

Chelsio Communications: 10-gigabit Network Adapter Performance Study

Test report prepared under contract from Chelsio Communications

Executive summary

Chelsio Communications (“Chelsio”) commissioned VeriTest, a division of Lionbridge Technologies Inc., to audit a performance study conducted by Chelsio of four 10GbE network adapters. To conduct this performance audit, VeriTest sent a Test Engineer to the Chelsio facility in Sunnyvale, California. The VeriTest Test Engineer verified the testing environment, tool, product and server configurations, and product performance results.

The study evaluated the throughput, latency, and CPU utilization performance of the following four products:

Chelsio Communications
T110-SR 10Gbit Protocol
Engine
Chelsio Communications
N110-SR 10Gbit Server
Adapter
Intel PRO/10GbE SR Server
Adapter
S2io Technologies Xframe 10
Gigabit Ethernet PCI-X
Server/Storage Adapter

Key findings

- ❑ VeriTest verified that, on average, the Chelsio Communications T110-SR 10Gbit Protocol Engine had a throughput 121% higher on TX and 68% higher on RX than the Intel PRO/10GbE SR Server Adapter and 128% higher on TX and 85% higher on RX than the S2io Technologies Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter.
- ❑ VeriTest verified that the Chelsio Communications T110-SR 10Gbit Protocol Engine had up to 5 times the CPU efficiency of the Intel PRO/10GbE SR Server Adapter and up to 7 times the CPU efficiency of the S2io Technologies Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter
- ❑ VeriTest verified that, on average, the Chelsio Communications N110-SR 10Gbit Server Adapter had a throughput 25% higher on TX and 10% higher on RX than the Intel PRO/10GbE SR Server Adapter and 27% higher on TX and 16% higher on RX than the S2io Technologies Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter.
- ❑ Veritest verified that the 1 byte latency of the Chelsio Communications T110 10Gbit Protocol Engine was 14.6µs and the latency of the Chelsio N110 Server Adapter was 15.5µs, compared to the Intel PRO/10GbE SR Server Adapter with a latency of 51.3µs, and the S2io Technologies Xframe with a latency of 25.8µs.

The results show that Chelsio T110-SR protocol offload engine (TOE) reduces CPU consumption by a factor of 2 to 7 when compared to the other three network adapters, resulting in higher throughput and/or CPU efficiency. The T110 is capable of fully utilizing the available bandwidth while using 1,500 byte frames, with plenty of CPU cycles available to user applications in a one processor system configuration. In contrast, the performance of the other adapters is limited because of CPU saturation and/or of internal adapter limitations. The results show that the T110 adapter has higher throughput, lower latency, better CPU utilization, and a flatter bi-directional performance profile over a wide range of connections (1 to 1,000).

The study has also shown that within the category of non-offload adapters, the N110-SR server adapter has the best performance characteristics.

The study used the standard network performance tools Netperf to measure latency and lperf, to measure throughput and CPU utilization for TCP traffic. Additional information on Netperf is available at

<http://www.netperf.org/netperf/NetperfPage.html>. Additional information on Iperf is available at <http://dast.nlanr.net/Projects/Iperf>.

VeriTest audited all the performance tests and they certified that the test configurations and execution were conducted in a fair and consistent manner. The experimental setup consisted of two identical single processor AMD Opteron 248 2.2GHz server systems configured with 1GB of PC3200 RAM, an 80GB IDE hard disk drive running RedHat 9.0 (2.6.6). One system housed each of the network adapters as testing was performed, and the other was equipped with a T110 Protocol Engine to serve as a peer. The two server systems were connected through a Foundry Networks NetIron 40G switch.

The following tests were conducted for all four products using the same experimental setup:

Transmit tests for 1, 10, 100 and 1000 connections to collect throughput and CPU utilization numbers, and varying the following parameters:

- MTU: 1500 and 9000 byte frames.
- I/O size: 256, 512, 1024, 4096, 8192, 16384, 32768 and 65536 bytes.

Receive tests for 1, 10, 100 and 1000 connections to collect throughput and CPU utilization numbers, and varying the following parameters:

- MTU: 1500 and 9000 byte frames.
- I/O size: 256, 512, 1024, 4096, 8192, 16384, 32768 and 65536 bytes.

Latency test with 1 connection using I/O sizes of 1, 512, and 1400 bytes

Throughout this study, the measure of throughput reported is the TCP “**goodput**” provided by Iperf. Goodput *does not include the header overhead associated with Ethernet, IP and TCP*. Due to the PCI-X 133 MHz bus’ limitations, the maximum goodput achievable with 1,500 byte frames is about 7.5 Gbps, corresponding to 7.9Gbps on the wire.

