

10 Gigabit Ethernet Routing on Linux

Martin Pels <martin@rodecker.nl>

Revision 1.4 – January 22, 2007

1 Introduction

Routing 10 Gigabit on the Internet is still very expensive. Professional hardware is available, but for a high price. If you need to push large amounts of traffic, but only have a small budget you are out of luck. But, there is an alternative...

It is possible to build a low-cost Linuxrouter with 10-GE network cards. The big question here is of course if this kind of device is actually able to approach the 10 Gigabit/s. The guys at the Amsterdam Internet Exchange[1] NOC kindly offered to hook the device up to the equipment in their lab, and run some tests on it. This document describes the results of these tests.

2 The basics

The machine used for the tests is a Dell PowerEdge 1950 with two dual core Intel Xeon 64bit CPU's at 2.33GHz and 2GB of RAM. For the networking two Myricom 10GE PCI-Express NICs[2] were used. The operating system used is Debian Etch, running a custom kernel 2.6.19.1. For the Myricom cards version 1.0.0 of the driver was used, which comes standard with a vanilla 2.6.19.1-kernel. Both NICs use firmware version 1.4.10, which was the most recent release at the time the tests were performed.

3 Performance tuning

Earlier trials showed that the bottleneck of a Linuxrouter is the number of interrupts it can handle. The more interrupts a device can process, the more frames per second it can push. To increase the number of interrupts the device can handle, hardware capable of using MSI (Message Signalled Interrupts) instead of legacy interrupts was used. In addition to this the box runs a recent Linux-kernel, capable of sharing interrupts among CPU cores. This means it is not limited to binding a NIC to a single CPU core, which is great for scalability. In this test setup the interrupts for each NIC are handled by two CPU cores.

The kernel used for this test was specifically configured for the purpose, disabling all features that might have a negative influence on the performance of the device (firewalling, connection tracking, etc.)

The following sysctl settings were used, as recommended by Myricom:

```
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.core.netdev_max_backlog = 250000
```

Since the testing was to be done at the AMS-IX lab some additional sysctl settings were used, which are recommend[3] for Linux-routers on the exchange:

```
net.ipv4.neigh.eth2.base_reachable_time=14400
net.ipv4.conf.eth2.arp_ignore=1
net.ipv4.conf.eth2.arp_announce=1
net.ipv4.neigh.eth3.base_reachable_time=14400
net.ipv4.conf.eth3.arp_ignore=1
net.ipv4.conf.eth3.arp_announce=1
```

4 Test setup

AMS-IX kindly offered to test the router in their lab, where they have equipment for doing RFC2544[4] tests. From this RFC the following tests were performed.

Throughput Establishes the maximum number of frames per second a device can process for different framesizes.

Latency The average latency the device has for different framesizes.

Both tests were done unidirectional and bidirectional.

The testing equipment used has a mandatory setting for loss tolerance. When this is set to 0.0 this means the test will fail if there is any packetloss. When set to 0.1 the test will fail if there is more than 0.1 percent frameloss. Both the unidirectional and bidirectional tests were done with a loss tolerance setting of 0.0% and 0.1%.

5 Results

Running the above tests produced the following results.

5.1 Bidirectional tests

The first test was the bidirectional throughput test, with a loss tolerance setting of 0.0 percent. Unfortunately, some frames were lost during this test. Because the tolerance was set to 0.0 the test stopped, without providing any results.

Next up was the same test, with a loss tolerance setting of 0.1 percent. This time there were results. Figure 1 shows the number of processed frames for a variety of framesizes. Clearly visible is that the bottleneck for this test was the number of frames per second. The device was not able to route more than 700.000 frames/second, regardless of the framesize. As figure 2 shows, at a framesize of 1518 bytes this means the maximum amount of traffic the device can route is about 8 Gigabit/second, or 4 Gigabit/s in each direction. For smaller framesizes this number drops linearly.

The results of the latency test are less straightforward. As figure 3 shows the latency for different framesizes varies greatly, and a pattern can not really be distinguished. This is most likely related to the way a Linuxsystem works. Since even on a stripped down Linux there are various processes running, the system is not busy with routing packets 100 percent of the time. Because the switch to a different process can happen at any time this may lead to seemingly random amounts of latency. Fortunately, the latency is always below one thousandth of a second, which is acceptable.

