

TCPDUMP SUBSAMPLING PROBLEM ON THE DETERLAB TESTBED

Soranun Jivasurat

Department of Computer Science and Engineering

The Pennsylvania State University

jiwasura@cse.psu.edu

1. Introduction

In this report, we discuss the packet subsampling problem when using TCPDUMP program to capture packets on the DETERLAB emulated router node running Redhat operating system¹. First, we describe the setup of our experiment for measuring the performance of TCPDUMP. We then explain about several of TCPDUMP that were used in the experiment. Finally, the results from the experiment are shown and discussed.

2. Setup the Experiment

In our experiment, we measured the performance of TCPDUMP programs on a PC 733MHz (Pentium III) running **Redhat 7.3** operating system. Figure 1 shows the topology used in the experiment. *NodeA* is a PC running an UDP traffic generator program. *NodeB* is a receiver node which acts as a traffic sink. *NodeC* is configured as an emulated router which routes traffic between NodeA and NodeB over 100Mbps Ethernet links.

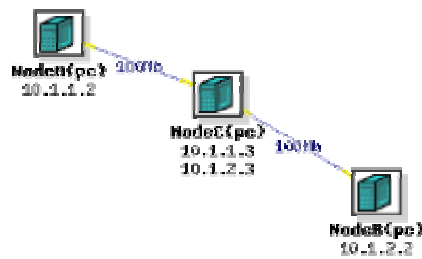


Figure 2 An experiment topology

The UDP traffic generator program, run on NodeA, is a C program that continuously sends **404-byte UDP packets** on a designated port to NodeB. In fact, we developed the UDP traffic generator program as a program that generates simulated SQL-slammer scanning traffic. The scanning rate of the UDP traffic generator program is adjustable and can be modified by users. The traffic sink application, run on NodeB, is also a C program that listens on the UDP designated port and silently drops all received packets. The TCPDUMP program was run on NodeC to capture scanning traffic from NodeA passing through its network interfaces. For the topology in Figure 1, two TCPDUMP programs were run to capture traffic on both interfaces of NodeC.

¹ Vern Paxson suggested that by enlarging the BPF buffer size under FreeBSD OS, the TCPDUMP subsampling problem will rarely exist.

3. Alternate versions of TCPDUMP

Three versions of the TCPDUMP program were tested in this experiment. The first one was TCPDUMP program version 3.6 with libpcap version 0.6, which is a standard version provided on the DETERLAB testbed. The other two versions were our compiled TCPDUMP programs, namely the TCPDUMP program (version 3.8.3 with libpcap version 1.0) with MMAP libpcap [1] and the TCPDUMP program (version 3.8.3 with libpcap version 0.8 on PF_RING kernel patch) with device polling libpcap [2]. The performance of these three versions will be discussed in next section. Note that *libpcap* is a programming library that provides a packet capture interface for user-space programs like TCPDUMP and SNORT.

