlines the details of each machine.

Machine	*Processor type*	*CPU (GHz)*	*RAM (MB)*
Box 1	Pentium 4	2.66	512
Box 2	Pentium 4	2.8	512
Box 3	Celeron	2.4	256
Box 4	Pentium 4	2.66	2048

Table 2 **Machines HW configuration**

C. Processing bit rate

We determine NetSniff's processing bit rate by considering the user time required to process a particular input file, and combining this result with the size of the input file. By dividing the size (MB and packets) by the run time of NetSniff, we can obtain a processing rate in both Mb/s and PPS(packets per second).

The user processing time is obtained with the `time` command, which gives real time, user time and system time. The user time corresponds to the CPU time spent executing instructions of the calling process, while the system time is the CPU time spent in the system while executing tasks on behalf of the calling process. Both include time spent for children processes.

The results for all tcpdump files on all test platforms are shown in figure 2. The immediate result is that the number of concurrent flows, as indicated by the number of unique hosts, has a strong impact on the processing rate. When processing a packet, NetSniff needs to determine to which flow it belongs, and maintain a database of all open, concurrent flows. As the number of concurrent flows increases, this lookup process takes longer. The decrease in processing rates is asymptotic as the number of hosts increases. The effect of the duplicated files is negligable.

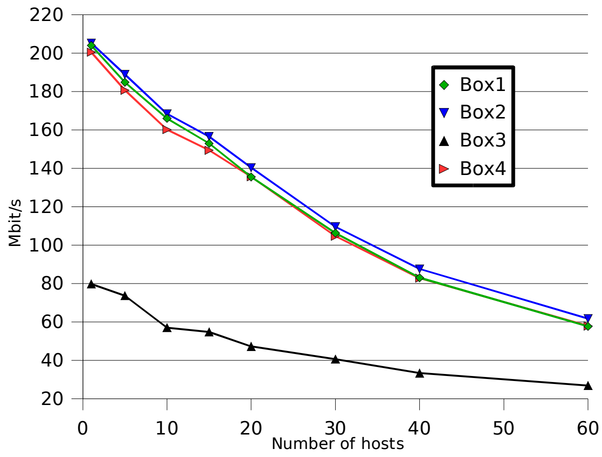

Fig.2 **Processing bit rate versus number of hosts**

The hardware configuration of the test platforms has a non negligible impact, particularly the processor type.

The Pentium4 processor can process packets at more than twice the rate of the Celeron CPU. This could be due in part to the larger cache present on the Pentium processors and also to the better internal design of these CPUs. The system clock speed has a lesser impact as is witnessed with the 2.8GHz Pentium4 versus the two 2.66GHz machines.

Finally, the amount of system memory appears to have little, if no, impact at all – witness the comparison between the two 2.66GHz Pentium4 computers with differing memory sizes. This could be due to the small memory footprint required by NetSniff under the test conditions, we would expect that if the capture device was performing other tasks then system memory, and in particular access to the swap space, would have an impact on the processing rate of NetSniff.

Similar results are observed when plotting the PPS versus the number of hosts. The processing packet rate for the fastest test machines decrease from 30000 pps for 1 host to 8450 pps for 60 hosts. The Celeron test platform processes data at a packet rate of 11750 pps for 1 host to 3950 pps for 60.

D. *Processing speed*

We define the processing speed as the ratio of the real recording time (timespan of the timestamps within the tcpdump file) and the measured processing user time. When this ratio is greater than 1, then NetSniff is processing packets at a rate greater than that at which the packets nominally arrive at the capture point. The crossover point indicates the maximum rate at which NetSniff can be expected to capture and process packets in real time.

These results are plotted in Figure 3, an extra curve is plotted with an adjusted ratio based on previously measured results with the duplicated files. The x-axis indicates the average bitrate of the tcpdump files under consideration.

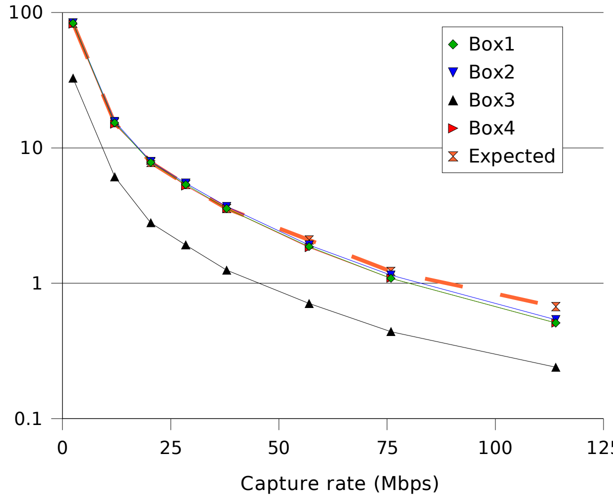

Fig.3 **Processing speed ratio versus capture bit rate (logarithmic scale)**

The predominant result is that for the Celeron based test platform the crossover point occurs at about 45Mb/s while the Pentium4 based machines experience this point at about 80Mb/s. These figures indicate the fastest average bitrate at which these test platforms could capture and process data in real time.

