CASPER Memo 21:
10GbE Network card benchmarking

J. Manley, T. Filiba, D. Werthimer

February 2008

Contents

1 Introduction

CASPER's FPGA-based hardware platforms (iBOB, BEE2 and ROACH) are able to sustain ethernet transmission rates of 10 gigabits per second (Gbps), however, up until recently a single computer was unable to maintain this data rate. Factors such as the computer's CPU, bus speed, memory allocation, the choice of network interface card (NIC), the packet size, and how much processing takes place on the received data (for example, data manipulation or disk storage) all affect the received data rate.

In our applications, we have either transmitted data from FPGA-to-FPGA or FPGA-to-switch at 10 Gbps, but typically avoided transmitting the full 10 Gbps data rate into a single computer due to computer performance restrictions. We were thus limited to either transmitting at lower rates or had to resort to a breakout switch to split the 10 Gbps rate from the FPGA into several computers at 1 Gbps. Modern computers with 10 Gbps PCI Express NICs are able to maintain the full 10 Gbps line rate.

This paper serves as a reference for the expected 10 gigabit ethernet (10GbE) performance of different computer systems receiving UDP packets sourced from the CASPER hardware. It will be a running record of 10GbE performance metrics. Additional entries will be made to this document as new hardware is tested.

This document is not intended to benchmark the ability of a given system to process the received data or even write it to disk, but simply to receive packets without loss.

2 Test procedure

The hardware platform under test is configured as follows:

2.1 FPGA

The FPGA is loaded with a 10GbE test-suite bitstream which has runtime configurable packet lengths (up to 12 kilobytes + UDP header) and packet rates (0.05 packets/second to 2 million packets/second). The packets' payloads contain a continuous 64 bit counter. The datacable is connected through the network under test - either directly or using a network switch.

2.2 Computer

Software was written to receive the above packets and check for correct payload length and continuous counter sequence. Should there be a break in the counter values, the difference is used to calculate the number of packets dropped. The

software by default reports the number of dropped packets per million received. The packet payload length and number of packets to receive are pre-configured for each test case.

Recording packet losses This setup ignores packet loss reported by the kernel (which we have found to be an unreliable metric), and calculates losses explicitly. The software reports system-wide losses (for example, packets dropped in an upstream network switch will also be reported, of which the kernel would be unaware). We believe this testbench is thus also useful for switch comparisons.

Packet processing and data storage Minimal packet data processing takes place and this document is not intended to benchmark the ability of a given platform to manipulate the received data or write it to disk. It is simply a reference against which other systems can be compared for the purposes of absorbing UDP packets at high data rates.

Process priority If the network interface card (NIC) cannot DMA packet contents directly to memory, its buffer overflows quickly and results in significant packet losses. Further, the user code needs to empty the receive buffer regularly to prevent overflows. Unless otherwise stated, the code is executed with a priority of -20 (*nice -n -20*).

3 Myricom Myri-10G PCIe

This test was executed between a BEE2 user FPGA and CASPER's "Bumblebee" development computer.

3.1 Computer Platform

Manufacturer: Custom whitebox with ASUS P5N32-E SLI mainboard

CPU: Intel Core2 Duo CPU E6550 @ 2.33GHz, 4MB cache

RAM: 2GB DDR2

Chipset: NVIDIA nForce 680i SLI

OS: Linux Ubuntu 7.10, with 2.6.20-16-generic kernel

NIC driver: myri10ge 1.3.1

3.2 Specific Configuration Changes

This was a standard Ubuntu install with all network settings bar one left at their defaults. The only change being the increased receive buffer size of 16MB:

```
sysctl -w net.core.rmem_max = 16777216
```

Increasing this buffer above 16MB made very little difference.

Further, the Myricom driver was left to its default settings. Myricom recommend multiple changes to improve performance:

```
http://www.myri.com/serve/cache/511.html#linux
```

For our purposes (UDP dumps from hardware, with variable packet sizes), we find that most of these are not necessary.

3.3 Results

Figures 1 and 2 illustrate how much of the sent data was received. Clearly, larger, less frequent packets result in higher throughput. Further, it appears that packet losses begin to occur above 570 thousand packets per second, irrespective of the payload size.

Process priority Running the receive code with a priority of -20 vs 20 resulted in approximately a factor of 2 improvement in throughput. This was especially evident at higher packet rates.

CPU utilisation Although not strictly measured, *top* indicated usage of less than 10% throughout the test.

