Measuring the Live Capture Performance of NetSniff¤

Julie-Anne Bussiere*, Jason But
Centre for Advanced Internet Architectures. Technical Report 051004A
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 capture performance of NetSniff. We have previously investigated the processing performance of NetSniff, in this technical report we subject NetSniff to a performance evaluation with regard to live capture of network traffic. We show the impact of increasing the captured traffic rate and in increasing the number of concurrent flows for NetSniff (release version v050722) to process on differing hardware configurations. Our results also indicate that the small PCAP (version 0.9.4) buffer (32kB) on a FreeBSD (version 5.3) based system limits the processing performance of NetSniff under high-bandwidth scenarios, while the Linux (kernel version 2.6) based PCAP library passes packets to NetSniff in non-chronological order – posing further problems in correctly determining TCP layer statistics.**

***Keywords*- NetSniff, live capture, performance.**

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], further we have also previously investigated the raw packet processing performance limitations of NetSniff when processing network traffic from a stored traffic dump file [2]. In this report we will continue our evaluation of NetSniff in the context of its performance in live network traffic capture and analysis.

All evaluation was performed on systems configured running either a FreeBSD 5.3 or a Linux 2.6 based kernel. Each system used version 0.9.4 of the PCAP capture library [4] and version v050722 of the NetSniff application [5]. All software is configured to run with default configuration settings to better simulate the scenario under which a naïve user might operate.

The report is structured as follows: In section two we discuss the means by which we implement and measure live capture performance. In section three we present the results of our live capture performance experiments. We first focused on how NetSniff handles high packet rates, then analysed the performance when dealing with concurrent flows and described hardware or machine configuration impact on performance. The two final sections give a global analysis, taking in account processing performance results obtained in [2], and suggest improvements for better NetSniff performance.

II. Live capture testbed

To perform live capture measurements, the **tcpreplay** [3] tool (version 2.2.2) was used to replay previously recorded traffic across a network. A second workstation running NetSniff is used to capture the generated traffic. **Tcpreplay** allows the user to set the packet rate at which to replay the traffic. NetSniff live capture performance was tested on 2 workstations. The first workstation is a Pentium4 based computer running at a clock speed of 2.66GHz with 512MB of system RAM, the NIC on this computer is an Intel Pro Gigabit Ethernet device. The second workstation is also a Pentium4 based machine running at 2.66GHz, instead installed with 2GB of system RAM and a Broadcom BCM5782 Gigabit Ethernet device. Both machines are configured with FreeBSD 5.3.

The traffic dumps replayed using **tcpreplay** were those generated during processing performance testing [2]. To analyse the packet rate performance of NetSniff, we use the dump file consisting of traffic generated by a single host that was continuously running sequential http, smtp, ftp, ssh and https sessions. This dump file consists of an average packet rate of 350 pkts/sec. Using **tcpreplay** we can increase this rate to measure NetSniff's capture performance with higher packet rates, independent of the number of concurrent flows.

Following this, NetSniff's live capture performance of concurrent flows will be evaluated using traffic dump files consisting of traffic generated by multiple hosts.

Of particular interest during testing are the number of packets dropped (defined as packets that NetSniff has not parsed and analysed because of a buffer overflow in the PCAP library) by NetSniff, as well as system metrics including memory and CPU usage.

¤ NetSniff version release v050722

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

When replaying dump files with a predefined packet rate, **tcpreplay** does not respect the timestamps and packet inter-arrival times stored in the traffic dump file [3]. Previous issues of several packets with equal timestamps in duplicated traffic files [2], are no longer relevant during live capture analysis. The average packet size for our test dump files is 870 bytes.

Since NetSniff reconstructs complete TCP flows to parse application layer statistics, it needs to capture and analyse all packets on the network, a packet dropped rate of 0% is desired. Missing packets will cause incorrect statistics to be calculated for the TCP flow under consideration, if too many packets are dropped prior to analysis then NetSniff will not be able to produce meaningful statistics for any TCP flows. The percentage of dropped packets we are willing to accommodate depends on the number of individual flows we are analysing, the average number of packets that make up these flows, and the number of flows we are willing to have incorrect statistics for.

III. Experimental Results

All experiments were performed with NetSniff version 050722, and PCAP capture library version 0.9.4.