5.2 Unidirectional tests (1)

After the bidirectional tests the unidirectional tests were ran, to find out if there is any difference in the performance if packets are only sent one way. These tests were first tried with a loss tolerance of 0.1 percent. Figures 4, 5 and 6 show the results of these tests.

The results are pretty similar to those of the bidirectional test. Again, the throughput is roughly 700.000 frames/s or 8 Gigabit/s at a framesize of 1518 bytes. The latency test again

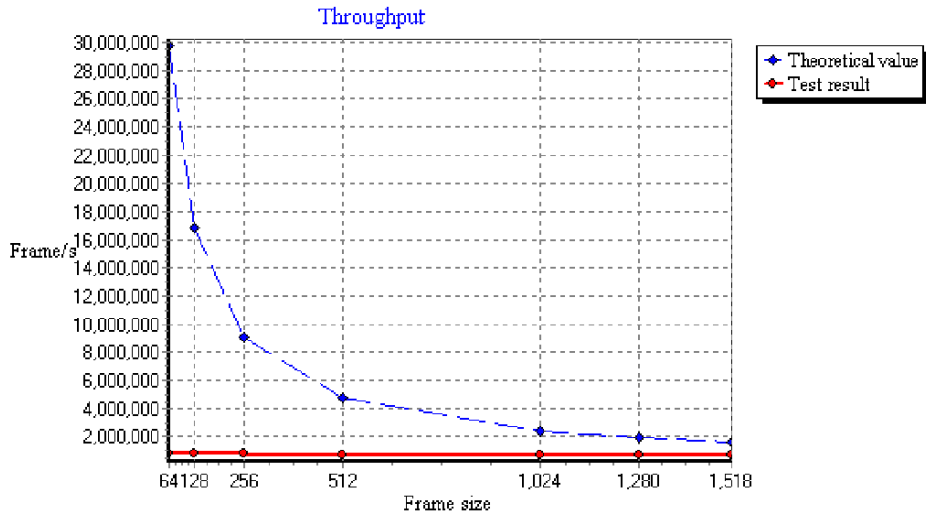


Figure 1: Bidirectional test - Throughput (frames/s)

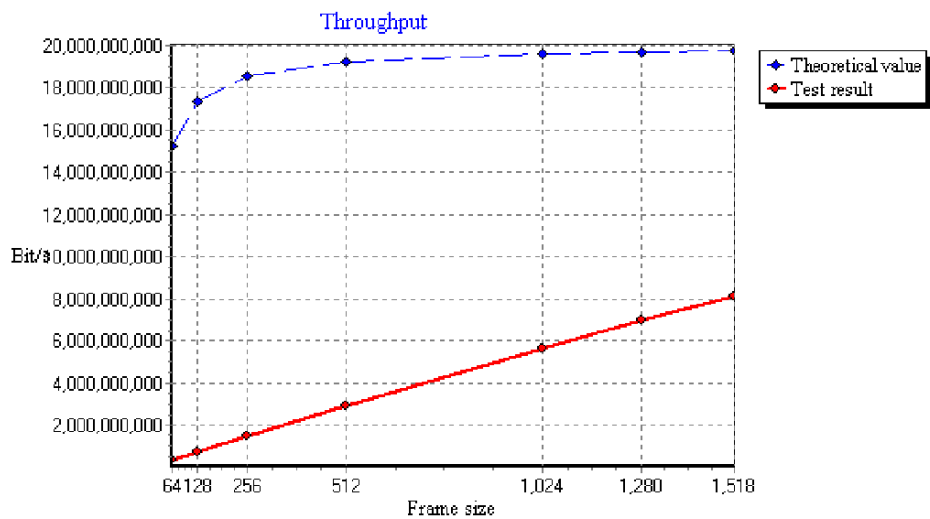


Figure 2: Bidirectional test - Throughput (bits/s)

