Improving NetSniff Capture Performance on FreeBSD by Increasing the PCAP Capture Buffer Size¤

Jason But, Julie-Anne Bussiere*
Centre for Advanced Internet Architectures. Technical Report 051027A
Swinburne University of Technology
Melbourne, Australia
jbut@swin.edu.au, julie-anne.bussiere@laposte.net

***Abstract*- NetSniff is an IP traffic analysis tool that is capable of multi-protocol, live capture analysis. Previous work has revealed that under live capture situations, NetSniff performance was often limited not by available system resources, but rather by the size of the buffer employed by the PCAP library to store captured packets prior to processing. In this paper we examine how this buffer size is determined and look at what performance gains we can achieve when we increase the size of the PCAP buffer.**

***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) [1]. 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.

One problem highlighted in earlier experiments is the size of the capture buffer used by the PCAP capture library which is employed by NetSniff [2]. Using the default configuration options, this buffer is set at 32kB, the relatively small size of this buffer led to frequent overflow scenarios under test conditions which meant that NetSniff was not able to capture and parse live network traffic at the same rate that its raw processing performance figures might suggest [3].

In this paper we discuss exactly how the underlying PCAP library sizes the buffer it uses when capturing network traffic on a FreeBSD based system, and how this size might be changed to potentially improve the real-time capture performance of NetSniff. Finally we will run some tests to confirm just how much the capture performance of NetSniff has been improved.

The report is structured as follows: In the section two we discuss the implementation of the PCAP network capture library on FreeBSD and the means by which the cacpture buffer size is set. In section 3 we discuss how the capture buffer size can be changed on a FreeBSD system and in section 4 we measure the impact of increasing the buffer size in limiting buffer overflow and improving NetSniff capture performance.

II. PCAP Implementation on FreeBSD

The PCAP packet capture library is developed and maintained by the developers of tcpdump [4]. The tcpdump application also uses the PCAP capture library to perform the actual packet capture. The advantages of using PCAP are:

- Packet capture is handled for you.
- Functionality to read/write tcpdump formatted files.

On a FreeBSD based system, the PCAP capture library uses the system provided BPF (Berkeley Packet Filter) interface to actually capture the network traffic. At the time of writing, the current version of the PCAP library is 0.9.4. When looking at the source code, the file that implements the interface between the PCAP API and the underlying BPF API is called pcap-bpf.c and the relevant lines of code are listed in Figure 1.

```
if ((ioctl(fd, BIOCGBLEN, (caddr_t)&v) < 0) || v < 32768)
  v = 32768;
for ( ; v != 0; v >>= 1) {
  /* Ignore the return value - because the call fails
   * on BPF systems that don't have kernel malloc.  And if
   * the call fails, it's no big deal, we just continue to
   * use the standard buffer size.
   */
  (void) ioctl(fd, BIOCSBLEN, (caddr_t)&v);
  (void)strncpy(ifr.ifr_name, device, sizeof
(ifr.ifr_name));
  if (ioctl(fd, BIOCSETIF, (caddr_t)&ifr) >= 0)
    break;  /* that size worked; we're done */

  if (errno != ENOBUFS) {
    snprintf(ebuf, PCAP_ERRBUF_SIZE, "BIOCSETIF: %s: %s",
             device, pcap_strerror(errno));
    goto bad;
  }
}
```

Fig. 1. Buffer Size Setting Code from pcap-bpf.c

This code functions as follows:

- Attempt to read the current BPF buffer size into the variable ***v***.

¤ All experiments performed on FreeBSD v5.3 running NetSniff release v050722 and libpcap release 0.9.4

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

- If the read fails, or the current BPF buffer size is less than 32768 (32kB), set the value of ***v*** to 32768.
- In a loop attempt to set the BPF buffer size to the value of ***v***. If the attempt fails, halve the value of ***v*** and repeatedly try again.

Under the default FreeBSD configuration, the BPF buffer size is set to 4096 bytes (4kB). This means that when the BPF socket is opened by the PCAP library, it reads this value into ***v***. Since this is less than 32kB, ***v*** gets instead set to 32768. The PCAP library then attempts to set the BPF buffer size to 32kB, this call succeeds since the default system parameters allows a buffer size of up to 512kB to be set.

As such, unless certain system wide parameters are changed (the default BPF buffer size and the maximum BPF buffer size), or the PCAP library is recompiled with a different initial buffer size of 32kB, then every network capture application launched under FreeBSD will use an underlying BPF buffer of 32kB in size.

III. Increasing the PCAP Buffer Size

Recompiling the PCAP library might be an option, but then we would have to relink all the traffic capture applications to make use of the newly compiled library. More useful would be to change one – or both of – the default and maximum BPF buffer sizes as defined within the operating system.

On a FreeBSD 5.3 system, both of these values can be modified using system control variables, which can be directly modified using the `sysctl` command.

The BPF buffer size within the system is defined by the system control variable `debug.bpf_bufsize`, which has a default value of 4096. Changing this value will determine the initial value to which ***v*** is set within the PCAP library.

The maximum BPF buffer size that can be set is defined by the system control variable `debug.bpf_maxbufsize`, which has a default value of 524288 (512kB). Any attempt to set a buffer size greater than this value will fail and cause the buffer size setting loop to halve ***v*** and try again. Changing this value to 16384 for example would cause all PCAP enabled applications to run with a packet capture buffer of size 16kB as the initial (***v***=32768) gets halved when the buffer size allocation fails.