A. High traffic impact

We replayed traffic generated by one host at different packet rates to analyse NetSniff capture and parsing performance. Figure 1 shows the percentage of packets dropped by NetSniff as a function of the programmed packet rate when captured on workstation 2. Similar results were seen when capturing on workstation 1.

At rates of up to 3000 pps, NetSniff does not drop any packets. For rates up to 5000pps the percentage of dropped packets is lower than 0.5%. With these network conditions, NetSniff is reliably capturing and analysing the traffic presented on the network card. Beyond 5000pps, the percentage of dropped packets increases quickly. The bandwidth usage corresponding to 5000 pps is 33.4 Mbps.

This leads to a conclusion that on this platform, NetSniff can be used to capture and analyse traffic at rates of up to 5000pps.

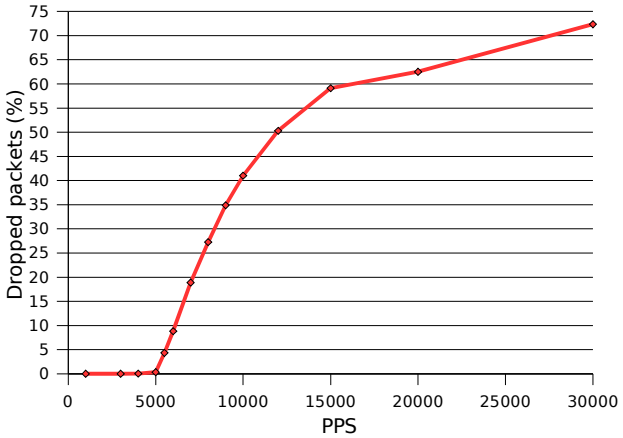

***Fig. 1* Percentage of dropped packets versus packet rate for 1 host traffic**

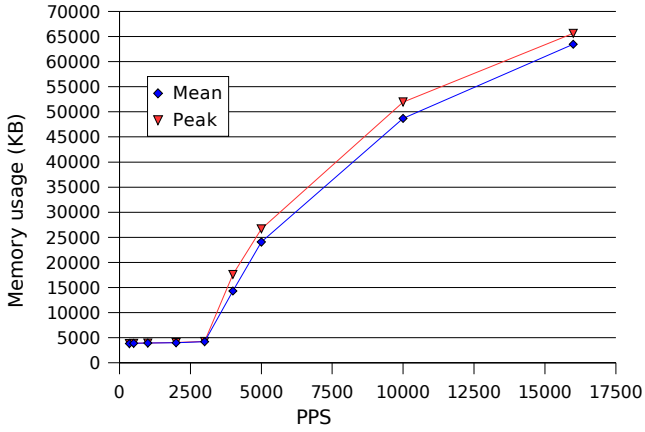

***Fig. 2* Mean and peak memory usage versus packet rate for 1 host traffic**

The dropped packet performance is compared against tcpdump. Both tcpdump and NetSniff use the PCAP library to capture traffic and pass it to the application for processing. Tcpdump was executed in "write-to-disk" mode, where each captured packet was immediately saved to a file with no post-processing. The results allow us to exclude the effect of the PCAP capture library from our capture performance tests. Under the same traffic replay scenario described above, the percentage of dropped packets exhibited by tcpdump remained under 0.5% for packet rates up to 20,000 pps.

These results imply that for packet rates above 5,000 pps, there is an issue in the implementation of parsing the captured packets in NetSniff such that a high proportion of captured packets are dropped, and as a result, not analysed.

Like tcpdump, NetSniff is implemented as a single thread - each packet is captured, parsed and analysed in turn. The PCAP library places incoming packets into a buffer while it waits for the capture application to complete processing of the previous packet. If the application takes too long to process these packets, or a burst of packets arrive at the network card, then the buffer fills while processing occurs. If the processing time is too long, the (fixed size) buffer overflows and packets are dropped, this can occur regardless of whether the average packet processing rate can cope with the number of incoming packets. An increased packet arrival rate has the side effect of filling the buffer more quickly, and therefore increasing the probability of dropping packets. Since tcpdump does little (or no) post-processing, buffer overflow is minimized.