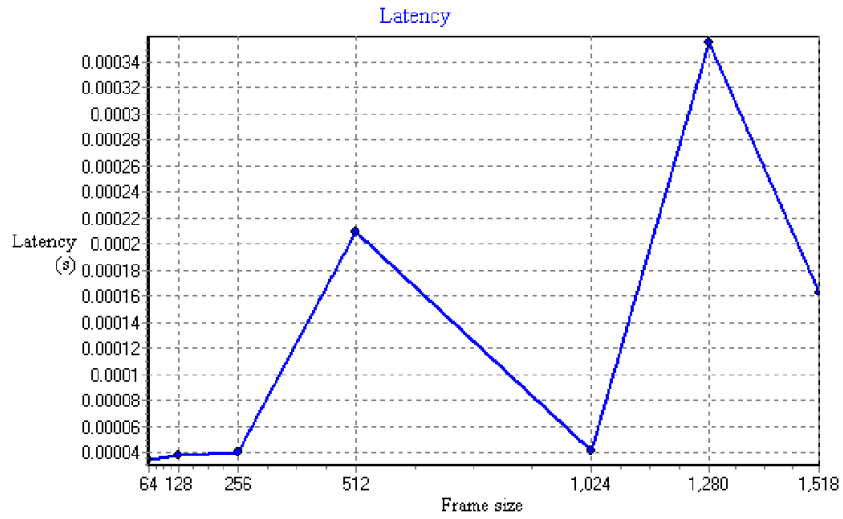


Figure 3: Bidirectional test - Latency

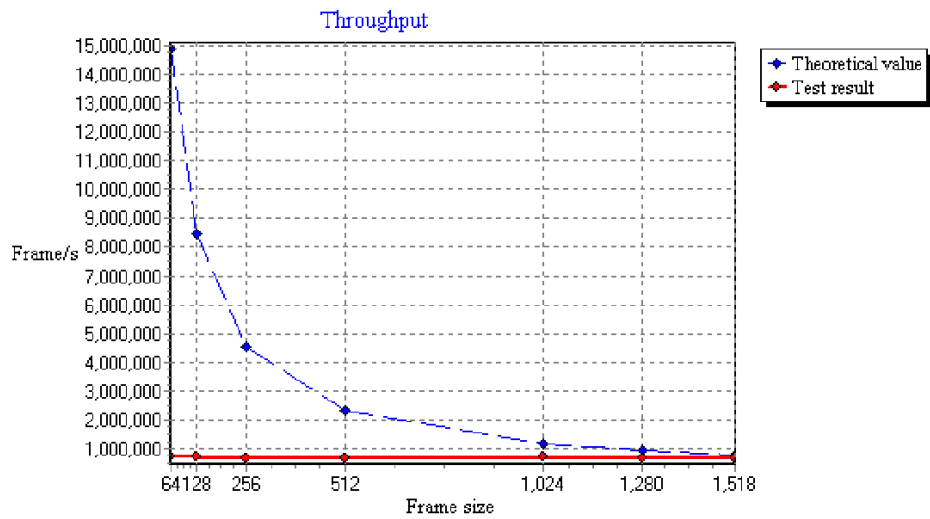


Figure 4: Unidirectional test 1 - Throughput (frames/s)

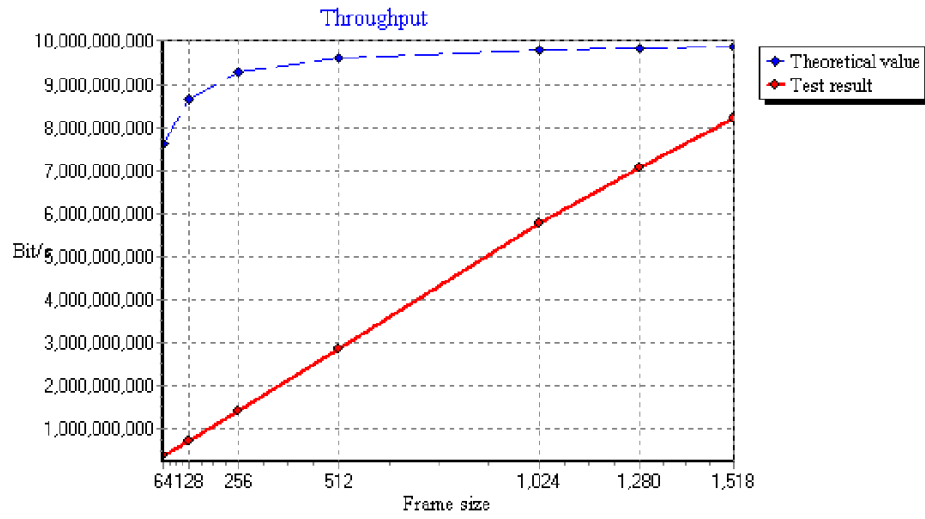


Figure 5: Unidirectional test 1 - Throughput (bits/s)