In order to normalize the CPU utilization among the different adapters, the results are mapped to a CPU efficiency measure defined as the throughput achieved divided by the CPU utilization, and presented in units of *Megabit per percent CPU (Mbps/%CPU)*.

We also note that the latency numbers presented include the switch latency in addition to the adapter latencies.

The following is a summary of the main findings:

Transmit Tests

In the transmit tests with 1,500 byte frames and varying the number of connections, the average throughput (goodput) of the Chelsio Communications T110 10Gbit Protocol Engine was 7.12Gbps, or 121% higher than the Intel PRO/10GbE SR Server Adapter (3.21Gbps) and 129% higher than S2io Technologies Xframe (3.11Gbps). The CPU utilization was measured on the host server during the tests and the CPU efficiency index was computed for each adapter. The results show that the CPU efficiency of the Chelsio Communications T110 10Gbit Protocol Engine was on average 4.3 times that of the Intel PRO/10GbE SR Server Adapter, and 4.8 times that of the S2io Technologies Xframe. Additionally, on average, the throughput Chelsio Communications N110 10Gbit Server Adapter was 4.29Gbps, or 25% higher than the Intel PRO/10GbE SR Server Adapter and 27% higher than the S2io Technologies Xframe.

In the transmit tests with 9,000 byte frames and varying the number of connections, the average throughput of the Chelsio Communications N110 10Gbit Server Adapter was 6.9Gbps, or 43% higher than the Intel PRO/10GbE SR Server Adapter (4.98Gbps) and 63% higher than S2io Technologies Xframe (4.36Gbps). Since the T110 fully offloads the TCP/IP stack, and offers the benefits of jumbo Ethernet frames while retaining those of using standard frames on the wire, it was not included in this test.

In transmit tests with 1,500 byte frames and varying the I/O size, the average throughput of the Chelsio Communications T110 10Gbit Protocol Engine was 7.35Gbps, or 114% higher than the Intel PRO/10GbE Server Adapter (3.32Gbps), and 110% higher than S2io Technologies Xframe (3.39Gbps). Additionally, the average throughput of the Chelsio Communications N110 10Gbit Server Adapter throughput was 4.92Gbps, or 32% higher than the Intel PRO/10GbE Server Adapter, and 31% higher than the S2io Technologies Xframe.

Receive Tests

In the receive tests with 1,500 byte frames and varying the number of connections, the average throughput of the Chelsio Communications T110 10Gbit Protocol Engine was 7.08Gbps, or 68% higher than the Intel PRO/10GbE SR (4.24Gbps) and 85% higher than S2io Technologies Xframe (3.84Gbps). The CPU utilization was measured on the host server during the tests and the CPU efficiency index was computed for each adapter. The results show that the CPU efficiency of the Chelsio Communications T110 10Gbit Protocol Engine was on average 3.5 times that of the Intel PRO/10GbE SR Server Adapter, and 3.8 times that of the S2io Technologies Xframe

Additionally, the average throughput of the Chelsio Communications N110 10Gbit Server Adapter throughput was 4.71Gbps, or 10% higher than the Intel PRO/10GbE SR Server Adapter and 18% higher than the S2io Technologies Xframe.

In the receive tests with 9,000 byte frames and varying the number of connections, the average throughput of the Chelsio Communications N110 10Gbit Server Adapter was 7.47Gbps, or 33% higher than the Intel PRO/10GbE SR Server Adapter (5.36Gbps) and 16% higher, on average, than S2io Technologies Xframe (6.15Gbps). Since the T110 fully offloads the TCP/IP stack, and offers the benefits of jumbo Ethernet frames while retaining those of using standard frames on the wire, it was not included in this test. The Intel Corp. and S2io Technologies adapters reach their peak performance of 5.87Gbps and 6.94Gbps respectively in the receive direction only with more than 10 connections and using jumbo frames. Their performance in the transmit direction is noticeably lower.

In the receive tests with 1,500 byte frames and varying the I/O size, the average throughput of the Chelsio Communications T110 10Gbit Protocol Engine was 7.16Gbps, or 91% higher than the Intel PRO/10GbE SR (3.72Gbps) and 160% higher than S2io Technologies Xframe (2.73Gbps). Additionally, the average throughput of the Chelsio Communications N110 10Gbit Server Adapter throughput was 4.54Gbps, or 18% higher than the Intel PRO/10GbE SR Server Adapter and 40% higher than the S2io Technologies Xframe.