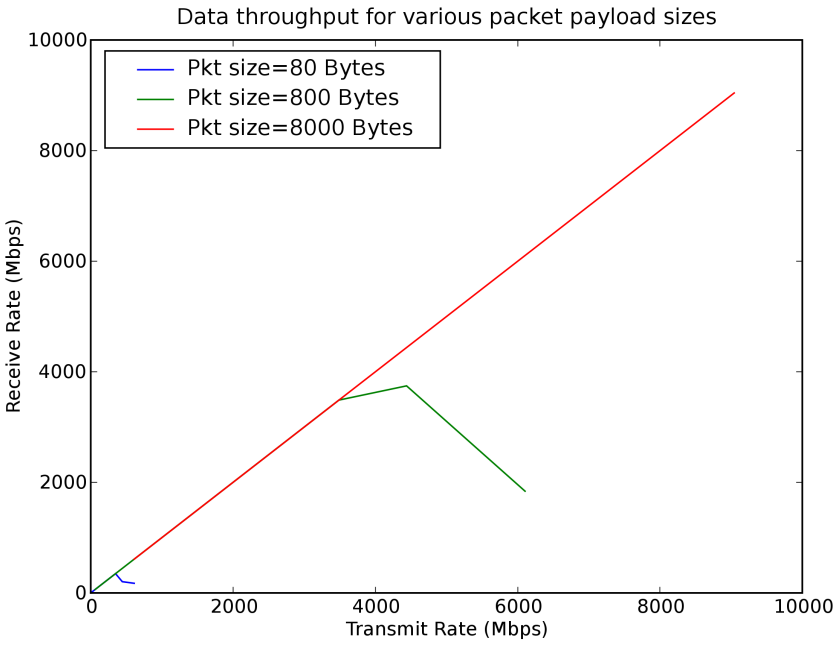

Figure 1: This plot shows the received data rate vs the transmitted data rate for three different packet sizes on Bumblebee's Myricom 10GbE card. Ideally, all sent data would be received, however, the results clearly show that larger packets result in better throughput.

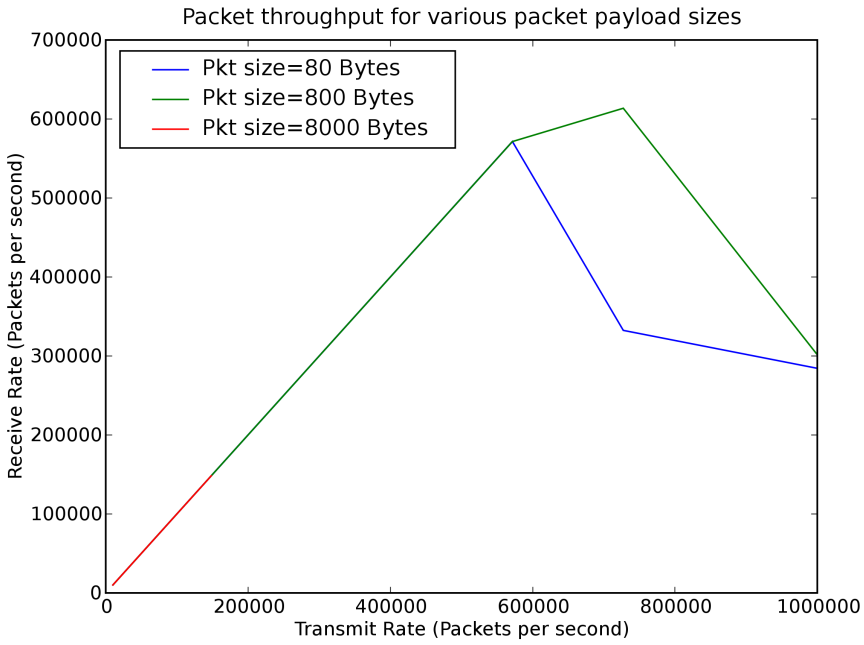

Figure 2: This plot shows the received packet rate vs the transmitted packet rate for three different packet sizes on Bumblebee's Myricom 10GbE card. Ideally, the received rate would be the same as the transmitted rate, however, the results clearly show that larger packets result in better throughput.

4 Recommendations

Use Jumbo Packets It is possible to achieve full 10Gbps throughput with careful selection of UDP packet size on a reasonably modern computer. Larger packets result in higher throughput. Use as large a size as your application allows. Most switches and network cards/drivers limit you to $\sim$ 9kB. Fujitsu switches such as the XG2000c allow up to 15kB frames. Full line-rate should be achievable with anything over 2200 byte frames.

Limit the packet rate The data throughput limitation for consumer PCs appears to be based on the packet rate, rather than the data rate. Packet rates above 570 000 packets per second results in significant losses. Lower packet rates with larger payloads results in lower datalosses and improved throughput. We thus recommend keeping the packet rate below 500 000 packets/second.

Increase the receive buffer size Most Linux installs default to a receive buffer of only 128kB. Increasing this to a few MB reduces packet loss figures significantly.

Run receive code with high priority To prevent buffer overflows, the receive code should run with a higher priority than other sporadic processes which may access the PCIe bus or memory.