4. Experiment Result and Discussion

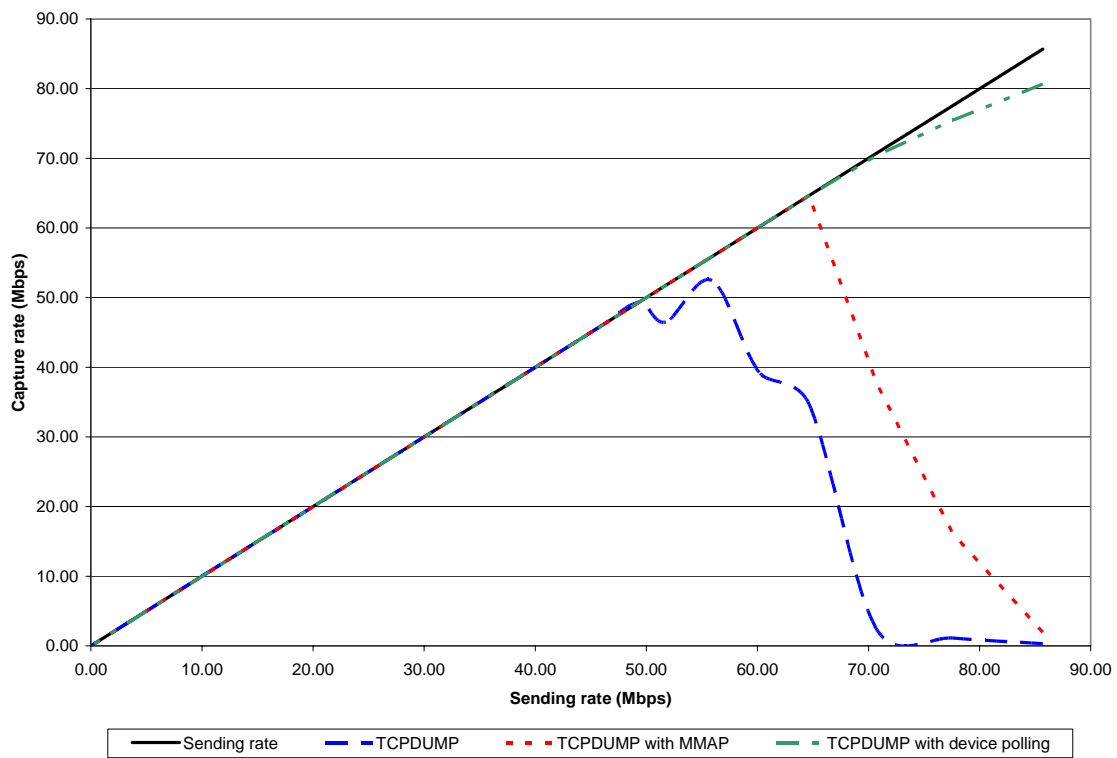


Figure 2 Capture rate of TCPDUMP programs on 100 Mbps link of NodeC

We now discuss about the experimental results. Figure 2 shows the packet capturing performance of TCPDUMP programs on 100Mbps link. Recall from Section 2 that the traffic in the experiment was generated by a constant-rate UDP traffic generator program. In the figure, the solid line represents the sending rate of the generated UDP traffic from NodeA. The dash lines illustrate the packet capturing performance of TCPDUMP programs.

As can be seen, the TCPDUMP (the version provided by the DETERLAB testbed) starts subsampling packets when the sending rate is 50Mbps and it misses almost all packets when the sending rate is greater than 70Mbps. The performance of the TCPDUMP with MMAP libpcap is not significantly

better since it starts dropping packets at the sending rate of 64Mbps and the capture rate declines dramatically after this point. The performance of the TCPDUMP with device polling libpcap has a superior performance over the other two TCPDUMP programs. It captured almost all packets from the traffic generator program. However, the TCPDUMP with device polling libpcap did have problems on an emulated router routine: in the experiment, we observed that there was no packet received at NodeB when the TCPDUMP with device polling libpcap was running.

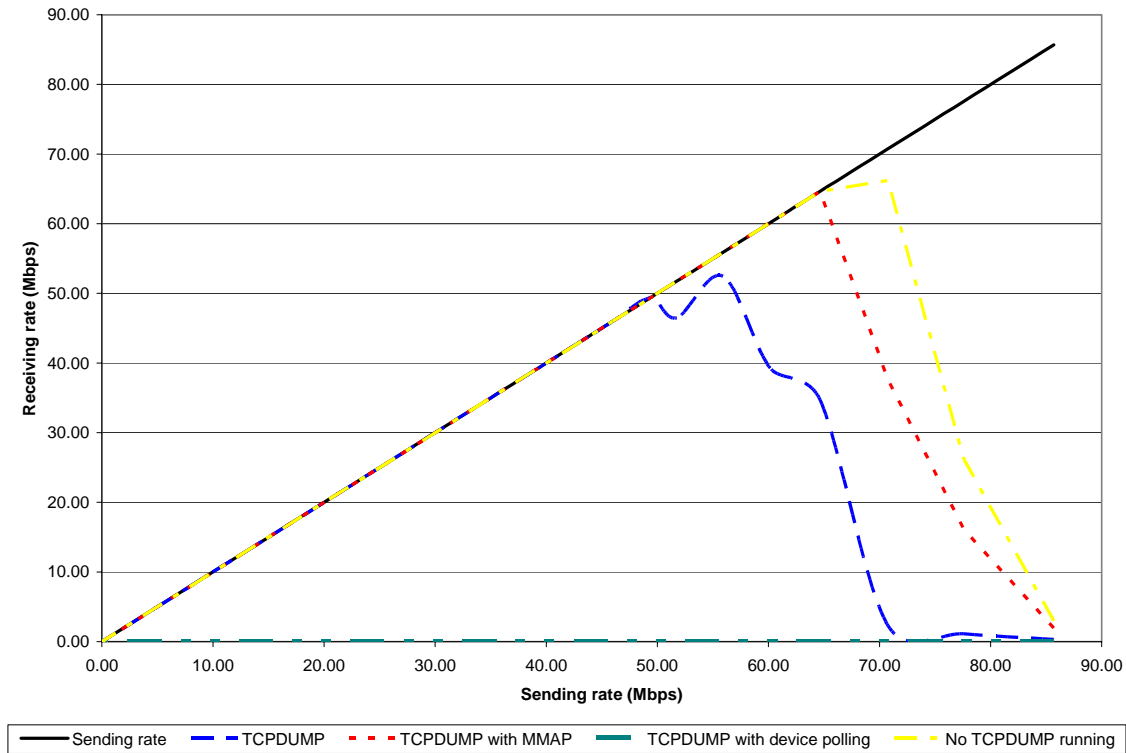


Figure 3 The receiving rate at NodeB under different versions of TCPDUMP programs