Latency Tests

In order to determine the application-to-application latency, we ran a Netperf TCP Request Response (TCP_RR) transaction test with 1 connection and used the following formula to calculate the one way latency:

$$\text{Latency} = ((1 / \text{transaction rate}) / 2)$$

Note that the latency figures obtained include the latency of both the sending and receiving adapters and that of the switch through which they are connected.

VeriTest observed the application-to-application latency of the Chelsio Communications T110 10Gbit Protocol Engine to be 14.6 μ s, or 28% of the latency of the Intel PRO/10GbE SR Server Adapter (51.3 μ s) and a 56% of S2io Technologies' Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter (25.8 μ s).

Additionally, VeriTest verified the performance tests of Chelsio's N110 product compared to 10Gb Ethernet products from Intel Corp and S2io Inc. These tests reported that the N110 consistently delivered a 1 byte latency of 15.5 μ s, which is 30% of the Intel's network adapter latency and 60% of the S2io's network adapter latency.

Conclusions

The test results established that the T110 consistently and clearly has the best throughput and CPU efficiency across the range of tests. The Intel server adapter's throughput seems to be limited by the server adapter, as it stays well below the maximum even when CPU cycles are available. The Chelsio N110 and S2IO Xframe are able to send more data if the cycles are available, and therefore are bound by the available processor power in the system, particularly when using standard Ethernet frames. However, the N110 shows superior throughput and latency performance, and better processor efficiency. The Intel Corp. and S2io Technologies adapters reach their peak performance only when using jumbo frames and in the receive direction. Their performance in the transmit direction is noticeably lower, and falls dramatically when using standard Ethernet frames.

In addition, Chelsio Communications, Inc.'s T110 and N110 adapters by far exhibit the lowest application-to-application latency. The best performing adapter in the latency tests was again the T110, with a latency 28% that of the Intel Corp.'s Pro/10000 adapter and 56% that of S2io Technologies' Xframe adapter. The N110 Server Adapter was the next best performing adapter, with slightly higher latency than the T110.

Results Summary

Chelsio commissioned VeriTest to independently verify a performance study of four following 10GbE adapters:

- Chelsio Communications T110 10Gbit Protocol Engine
- Chelsio Communications N110 10Gbit Server Adapter
- Intel PRO/10GbE SR Server Adapter
- S2io Technologies Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter

For this study, two identical AMD Opteron 248 2.2GHz server systems configured with 1GB of PC3200 RAM, and an 80GB IDE hard disk drive running RedHat Linux 9.0 (Kernel 2.6.6 uniprocessor build) were used. One system contained each of the 10GbE network adapters under test and the other contained a Chelsio T110 Protocol Engine. The two servers were connected through a Foundry Networks NetIron 40G switch using two 10Gbps ports.

The network benchmarking tool Netperf v2.2pl4 was used to measure latency and Iperf 1.7.0 was used to measure the TCP throughput (goodput) and CPU utilization, while changing network parameters, such as the number of network connections, the I/O size, and frame size. Additional information on Netperf is available at <http://www.netperf.org/netperf/NetperfPage.html>. Additional information on Iperf is available at <http://dast.nlanr.net/projects/Iperf>.

Both tools have their strengths and weaknesses and are therefore needed to obtain the performance measures of interest here. Netperf provides the transaction test feature required for latency measurements but scales poorly as the number of connections is increased because it spawns a process for each connection. Iperf scales better because it spawns somewhat lighter weight threads instead but does not provide the transaction test feature.

We note that the latency figures obtained include the switch latency in addition to the latency through the two 10Gbps adapters.

The following sections contain the main measurement results.

