

# Performance Evaluation of Packet Capturing Systems for High-Speed Networks

Fabian Schneider  
Technische Universität München  
fabian@net.in.tum.de

Jörg Wallerich  
Technische Universität München  
jw@net.in.tum.de

## ABSTRACT

Using commodity systems for capturing packets in a Gigabit environments is a challenging task. This applies especially to a full capture of packet headers along with their data. Today's commodity PC systems come in different flavors in terms of processor hardware as well as in terms of operating systems. In this paper we describe a methodology for evaluating different systems with respect to their maximum capture rate and present our preliminary results of comparing Intel Xeon against AMD Opteron based systems running either Linux or FreeBSD.

**Categories and Subject Descriptors:** C.2.3 [Computer Communication Networks]: Network Operations – *Network monitoring*

**General Terms:** Measurement, Performance

**Keywords:** FreeBSD, Linux, Packet Capturing

## 1. INTRODUCTION

Packet capture is an essential component of most network measurement systems and network security systems including intrusion detection and prevention systems. Contemporary Gigabit environments present a challenge for today's commodity systems as their throughput is close to the maximum capacity of such systems. One way to ensure that all data in Gigabit environments is captured, is to use specialized hardware such as network monitoring cards by Endace [2]. Such specialized cards are in general not part of a commodity system. Therefore it is crucial to periodically reevaluate the performance limitations of various combinations of hardware and operating systems in order to allow an appropriate platform choice.

Among the challenges that have to be addressed to answer the above question are: how to ensure that all systems are subjected to the same input, how to make sure that all systems are running at an equal level of OS optimization, and finally most important how to identify the reasons for the performance differences. For example in the past it turned out to be important to minimize the number of system interrupts [4] and to use smarter buffers either within the kernel [1] or as an enhancement to the most commonly used capture library libpcap [3, 7].

This paper focuses on the question of packet capture only. It does not study the additional system requirements needed for further processing of the captured data, e. g., an intrusion detection system. The goal of this work is to describe a methodology to figure out which combination of hardware architecture and operating system performs best. Accordingly, Section 2 describes our measurement setup while Section 3 presents our results. We finish with an outlook in Section 4.

## 2. SETUP

For our comparison we used systems that were purchased at the same time with comparable components. Indeed, the systems have the same optical Intel Gigabit Ethernet card, the same kind and amount of memory and the same type of hard disk and RAID controller. We chose to use dual processor machines to be able to evaluate the benefit/penalty of multi-processor boards. We selected four systems, two AMD Opteron based systems and two Intel Xeon based systems, to be able to evaluate two different operating systems on both architectures: Linux 2.6.11 and FreeBSD 5.4. The kernels are basically vanilla kernels with only a few performance optimizations. For both operating systems we increased the amount of memory available for the kernel capture engine by setting either the `debug.bpf_bufsize sysctl` parameter (FreeBSD) or the `/proc/sys/net/core/rmem*` parameter (Linux) to 10 MBytes. For Linux we enabled device polling (NAPI) in the device driver.

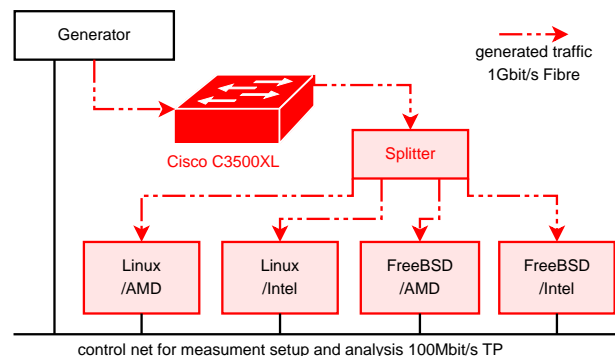
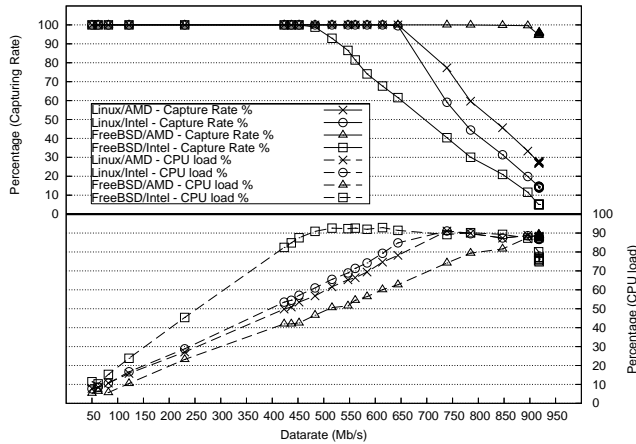


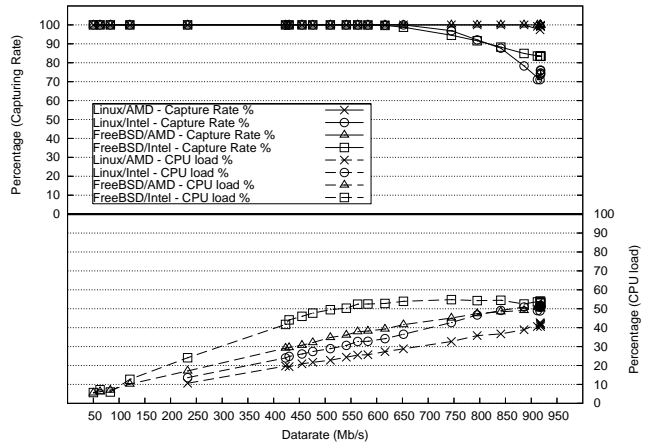
Figure 1: Measurement setup.