For example, to set the default buffer size to 512kB and the maximum buffer size to 1MB (with the result that NetSniff and other PCAP enabled applications would use a buffer of size 512kB) we could issue the commands [5]:

```
>sysctl debug.bpf_maxbufsize=1048576
>sysctl debug.bpf_bufsize=524288
```

Under FreeBSD, it would be prudent to add the following lines to `/etc/sysctl.conf` to ensure that these values are correctly set each time the system is rebooted[6].

```
debug.bpf_maxbufsize=1048576
debug.bpf_bufsize=524288
```

IV. Experimental Results Running NetSniff

We reset the BPF buffer size system control variable `debug.bpf_bufsize` to 512kB (524288). This larger buffer size can potentially be filled in about 40ms at an incomming rate of 100Mb/s. The expected effect is that the required traffic burst to overflow the data will be increased in size and fewer packets will be dropped as long as the average bitrate of captured traffic remains below average processing rate that NetSniff can maintain [3].

We have replayed the previously generated traffic traces consisting of traffic from 1 and 30 unique hosts [2] at a variety of different rates and consider the percentage of dropped packets as NetSniff attempts to analyse the traffic. The original results (buffer size is 32kB) are shown in Figure 1, while the new results with the larger buffer are shown in Figure 2.

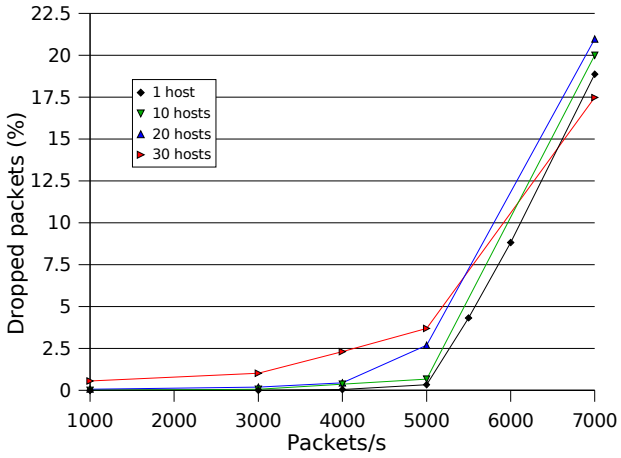

Fig. 2. **Percentage of dropped packets for different numbers of concurrent hosts**

As expected, the increased buffer size has resulted in an increase in the packet rate at which traffic can be successfully captured and analysed by NetSniff before too many packets are dropped and NetSniff cannot be used properly.

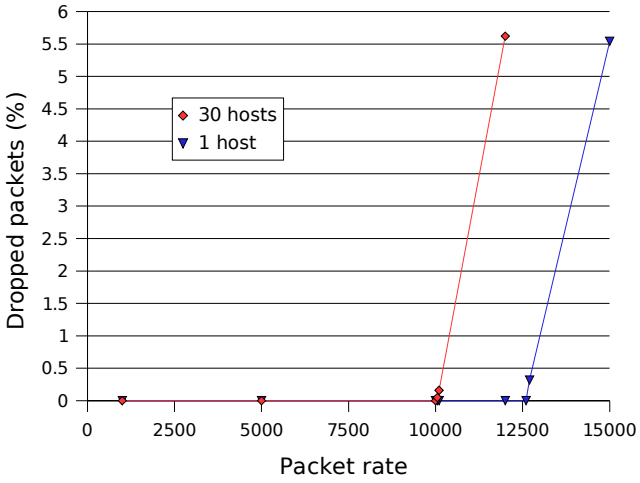

Fig. 3. **Dropped packet rate with a 512kB PCAP buffer**

For the case of traffic from one unique host, the threshold where zero packets are dropped by NetSniff has shifted from 3,000pps to 12,600pps. This

corresponds to an increase of the traffic bitrate to 85.7Mb/s. Similarly in the case of traffic from 30 unique hosts, the zero packet loss threshold has increased to 10,000pps (about 68.3Mb/s)

These increased processing rates correlate well with the expected maximum processing rate of NetSniff on a Pentium4 equiped platform of about 80Mb/s [2]. We would expect that on future machines with increased processing power, we would again be able to match the raw NetSniff processing rate with the live capture processing rate through a judicious increase in the PCAP buffer size.

V. Conclusion

Previous experimentation revealed that NetSniff performance in a live packet capture scenario was not limited by available system resources (processor type, clock speed, system memory), but instead by the limited size of the fixed buffer employed by the PCAP capture library. In this paper we examine how this buffer size is determined within the PCAP library and measure the performance of NetSniff with a larger configured buffer.

The default configuration of a FreeBSD based system means that any traffic capture and analysis software that uses the PCAP capture library will run with a fixed buffer size of 32kB. It is possible to modify this behavious through changing the values of two system control variables `debug.bpf_bufsize` and `debug.bpf_maxbufsize`. The values of these variables can be changed permanently by setting their values in the file `/etc/sysctl.conf`

We increased the system buffer size from the default value of 4kB (which results in PCAP setting a buffer size of 32kB) to 512kB and re-ran the live capture experiments from [3]. Under the new system conditions, we discovered that NetSniffs live capture performance now correlated with its raw packet processing performance, an indication that NetSniff was now being limited by system resources rather than the PCAP buffer size.

Our results pertain specifically to a FreeBSD 5.3 based system running version 0.9.4 of the PCAP library and v050722 of NetSniff. While some of these problems may be addressed in future implementations of any of these three software products, we expect that our solution can continue to be applied while the buffer overflow problem remains.

References

[1] Inverted Capacity Extended Engineering Experiment (ICE[3]), http://caia.swin.edu.au/ice, accessed August 2005

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

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

[4] TCPDump, http://tcpdump.org, accessed June 2005

[5] FreeBSD System Managers Manual, **sysctl (8)**, System Man Pages

[6] FreeBSD File Formats Manual, **sysctl.conf(5)**, System Man Pages