Transmit Throughput Measurements

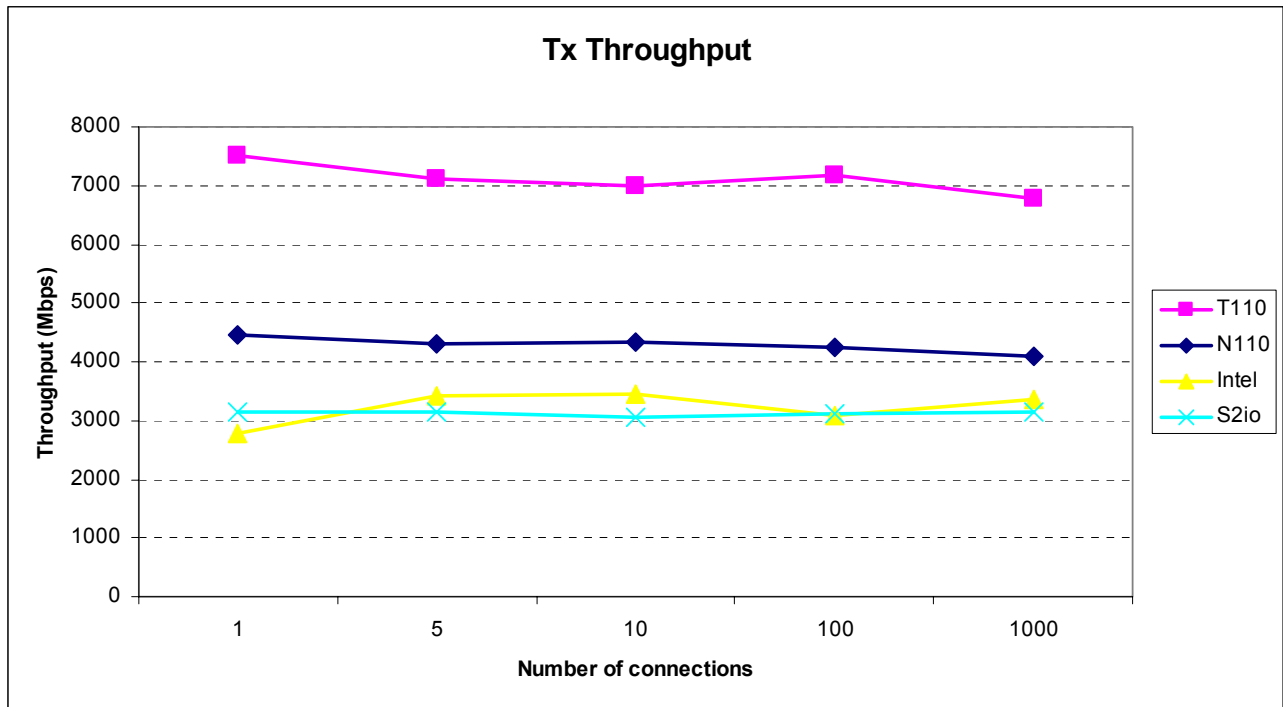


Figure 1a: Tx TCP goodput versus number of connections (1,500 byte frames)

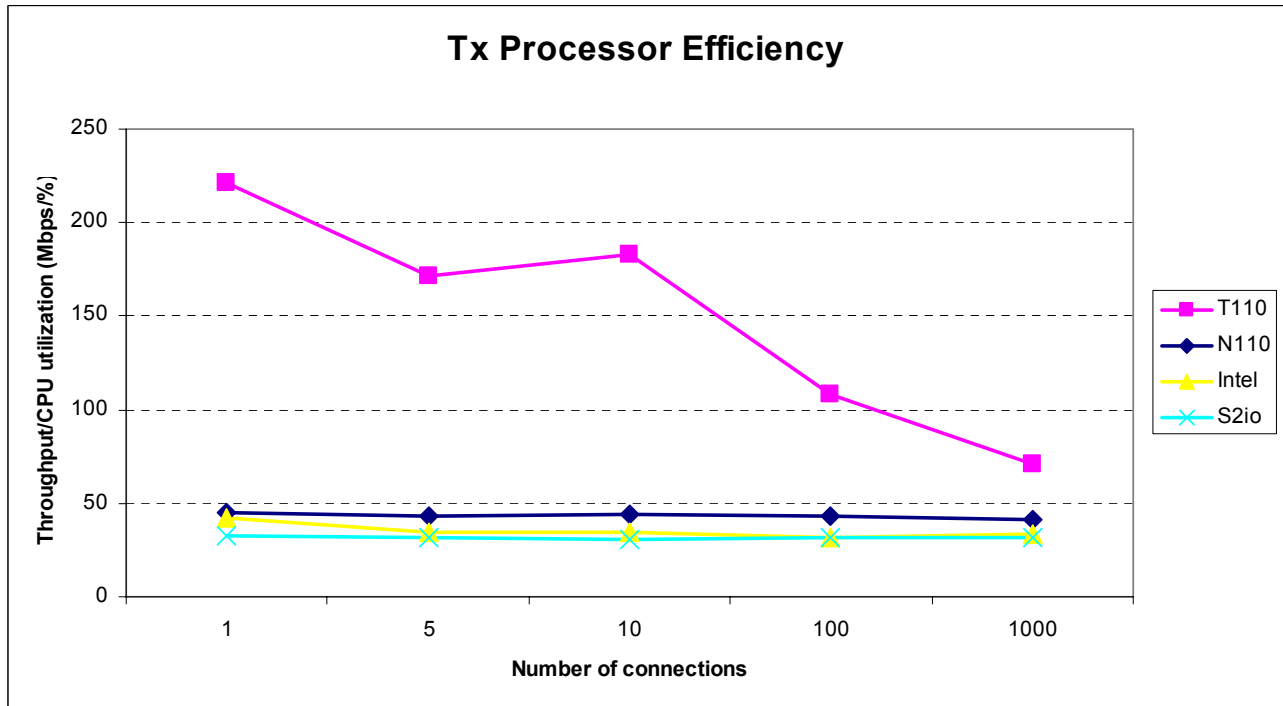


Figure 1b: Tx processor efficiency versus number of connections (1,500 byte frames)