To take full advantage of the four systems under test we used the topology shown in Fig. 1. The optical splitter ensures that all systems see the same input data generated by a traffic source. In principle the traffic source can either be a live traffic stream or synthetically produced by a traffic generator. The Cisco switch is used to count the number of generated packets and to assure that the configured input rate at the generator is indeed achieved.

Our measurements rely on reproducible traffic at scalable rates. Therefore live traffic is not our first choice. In general packet capture is harder if the packet rate is high, which can be achieved by using small packets, and easier if the packet rate is low, obtained with large packet sizes. Instead of exploring all different packet sizes we decided to use input traffic with about the same packet size distribution as actual traffic. For this purpose we enhanced the Linux Kernel Packet Generator [6] which by default can only generate packets at a given packet size. Using a packet size distribution as input, the generator generates a packet stream with packet sizes



(a) Single processor mode



(b) Multiple processor mode (with SMP)

**Figure 2: Performance: Capture rate (upper part)/CPU Usage (lower part) versus data-rate.**

that are consistent with the input distribution. We decided to derive the packet size distribution from a 24 h trace obtained at the uplink of the Münchner Wissenschaftsnetz [5]. It shows the typical peaks at 40–64, 576 and 1420–1500 bytes. Using this trace, the modified packet generator is able to achieve data rates of up to 920 Mbit/s which corresponds to packet rates of about 180,000 pps. Thus the input traffic is able to load the Gigabit link to its limits. The data rates are scaled by using different inter-packet gaps.

For the purpose of comparing the different systems, we are interested in how the capture rate and the CPU usage changes as the data rate increases. The capture rate tells us if the system is able to receive all packets or if it is losing packets. The CPU usage indicates how much of the CPU is available to other applications, e. g., those that do the data aggregation or/and inspection. The capture rate is determined with a program that, using libpcap [3], counts the number of received packets of each size. The CPU usage is measured applying a program that uses the same mechanism as top and records the results in a file.

### 3. RESULTS

Using the above setup we now evaluate the performance of the four systems as the data rate is scaled using 26 different inter-packet gaps between 0.1 ms and 0 ms resulting in rates from 50 Mbit/s to 920 Mbit/s. Each experiment consisted of 1,000,000 packets and was repeated 7 times. Fig. 2 shows the average capture rate and the average CPU usage. To keep the plot simple we chose to not include the standard deviation. Note that all measurements have a standard deviation of less than 2 percent.

Overall the results, plotted in Fig. 2, show that packet capture consumes significant CPU resources. As the data rate increases the capture rate decreases and the CPU usage rises. Enabling the second CPU improves the performances of all systems. Note that the CPU usage is for the overall system. This implies that the single-threaded packet capture application can impose a maximum CPU utilization of 50 % on multiprocessor systems.

In single processor mode (Fig. 2(a)) all machines begin losing packets once the processor is fully utilized<sup>1</sup> Without increasing the memory available to the “packet-capturing stacks” all systems lose major fractions of the packets without fully utilizing their

CPU. The FreeBSD/AMD combination on the one hand loses almost nothing, on the other hand the FreeBSD/Intel combination already starts dropping packets at about 500 Mbit/s. Neither of the Linux systems loses packets until roughly 650 Mbit/s. From that point onward their performance deteriorates dramatically with increasing data rates. The efficiency of the FreeBSD/AMD machine is especially surprising as FreeBSD performs an additional (kernel) packet copy operation and does not use device polling which proved beneficial on Linux systems.

While in our experience FreeBSD outperforms Linux, the hardware architecture also has a significant impact. The disadvantage of Intel systems is especially visible in the multiprocessor mode, Fig. 2(b). At about 650 Mbit/s the Intel based machines both begin dropping a significant fraction of packets. This may be due to a problem in the overall SMP design of Intel’s Xeon CPU’s, most likely within the memory management.

### 4. SUMMARY

Overall our methodology allows us to evaluate the packet capture rates of various commodity systems. Our preliminary results indicate that the multiprocessor Opteron system with FreeBSD 5.4 outperforms all other systems. Its maximum loss rate is less than 0.5% while utilizing one of its CPUs.

We are still investigating the differences in the performance results. Furthermore we are planning to explore other parameters that might impact the performance such as hyper-threading, complex BPF filters and newer OS versions. In addition we plan to investigate the performance impact of background load, multiple packet capturing applications, as well as basic data analysis applications. Being able to identify the appropriate architecture/operating system pair for the particular task is the ultimate goal of this work.

### 5. REFERENCES

- [1] L. Deri. Improving passive packet capture: Beyond device polling. <http://luca.ntop.org/Ring.pdf>, Nov. 2003.
- [2] Endace Measurement systems. <http://www.endace.com>.
- [3] V. Jacobson, C. Leres, and S. McCanne. libpcap. <http://www.tcpdump.org>.
- [4] J. C. Mogul and K. K. Ramakrishnan. Eliminating receive livelock in an interrupt-driven kernel. *ACM Transactions on Computer Systems*, 1997.
- [5] Leibniz Rechenzentrum, The Munich Scientific Network. <http://www.lrz-muenchen.de/wir/intro/en/#science>.
- [6] R. Olsson. Linux kernel packet generator.
- [7] P. Wood. Burnt offerings. <http://public.lanl.gov/cpw/>.

<sup>1</sup>Full CPU utilization is plotted at about 90 % because of summary impacts.