Figure 2 shows the memory usage of NetSniff while capturing the replayed traffic stream at different programmed packet rates. From this, we can see that memory usage is highly correlated to the percentage of dropped packets. In all instances where no packets were dropped, memory usage does not exceed 5MB. As packets are dropped, NetSniff's memory requirements increase. This is likely due to NetSniff re-assembling TCP streams. When a packet from a TCP stream is dropped, NetSniff buffers all subsequent packets from that stream while it waits to see if the missing packet will be retransmitted. As the proportion of dropped

packets increase, the number of TCP streams affected will also increase, therefore increasing NetSniff's buffering requirements.

This extra memory usage is capped, since NetSniff will eventually free these resources when it determines that a TCP stream has timed-out. Also, the peak memory usage at 16,000 pps (at which rate 60% of incoming packets are dropped) is only 65 MB. This indicates that even under extreme packet loss, excess memory usage is limited and available system RAM does not constitute a hardware limitation for high rates of traffic generated by one host. Further, we do not recommend using NetSniff at packet dropped rates greater than 1%.

Figure 3 shows CPU usage by NetSniff while capturing the replayed traffic stream at different programmed packet rates. We note that the CPU requirements are not correlated to the dropped packet rate, but instead to the packet arrival rate. These results were measured on workstation 2, with similar results gathered on workstation 1 (with the same processor and clock speed). As previously measured [2], CPU use is expected to be higher for slower machines. In this instance, the number of dropped packets (and subsequent non-parsing of these packets) means that CPU utilisation remains low.

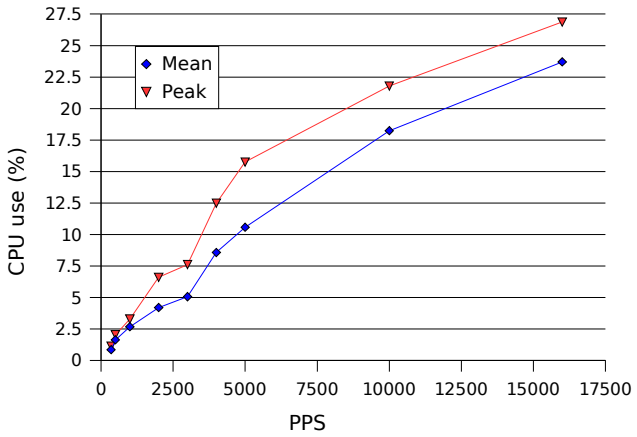

***Fig. 3* Mean and peak CPU use versus packet rate for 1 host traffic**

This result signifies that while NetSniff processing performance can cope with this - and higher – packet arrival rates, there are other factors limiting the rate at which NetSniff can properly capture traffic. In particular it appears that the limited PCAP buffer size (default 32kB) and its subsequent overflow, plays a major part in limiting NetSniff performance.

While the PCAP capture library is an open source product that can be freely modified to provide increased buffer capacity, the experiments performed here are specifically aimed at measuring the performance of NetSniff under default system configurations.

B. Impact of Concurrent Flows

We next replayed the traffic dump files consisting of concurrent flows generated by multiple hosts [2], these dump files were also replayed at different packet rates. Importantly, see Figure 4, the percentage of packets dropped by NetSniff does not significantly increase at a given packet rate with an increased number of concurrent flows. There is still PCAP buffer overflow, and it occurs a slightly lower packet rates as the number of concurrent flows increase, but this increase in dropped packets is minimal.

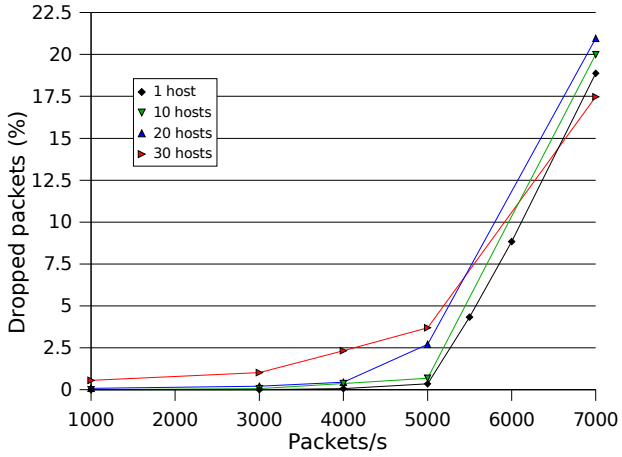

***Fig. 4* Percentage of dropped packets for different numbers of concurrent hosts**

On the other hand, increasing the number of concurrent flows does have an impact on NetSniff's process size (memory usage) and CPU usage. Figure 5 shows NetSniff's mean and peak memory usage while capturing the different trace files replayed at a programmed rate of 1,000 packets per second (12 Mb/s), a rate at which NetSniff drops minimal packets. Results show a peak usage of 5MB for 1 host against 70MB for 60 hosts. This increase is not related to the dropped packets rate (section III.A), but rather to the increased number of flows being reconstructed in parallel [2].

