

CESNET technical report number 10/2003
Field trial with Intel 10 Gigabit Ethernet adapters for
PC

Sven Ubik, <ubik@cesnet.cz>
CESNET, Prague, Czech Republic

October 21, 2003

Abstract

Intel PRO/10GbE LR adapter is the first commercially available 10 Gigabit Ethernet adapter for PC. In this report we describe our observations about the adapter functionality and performance, with respect to the possibility of the use for 10 Gigabit end-to-end communication over a wide-area network.

1 Obtaining adapters

Intel PRO/10GbE LR is the first commercially available 10 Gigabit Ethernet adapter for PC. It was announced by Intel in 2002, but it took about a year until the adapter moved from strictly limited sampling for carefully selected customers to almost normally available commercial product. Still, we had to sign NDA with Intel not to disclose any confidential information to be allowed to buy two of these adapters. Everything in this report is based on our own independent observations and publicly available information and is in compliance with NDA.

2 Installation and test setup

The test setup is shown in Fig. 1. Each adapter was installed in a PCI-X 64-bit 133 MHz slot of a Dell 2650 server. This slot has its own PCI bus, that is no other device shared the same PCI bus. The server was equipped with one Intel Xeon 2.4 GHz processor and 1 GB RAM. The two adapters were connected back-to-back with a short optical patch cable. Both machines ran Debian Linux with 2.4.22 kernel.

Intel provides driver for Linux including source code on its web server [1]. Binary drivers for Windows Server 2003 and Windows 2000 are also available. Driver installation was smooth and without any problems. Intel instructs that if there is a PRO/1000 family adapter installed in the same machine with a PRO/10GbE LR adapter, both drivers must be from the same “set”. We had three PRO/1000 family adapters installed in the same machine. We used the latest drivers for PRO/1000 family adapters also available from the Intel web server and all adapters worked without problems.

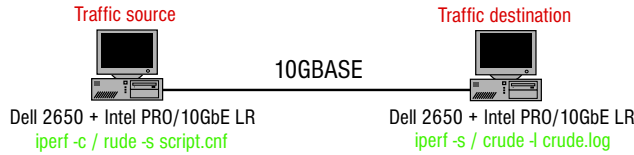


Figure 1: Test setup

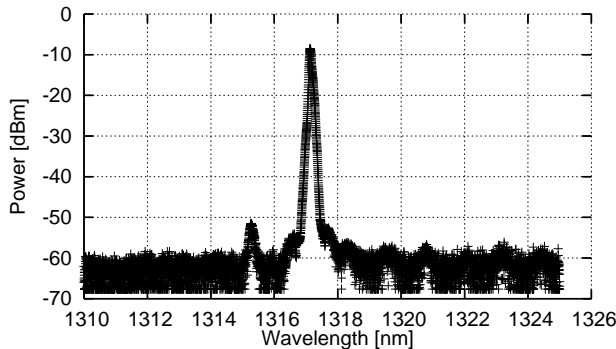


Figure 2: Spectral distribution of Intel PRO/10GbE LR laser

3 Optical power budget

The Intel PRO/10GbE LR adapter uses a 1310 nm laser with the specified reach of 10 km. Unfortunately, the transceiver is of the “300-pin” type and cannot be replaced with another transceiver (such as one using a 1550 nm laser, which would be an attractive option), which is possible with XFP or XPAK type transceivers.

The output power measured by the Expo FOT-90A fibreoptic power meter was -3.2 dBm ($482 \mu\text{W}$) on one adapter and -5.0 dBm ($313 \mu\text{W}$) on the other adapter. We inserted the Expo FVA-60B variable attenuator between the two adapters to find the maximum acceptable power budget between the sender and receiver. We found that the maximum acceptable attenuation with no packet losses was 7.85 dB in one direction and 8.30 dB in the other direction. Taking the lower value and considering attenuation of 0.35 dB/km at 1310 nm on standard optical fibers, we can estimate the maximum possible distance between the sender and receiver to approximately 22 km.

We also measured spectral distribution of the adapter laser with the Expo FTB-400 Optical Spectrum Analyser, see Fig. 2. We can see that the laser produces clear one-peak spectrum.