E. *CPU usage*

We next consider CPU usage while running NetSniff with each of the input tcpdump files. We use the `top` command to obtain information on the process CPU use and process size at one second intervals. These results are collated and averaged for the duration of the NetSniff run time.

Figure 4 shows the mean value of the WCPU (weighted CPU) variable on each test platform for each tcpdump file, where WCPU is a decaying average over up to a minute of previous (real) time and indicates the CPU resources used by the process.

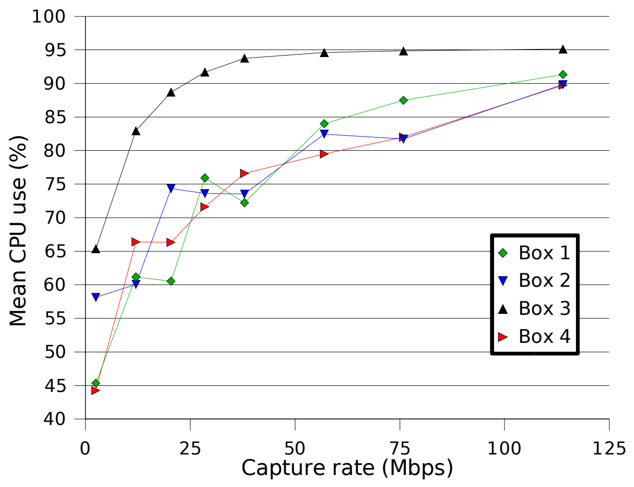

Fig.4 **Processing mean CPU use versus live capture bit rate for each box**

The mean CPU use increases as the amount of processing performed by NetSniff increases. It can be argued that for the smaller tcpdump files that processing time was so quick as to not require the assignment of excess CPU cycles to the process. When considering the processing of data at the processing speed crossover point (~80Mb/s on the Pentium4 platforms), we not that CPU usage is in the range 80-90%. While this can be considered high, we would expect that a purpose built box deployed to capture and analyse traffic would not be running other concurrent applications.

It should also be noticed that at lower average capture rates the CPU usage should be downgraded as our test conditions are performed with NetSniff processing all packets as quickly as possible rather than as they arrive. In these instances NetSniff will spend more time idle waiting for packets to arrive and thus lower the CPU usage figures.

Figure 5 shows CPU usage weighted by tcpdump file duration. It is obtained by dividing the CPU usage percentage by the speed ratio. These results are more indicative of NetSniff's CPU requirements during live capture. The points at which CPU usage nears and exceeds 100% indicates at which data rate we can expect NetSniff to not be able to process captured packets in real time (~80Mb/s on Pentium4, ~45Mb/s on Celeron).

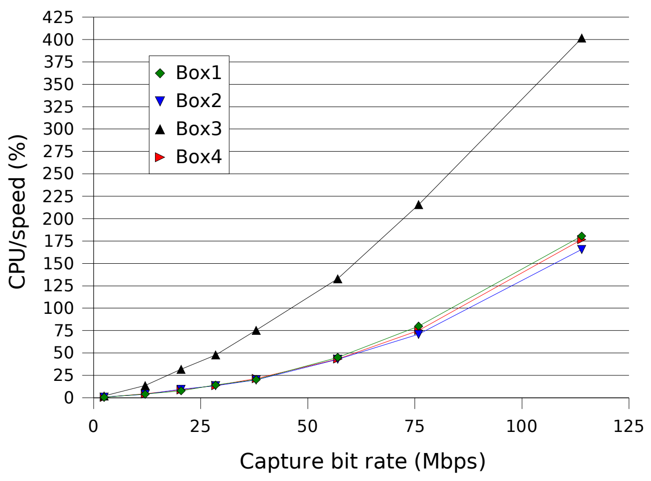

Fig.5 **CPU usage weighted by time over capture bit rate**

F. *Process Size*

Another aspect to consider is the process size, or system memory resources required by NetSniff to process the packets. Again we can use the `top` command to obtain information on the process size at one second intervals.