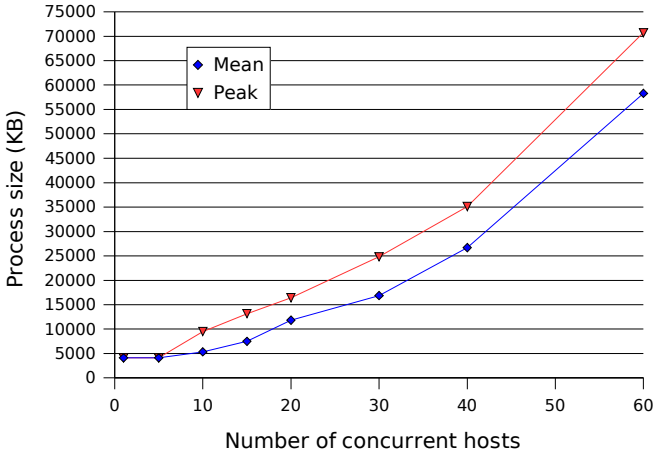

***Fig. 5* Mean and peak memory usage versus number of concurrent hosts for 1000 pps**

Figure 6 shows mean and peak CPU usage for the same conditions described above. The peak value reaches 18% for 60 hosts, but on average does not exceed 10 %. Again NetSniff is limited more by the PCAP buffer than by its actual packet processing capabilities [2].

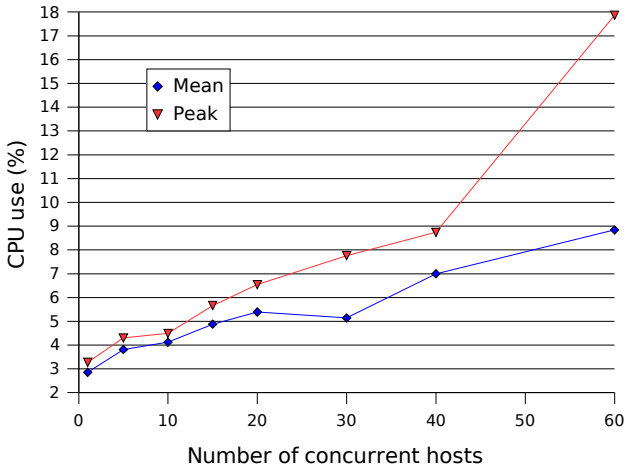

***Fig. 6* Mean and peak CPU use versus number of concurrent hosts for 1000 pps**

C. NetSniff Application Parser Bugs

During performance testing of live capture with multiple concurrent streams, we discovered a series of bugs in one, or more, of the TCP application layer parsers, which caused NetSniff to crash. The bug has been isolated as being caused by reading beyond the end of the reconstructed application data buffers and has been scheduled to be fixed in the near future.

NetSniff was recompiled minus these application parsers for further tests. All TCP flows are still reconstructed, however their statistics are now logged to a generic TCP stream output rather than to application specific logfiles. Disabling the application parsers had no impact on the percentage of dropped packets. The following figures show CPU and memory usage of NetSniff running without application parsers, capturing traffic at a rate of 1000 pps, and comparing with previous results.