4 Interconnection with 1550 nm devices

High-speed optical ports on routers and switches use 1550 nm lasers more frequently than 1310 nm lasers. The reason is most optical fibers have lower attenuation at 1550 nm and the signal at this wavelength is also easier to amplify, which allows to span longer distances. It is important for high-speed ports, which are usually used for long-distance backbone circuits. On the other hand, PC adapters are usually designed for use in local

MTU	Throughput	CPU load on sender / receiver	Interrupts on server / receiver
1500 B	1.3 Gb/s	70 % / 63 %	8957 / 9040
16114 B	2.6 Gb/s	99.9 % / 31 %	179823 / 13222

Table 1: TCP throughput, CPU load and number of interrupts

networks and are therefore often equipped with cheaper 1310 nm lasers. In this case, for end-to-end communication over a wide-area network we need to resolve how to connect 1550 nm and 1310 nm devices together.

Optical wavelength convertors are not yet commercially available. We can expect that if such a device becomes available, it will be expensive. We can however take advantage of broadband sensitivity of most optical receivers. We tried to connect the Intel PRO 10GbE LR adapter directly to the Cisco Catalyst 6500 switch with the WS-X6502-10GE 1-port 10 Gigabit Ethernet adapter, which uses an extended reach 1550 nm laser. The 1-port adapter has a fixed non-interchangeable transceiver, unlike the more expensive 2-port and 4-port adapters, which use XENPAK type interchangeable transceivers. A 5 dB attenuator was inserted in each direction to protect the receiver from a possible damage by a high power level from the sender (it was actually needed only in the direction from the extended reach laser). Communication worked without any problem and we successfully sent and received 6 millions of 1500-byte packets, that is $3.6 * 10^{11}$ bits without any packet loss or damage. This shows that the Intel PRO 10GbE LR adapter can be reliably used to connect to the Catalyst’s 1550 nm transceiver, thus enabling an end-to-end 10 Gigabit Ethernet communication over a wide-area network.

5 TCP throughput

We used iperf to measure TCP throughput. Both sender and receiver socket buffers were set to 1 MB. Transmission interface queue (txqueue) was set to 10000 packets. This value was computed using a rule of thumb [2] that it should cover transmission at the physical interface rate (10 Gb/s) for the duration of the operating system scheduling timer (10 ms). PCI-X burst transfer size was increased from the default value of 512 bytes to 4096 bytes by writing to the adapter MMRBC register with the command `setpci -d 8086:1048 e6.b=2e` (see Appendix A for more information). We used a standard MTU of 1500 bytes and the maximum MTU of 16114 bytes supported by the adapters. We also monitored CPU load with the `top d 1` command and the number of generated interrupts by reading the `/proc/interrupts` file before and after the test. Achieved throughput, observed CPU load and the number of interrupts are shown in Table 1.

Observations:

- Achieved throughput was much lower than a known limitation by PCI-X 64-bit 133 MHz bus, which should have a theoretical throughput of 8 Gb/s.
- CPU was saturated on the sender with MTU=16114 bytes, but not on the receiver and not with MTU=1500 bytes, so the speed of CPU probably limited throughput with MTU=16114 bytes, but there was another limitation of throughput with MTU=1500 bytes.
- With the exception of the sender with MTU=16114 bytes, the number of interrupts was much lower than the number of packets sent or received. We did not explicitly configured delaying of interrupts when loading the adapter driver. Despite of that, interrupt coalescing was performed.
- With MTU increased to 16114 bytes, the number of interrupts on the sender increased significantly to approximately one interrupt per packet (but still slightly less). It appears that interrupt coalescing stopped working for some reason. We previously experienced the same effect with Intel PRO/1000 family adapters. The increased number of interrupts was probably a cause of CPU load increased to 100 %.

We tried setting of interrupt delaying when loading the adapter driver by `RxIntDelay` and `TxIntDelay` command line arguments with various values. We also increased the number of packet descriptor buffers by `RxDescriptors=4096` and `TxDescriptors=4096` arguments. However, this change has not resulted in apparent change in throughput.