Figure 3 shows the receiving rate at NodeB when there were different versions of TCPDUMP programs running at NodeC. As can be seen, the receiving rate under TCPDUMP with device polling libpcap is always zero. In contrast, we observed that the receiving rates at NodeB, when TCPDUMP or TCPDUMP with MMAP libpcap running, are likely the same as the capture rates as shown in Figure 2. Interestingly, when there was no TCPDUMP program running at NodeC, we observed a better receiving rate at the sending rate higher than 64Mbps compared to the receiving rates when TCPDUMP programs running. We believe that the TCPDUMP program could impose some burden on the CPU when capturing traffic at speeds higher than 64Mbps, resulting in some dropped packets at the emulated router.

To work around this problem, Nick Weaver² and Roman Chertov³ suggested running TCPDUMP program on a sink/terminating end-system node instead of on an emulated router node (NodeC). This sink

² Nick Weaver is with the ICSI, Berkeley CA.

node is directly attached to a link between a sender/receiver node and an emulated router node as show in Figure 4. By this way, we can avoid the burden from TCPDUMP program on an emulated router node and expect to see a better capturing performance of TCPDUMP program.

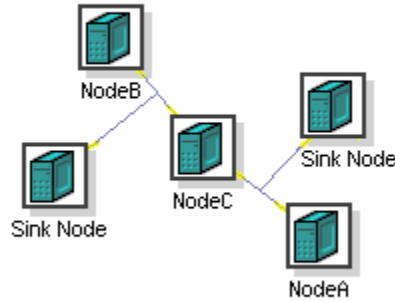


Figure 4 A sink node is attached to a link between a sender (NodeA)/ receiver (NodeB) node and an emulated router (NodeC) for running TCPDUMP program

Another feasible solution is to run TCPDUMP program with **FreeBSD** operating system on a sink node. As suggested by Vern Paxson and Nick Weaver, the BPF buffer size (`debug.bpf_bufsize`) needs to be increased to the size much larger than the default value (4096 bytes) in order to capture high traffic volume (100+Mbps) by TCPDUMP program. To set the BPF buffer size, one can use `sysctl` command on FreeBSD. We expect that the capture rate of TCPDUMP program is much higher under FreeBSD than under Redhat.

Lastly, we conducted the same set of experiment using a PC2800 (Pentium 4 Xeon) as an emulated router. We found that the performances of all TCPDUMP programs running on a PC2800 are about 10-15% better than the performances on a PC733. Moreover, we repeated the above experiment but replacing 100Mbps Ethernet links with 1Gbps Ethernet links, and PC733 with PC2800 for all nodes. We observed that all TCPDUMP programs captured higher traffic rates and the capture rate is maximum at about 231Mbps for TCPDUMP with MMAP libpcap. In a pattern similar to the above results, all TCPDUMP programs start subsampling packets after their maximum capture rate.

5. Conclusions

We observed the packet loss problem when using TCPDUMP programs to capture packets on an emulated router. We first compared our compiled TCPDUMP program with MMAP libpcap with the TCPDUMP program provided by the DETERLAB testbed and found that the performance of TCPDUMP with MMAP libpcap is about 25% better than the performance of the DETERLAB TCPDUMP. Moreover, we found that the TCPDUMP with device polling libpcap has a superior performance in capturing packets. However,

³ Roman Chertov is with the CSE Dept of Purdue.

running the TCPDUMP with device polling libpcap has an effect on the emulated router routine, i.e., the emulated router routine stops routing traffic. Finally, we found that running TCPDUMP on an emulated router could cause some dropped packets when it captures the traffic at speeds higher than 64Mbps. We believe that the TCPDUMP program could overload the emulated router node when capturing traffic at high as a 64% of link speed. Finally, we discussed several work-around solutions including running TCPDUMP program on a terminating end-system node instead of on an emulated router node and running TCPDUMP program with FreeBSD operating system.

References

- [1] Phil Wood, “*Libpcap with MMAP support*”, available at <http://public.lanl.gov/cpw/>
- [2] Luca Deri, “*Improving Passive Packet Capture: Beyond Device Polling*”, available at <http://www.ntop.org>, January 2004