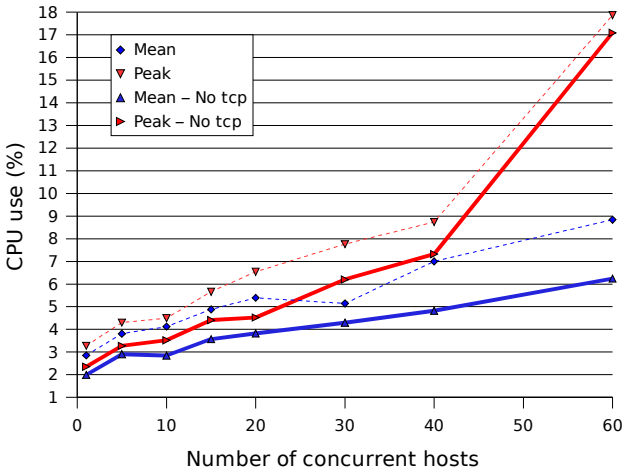

***Fig. 7* CPU usage without TCP application parsers**

In figures 7 and 8, previous results appear in fine dashed to compare with newer results (called "No TCP" on the graph). The CPU usage is on average 2% lower (due to less processing taking place), but the rate of increase shows that CPU requirements are still directly related to the number of concurrent flows. Similarly, memory usage peak values are slightly lower. In summary, the TCP application parsers do not greatly contribute to the system CPU and memory resource requirements for NetSniff.

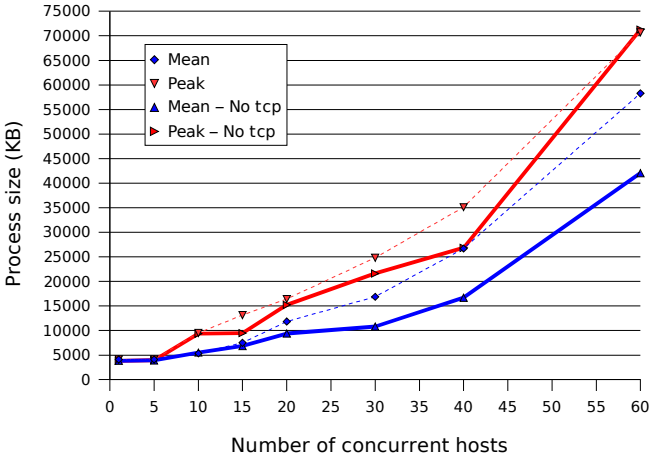

***Fig. 8* Memory usage without TCP application parsers**

In figures 7 and 8, previous results appear in fine dashed to compare with newer results (called "No TCP" on the graph). The CPU usage is on average 2% lower (due to less processing taking place), but the rate of increase shows that CPU requirements are still directly related to the number of concurrent flows. Similarly, memory usage peak values are slightly lower. In summary, the TCP application parsers do not greatly contribute to the system CPU and memory resource requirements for NetSniff.

D. Hardware impact

We witnessed no significant difference in performance on the two systems with varying system RAM and NICs. Previous results [2] would indicate that the processor type (Pentium4 vs Celeron) would have a major impact on the processing performance with clock speed contributing a little to performance. We also previously concluded that the impact of system memory was minimal.

The two network cards under consideration also contributed little to the performance of capturing and analysing data in real-time with NetSniff. Both network cards are typical consumer level NIC devices and not specifically designed for high performance traffic capture, however our results indicate that either NIC would be suitable for capture of traffic on a FreeBSD based system at average traffic rates of up to about 30Mb/s.

E. Linux Based Capture Devices

During experimental evaluation of running traffic capture applications (such as NetSniff and tcpdump) under Linux (Gentoo and Suse Linux Distributions, Kernel version 2.6), we have noticed a problem. While the Linux kernel appears to correctly timestamp the captured packets, they are not passed up to the capture application in chronological order. As such, the capture application will occasionally be provided with packet ***n*** after it is provided with packet (***n+1***).

This behaviour can be witnessed by simply running tcpdump and storing the resultant capture to a disk dump file. Examining the packets stored within the tcpdump file will show that they have not been stored in chronological order although the actual packet timestamps are correct.

This is not a major problem for tcpdump since it deals with packets as atomic objects, however capture applications such as NetSniff are reconstructing traffic application flows. If packets are not presented to NetSniff in chronological order, some major errors are made in determining the statistical properties of TCP streams.

The possible solutions to this problem are:

- Do not use Linux on the traffic capture and analysis workstation.
- If you must use Linux, capture to a disk file and then pre-process the data by sorting the packets into chronological order prior to further analysis with NetSniff.