6 UDP throughput

In order to eliminate overhead of TCP caused by its protocol processing tasks and possibly by congestion control we also measured UDP throughput (it was difficult to monitor TCP connection because we did not want to run `tcpdump` or `web100` on the same machine, which would most likely increase overhead and we did not have a third machine with 10 Gigabit Ethernet adapter to capture mirrored traffic). We used `RUDE/CRUDE` utilities [3] to send UDP streams of 1500-byte and 16114-byte packets at the rate increasing from 300 Mb/s to 5 Gb/s in 100 Mb/s steps. The measured throughput and packet loss are shown in Fig. 3 for 1500-byte packets and in Fig. 4 for 16114-byte packets. We can see that the achieved throughput was about 1380 Mb/s for 1500-byte packets and 2110 Mb/s for 16114-byte packets. That is UDP throughput was approximately the same as TCP throughput for 1500-byte packets, but lower for 16114-byte packets.

We then sent a constant-rate UDP stream of 2 Gb/s to check CPU load. On the receiver side, the CPU load was 95 %. On the sender side, `RUDE` uses active waiting by repeatedly calling `gettimeofday()` until the next packet should be send. Therefore, it always consumes all CPU it is allowed by process scheduling. We used the `strace` command to check how much time was spent in individual system calls. We found that `sendto()` was called 265926 times, which exactly corresponds to the number of packets sent and it consumed 86.1 % of CPU time. `gettimeofday()` was called 531856 times, which is two calls per packet plus 4 extra calls and it consumed 13.8 % of CPU time.