Figure 6 shows the mean and peak process sizes while processing each tcpdump file. As expected, the results are equal on all test platforms as the same application (NetSniff) is processing the same datafile. In all cases where “real” tcpdump files were used – up to 20 hosts – the process size appears to be increasing. Where the “fake” tcpdump files are used, memory usage and process size is significantly lower than expected.

An expected curve is plotted using a polynomial model for process size however it should be noted that:

- There are not enough data points to properly confirm the polynomial model.
- As the number of hosts increase the bitrates get extremely high, since all data was captured on a Fast Ethernet (100Mb/s) card, these figures may not be reliable.

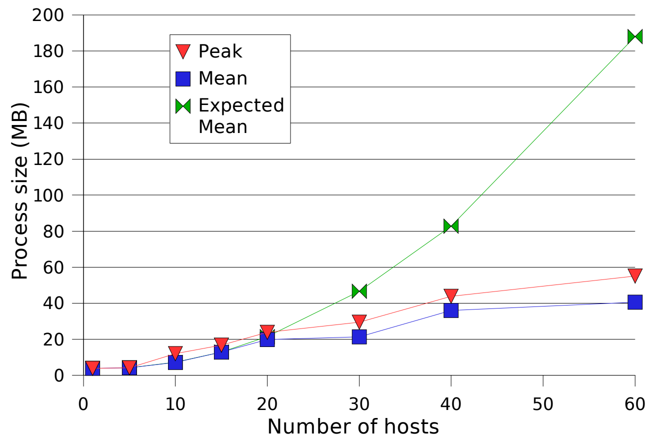

Fig.6 **Process size versus number of hosts**

Even so, the witnessed memory use and process size falls within the range of the amount of system memory that most modern computers have installed, and that a recommendation of 512MB or 1GB on the capture machine would ensure that enough system memory is available to minimise the use of system swap space.

G. *Anonymisation impact*

NetSniff provides three different anonymisation algorithms [5]. These algorithms are used to protect the privacy of the network users whose data is being collected. NetSniff was executed with the default anonymisation algorithm (tcpdpriv – as implemented by the NetSniff developers), the cryptopan mode and the NullIP mode to determine its effect on the processing rate that NetSniff can achieve.

The results for NullIP and tcpdpriv style anonymisation indicate that on the Pentium4 test platforms, processing user time increases by on average 1.3% when anonymising traffic while the Celeron based platform experienced an increase in the user time of about 2%. CPU usage and mean process size are not significantly affected by anonymisation. However, when considering anonymisation using the cryptopan mode [5], we noticed a significant increase in the user processing time.

IV. Conclusion and future work

In this paper we have run NetSniff to process a number of different tcpdump capture files on a variety of different hardware platforms. The results can be used as indicative of the raw packet processing performance of NetSniff without regard to live packet capture issues. We generated the tcpdump files by running real networked applications on a small network testbed running numerous virtual workstations. The result is that “real” traffic was generated in a controlled scenario while using minimal resources.

The resultant files were then trimmed to a certain size and in some cases duplicated to obtain tcpdump files containing flows from numerous different hosts. The summary of our results are:

- Processing performance decreases as the number of active concurrent flows increases.
- The CPU type has a significant impact on processing performance with a Pentium4 based system being able to process data more than twice as quickly as a Celeron based system.
- CPU clock speed has a minimal impact on processing performance.
- System memory has a negligable impact on processing performance.
- The Pentium4 based test platforms were observed to be able to process captured data at virtual real-time at average bitrates of up to about 80Mb/s.
- NetSniff data anonymisation has negligable impact on its processing performance.

Future research should focus on testing NetSniff performance in a live capture scenario. Results in this paper indicate the raw packet processing performance that NetSniff can achieve without regard to the issues of live capture. Consideration of this situation will investigate the performance of packet capture using the PCAP library that NetSniff employs and, in conjuction with the results from this paper, would indicate the expected performance of NetSniff as a tool to capture and analyse network traffic in real-time.

References

[1] J.Bussiere, J.But, "Measuring the performance of Netsniff: Testbed design".

[2] TCPSlice, http://www.die.net/doc/linux/man/man8/tcpslice.8.html, accessed June 2005

[3] TCPReplay, http://tcpreplay.sourceforge.net/man/tcprewrite.html, accessed June 2005

[4] Ethereal, http://www.ethereal.com/docs/man-pages/mergecap.1.html, accessed June 2005

[5] NetSniff – Anonymisation features and functionality, http://caia.swin.edu.au/ice/tools/netsniff/anonymisation.html, accessed June 2005