IV. Summary of Results

The key points to come out of our experimental analysis of NetSniff (v050722) during processing of both stored capture files [2] and during live capture are:

1. Processing NetSniff directly on tcpdump files has no performance limitations as real-time processing is not required.
2. Processing performance is significantly affected by CPU type, less so by the system clock speed.
3. Performance is adversely affected by an increased number of concurrent flows as NetSniff must maintain an active database of each of these flows.
4. A Pentium4 based system can process captured network traffic at an average rate of about 80Mb/s
5. Anonymisation of data has minimal impact on the processing performance of NetSniff.
6. NetSniff live capture performance is limited by the small default buffer used in the PCAP (0.9.4) capture library (32kB) on FreeBSD (5.3)
7. NetSniff CPU and memory usage increases markedly when NetSniff performance is such that greater than 1% of packets are dropped during capture. Using the default configuration, NetSniff is able to capture and analyse network traffic at an average rate of about 33.4Mb/s
8. There is an error in the implementation of one or more of the TCP application parsers which can cause NetSniff to crash under heavy load.
9. Linux is not suitable as a capture platform due to the PCAP (0.9.4) library implementation not passing packets to the capture application in chronological order.

V. Possible improvements

There are two areas in which the performance of NetSniff needs to be improved, they are increasing the PCAP capture library buffer size and to fix the errors in the TCP applications parser implementations.

A. PCAP Capture Library Buffer Size

NetSniff uses the underlying PCAP library to capture packets on a network card and make them available for processing. On a default configuration (PCAP version 0.9.4 and FreeBSD 5.3), these buffers are only 32kB in size. At data rates of 100Mb/s, this buffer can potentially be filled in 0.04ms.

The major issue is that while the average packet processing rate of NetSniff may indicate that data can be processed at average rates of about 80Mb/s, it may not be that each individual packet will be processed in a fixed amount of time. Further the arrival of a burst of packets may cause the buffer overflow which we have witnessed during testing.

To this end we recommend that work be done with an eye to increasing this buffer size. The aim of doing so would be to allow NetSniff to run to its full potential when processing captured packets before resulting in a buffer overflow and the subsequent loss of data. It is expected that this will allow to capture and process traffic in real time at rate beyond our currently measured maximum of 5000pps (~33.4Mb/s) to a rate closer to 12000pps (~80Mb/s).

B. Errors in TCP Application Parsers

We found that errors exist in the implementation of one or more of the TCP application layer parsers. These parsers are passed data from the TCP stream reconstruction module to parse and extract application layer statistics. Since the TCP data stream is reconstructed as the packets are parsed, the application layer stream is passed to the application parser in small blocks – the parser is expected to be able to read and process this data in this fashion. These parsers are occasionally trying to parse data that does not yet exist (and as such has not been provided to the parser) and reading beyond the end of an allocated block of memory. The resultant access to un-owned memory is causing a core dump.

This bug has been scheduled for correction in the next release of NetSniff.

VI. Conclusion

This paper provides results characterizing NetSniff (v050722) live capture performance in high traffic scenarios. We used previously generated tcpdump traffic files played back at varying speeds – using tcpreplay – to generate data for NetSniff to capture and analyse.

Our results show that version v050722 of NetSniff using the version 0.9.4 of the PCAP packet capture library on a FreeBSD 5.3 system has a couple of implementation problems. First we note that a default system installation uses small (32kB) PCAP buffers, resulting in buffer overflow and subsequent dropped

packets occuring at a data rate that previous results have shown NetSniff to be able to manage. Further, errors in the application parser modules of NetSniff v050722 have been identified.

Our results have also indicated that the PCAP (0.9.4) implementation for Linux (kernel version 2.6) does not pass packets up to the NetSniff application in chronological order.

On a default FreeBSD implementation however, it is possible to reliably capture and process network data using NetSniff at rates of up to 5000 pps (33.4Mb/s). We would expect that increasing the PCAP buffer size should allow NetSniff to process data at the rates indicated in [2].

References

[1] J.Bussiere, J.But, "Measuring the performance of Netsniff: Testbed design", CAIA Technical Report CAIA-TR-050623A, June 2005

[2] J.Bussiere, J.But, "Measuring the processing performance of NetSniff", CAIA Technical Report CAIA-TR-050823A, August 2005

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

[4] PCAP Packet Capture Library, http://tcpdump.org, accessed June 2005

[5] NetSniff, http://caia.swin.edu.au/ice/tools/netsniff, accessed August 2005