Figure 6: Unidirectional test 1 - Latency

shows low values. It seems the latency is lower for bigger framesizes. Considering the results of the bidirectional test his may be coincidental.

5.3 Unidirectional tests (2)

A second unidirectional test was done, this time with the loss tolerance set to 0.0 percent. In contrary to the bidirectional test, the unidirectional test did not fail on this setting. Figures 7, 8 and 9 show the results.

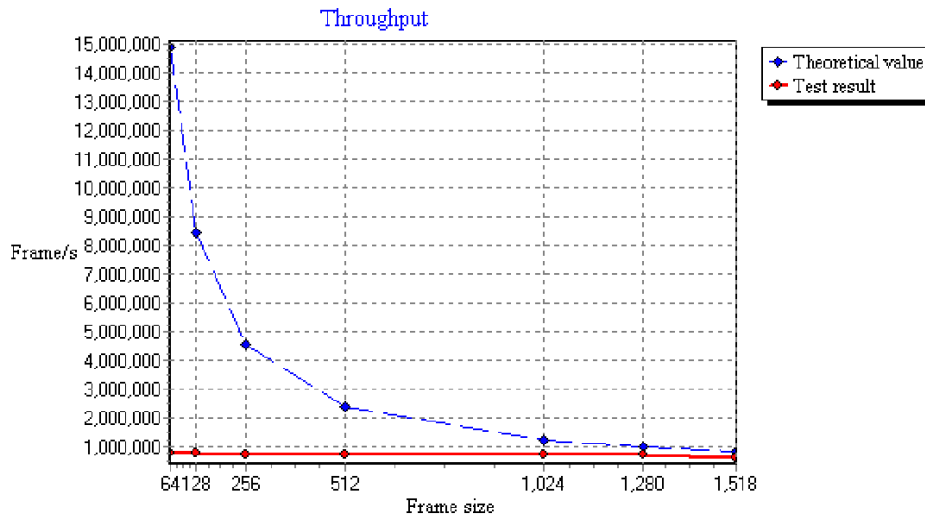


Figure 7: Unidirectional test 2 - Throughput (frames/s)

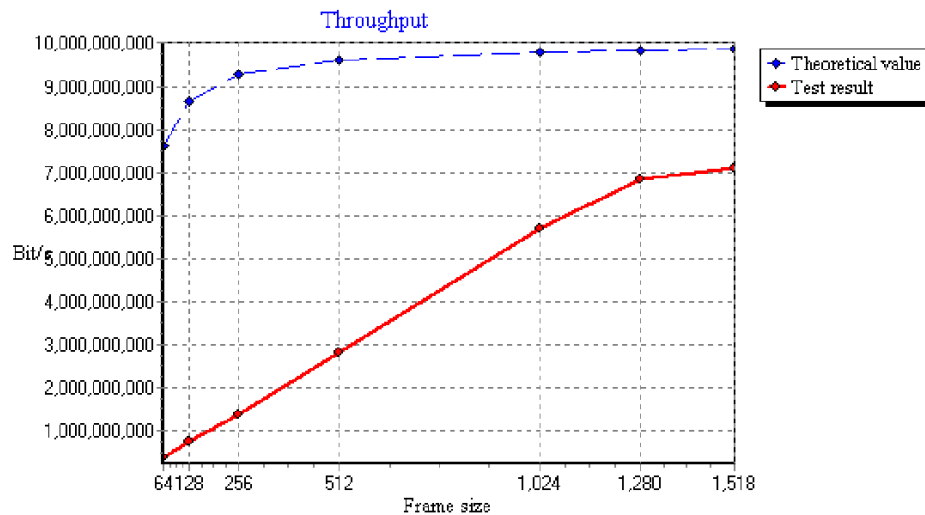


Figure 8: Unidirectional test 2 - Throughput (bits/s)

Again, the results of the throughput test are quite similar to those of the previous tests, except for the results for a framesize of 1518 bytes. Here the throughput is only about 550.000 frames/s, or 7 Gigabit/s. It is unclear what the cause of this large difference is, compared to the results of the first unidirectional test. The results of the latency test correspond with the previous assumption that there is no relationship between the used framesize and the amount of latency, given that the test results are again seemingly random.