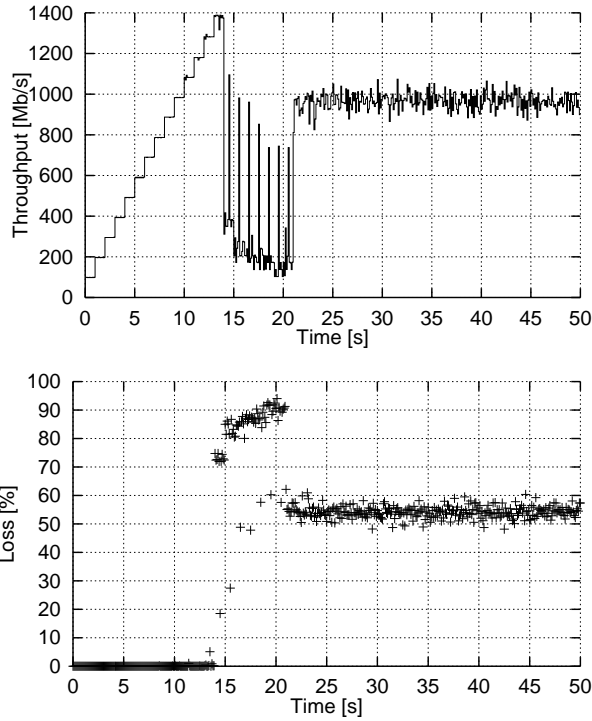


Figure 3: UDP throughput and loss for 1500-byte packets

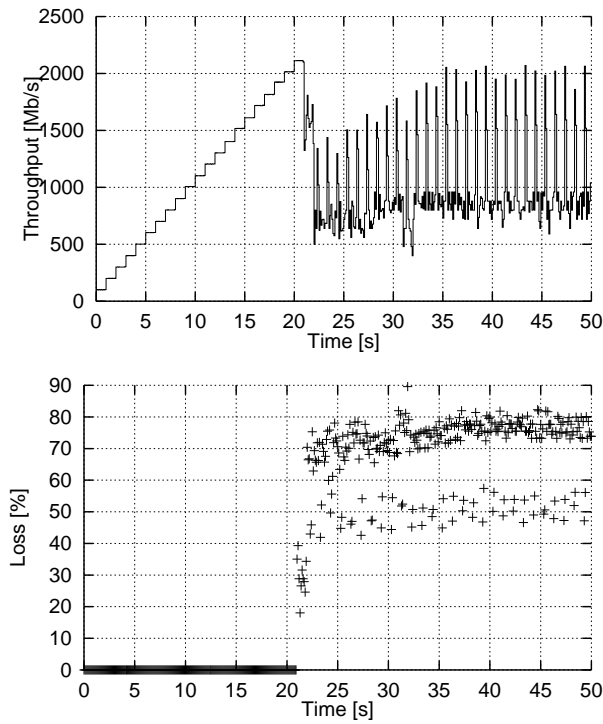


Figure 4: UDP throughput and loss for 16114-byte packets

Almost all CPU time was spent in one of these system calls. As `gettimeofday()` was called only twice before scheduling time of each packet, there was not much reserve in active waiting in RUDE. This implies that almost all CPU time of the sending machine was spent by processing packets in kernel. RUDE did not report any unsuccessful attempts to send data, which could happen when `sendto()` system call returns the number of sent bytes lower than the number of supplied bytes, until the end of test, when the sending rate was set to 5 Gb/s. However, the maximum actual sending rate according to the number of sent bytes reported by `sendto()` calls was 3.88 Gb/s for 16114-byte packets. The higher possible sending rate than the receiving rate is probably a consequence of more complex processing in the receiving protocol stack.

Feng et al. [4] also conducted performance measurements with the Intel PRO10GbE LR adapters and achieved preliminary throughput of 7.2 Gb/s with Itanium-II machines. However, they performed most of their measurements with slower machines and they suggest that the likely bottleneck was the host software's ability to move data between components in the system.

7 Conclusion

We achieved TCP throughput of 1.3 Gb/s with 1500-byte packets and 2.6 Gb/s with 16114-byte packets on Dell 2650 servers. Interrupt coalescing did not work effectively with 16114-byte packets on the sender side causing the CPU load to rise to 100%. If this problem is resolved, the achieved throughput could probably be higher. However, the use of jumbo packets (larger than 1500 bytes) is required to achieve throughput significantly higher than 1 Gb/s.

We found that the maximum acceptable power budget of the Intel PRO/10GbE LR adapter is approximately 7.85 dB, which should allow communication up to approximately 22 km. Interconnection with 1550 nm transceiver on Cisco Catalyst switch worked without problems. This enables an end-to-end 10 Gigabit Ethernet communication over a wide-area network (although not at the full speed with current PCs).

The adapters could be also used to build a relatively inexpensive 10 Gigabit Ethernet router or switch, with multiple adapters in one PC running proper routing or switching software. With current PCs, the throughput would be however much lower than the full line rate. Another possible use is for emulating fast long-distance networks with NIST Net network emulator.

Appendix A: Tuning HW performance

Performance guides for using some network adapters in Linux operating system suggest to type command similar to `setpci -d 8086:1048 e6.b=2e` without much explanation. This command increases the PCI-X burst transfer size from the default value of 512 bytes to 4096 bytes by writing to the adapter MMRBC register. We can list PCI devices installed in the PC in a tree-like diagram by `lspci -t -v` command. In our case one device listed was `+--1d.0-[02]----02.0 Intel Corp.: Unknown device 1048`, indicating that it was installed on PCI bus number 2 and it was device number 2 on that bus. To find its vendor ID and device ID assigned by the manufacturer, you can use command

“`lspci -s 2:2 -n`”. In our case, it printed “02:02.0 Class 0200: 8086:1048 (rev 01)”. And these are the two magic numbers that can then be used with `setpci` command. Another way to find vendor ID and device ID is by reading file `/proc/pci`, which in our case included: “Bus 2, device 2, function 0: Ethernet controller: PCI device 8086:1048 (Intel Corp.)”. In order to check that the specified memory location was set to the specified value, the device address space can be printed in hexadecimal by command “`lspci -s 2:2 -xxx`”.

References

- [1] Intel PRO/10GbE LR driver, Intel corporation, http://downloadfinder.intel.com/scripts-df-external/Product_Filter.aspx?ProductID=943
- [2] Sven Ubik, Pavel Cimbali. *Achieving reliable high performance in LFNs*, Terena Networking Conference 2003, Zagreb, 19.-23. May 2003, Zagreb, Croatia.
- [3] RUDE/CRUDE utilities, <http://rude.sourceforge.net>.
- [4] Wu-chun Feng et al. *Optimizing 10-Gigabit Ethernet for Networks of Workstations, Clusters and Grids: A Case Study*, SuperComputing 2003, November 15-21, Phoenix, Arizona, USA.