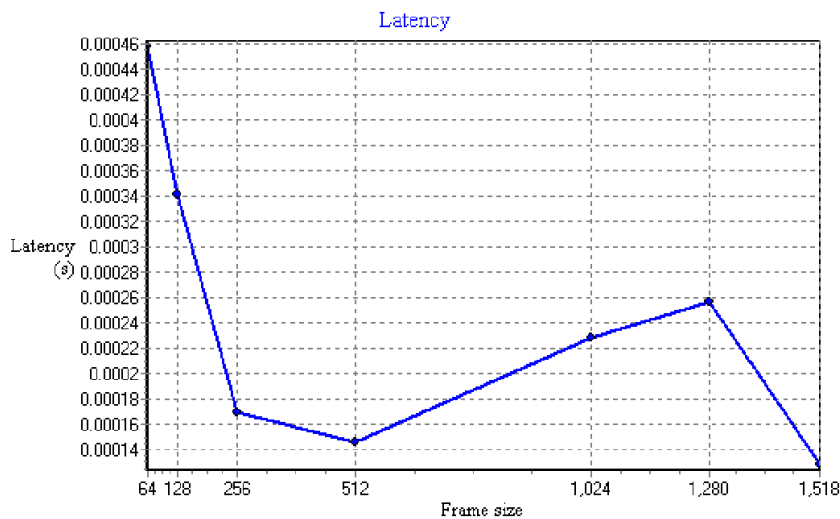


Figure 9: Unidirectional test 2 - Latency

6 Conclusions

Obviously, the above results are in no way conclusive on how well Linux performs as a 10 Gigabit Ethernet router. In a real-world situation the device might be running BGP, with a full routing table. This will surely affect the performance of the device. Unfortunately, with the used test setup it was impossible to determine exactly how large this performance difference is. The results however do give a nice estimate. It is good to have some indication of what to expect from a device like this in terms of performance, when using it in a real-world situation.

The results show two two worrying things that could influence the decision of using this type of device in the real world. The first is the frameloss in the bidirectional test. A frameloss of 0.1% at 700.000 frames/s is downright unacceptable. Fortunately, in reality the loss was much lower than the tolerance limit used in this test. Still, it was never zero, which could certainly lead to problems in some situations.

The second issue to worry about is the limit on the number of frames per second. This relatively low limit poses a Denial of Service risk. Using frames with a size of 64 bytes, it is possible to bring the device to its knees with a traffic flow of 358.4 Megabit/s, assuming a limit of 700.000 frames/s. A large botnetwork can easily generate this kind of traffic.

7 Final remarks

The results of these tests clearly show that Linuxrouters are still far from beating professional hardware, in terms of frame processing performance. For situations where a large number of small packets need to be pushed, or where even a small amount of packetloss is unacceptable, this type of device is not at all suitable. However, since the cost of a Linuxrouter is many times lower than that of a professional router, it is definitely an interesting option to look at if you have a large amount of frames with a big payload to push, but only have a small budget.

References

- [1] *Amsterdam Internet Exchange (AMS-IX)* , <http://www.ams-ix.net/>
- [2] *Myri-10G PCI-Express NIC with a 10GBase-R port* , <http://www.myri.com/Myri-10G/NIC/10G-PCIE-8A-R.html>

- [3] *AMS-IX Linux Configuration Hints* ,
https://www.ams-ix.net/technical/config-guide/linux_configuration_hints.html
- [4] *RFC2544: Benchmarking Methodology for Network Interconnect Devices* ,
<http://www.ietf.org/rfc/rfc2544.txt>