As shown in Figures 1a and 1b above, the Chelsio Communications T110 10Gbit Protocol Engine had by far the largest transmit goodput across all number of connections from 1 to 1000. It also provided up to 6.9 times

the CPU efficiency (Mbps per %CPU) of the other adapters for tests with 1,500 byte frames. Among the non-offload adapters, Chelsio's N110 server adapter had the best performance and highest CPU efficiency.

The T110 had an average TCP goodput of 7.1Gbps across the range of connections. The N110 had an average TCP goodput of 4.3Gbps, the Intel PRO/GbE SR network adapter had an average TCP goodput of 3.2Gbps, and the S2io Xframe had an average TCP goodput of 3.1Gbps. Therefore, the T110 had an average TCP goodput that is 121% higher than the Intel PRO/GbE SR network adapter and 129% higher than the S2io Xframe. The N110 had an average TCP goodput that is 25% higher than the Intel PRO/GbE SR network adapter and 27% higher than the S2io Xframe.

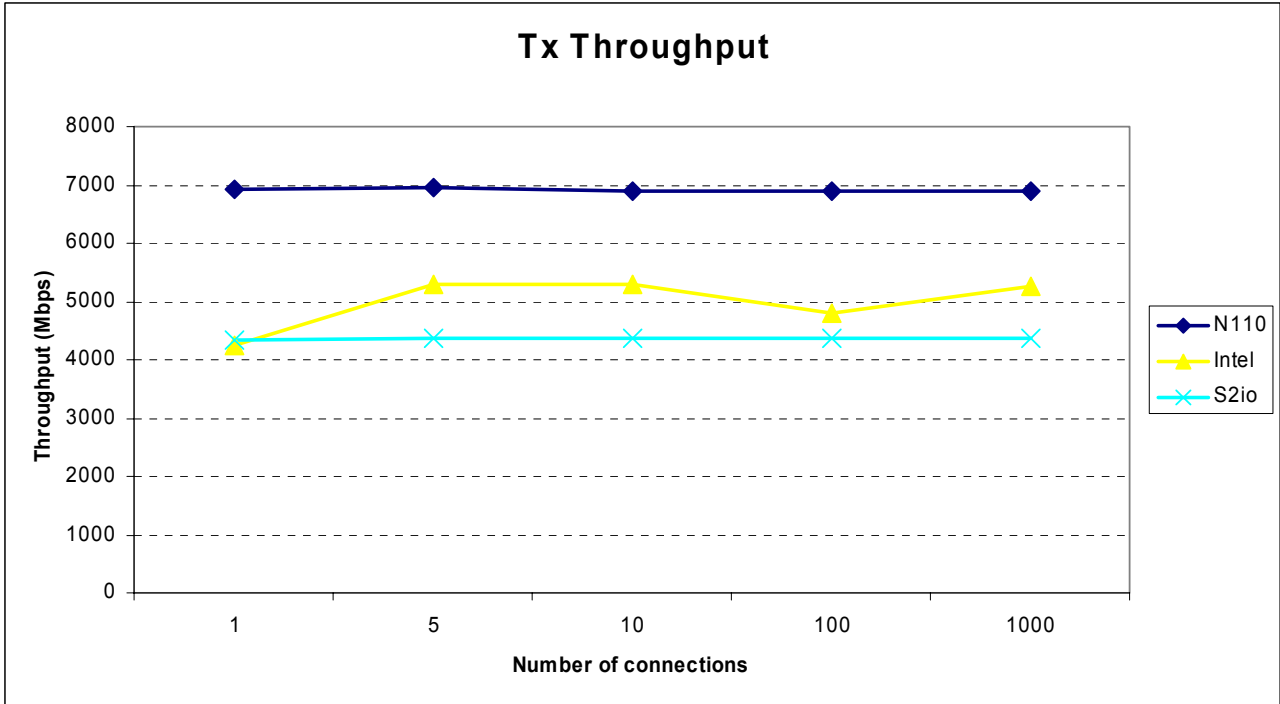


Figure 2a: Tx TCP goodput versus number of connections (9,000 byte frames)

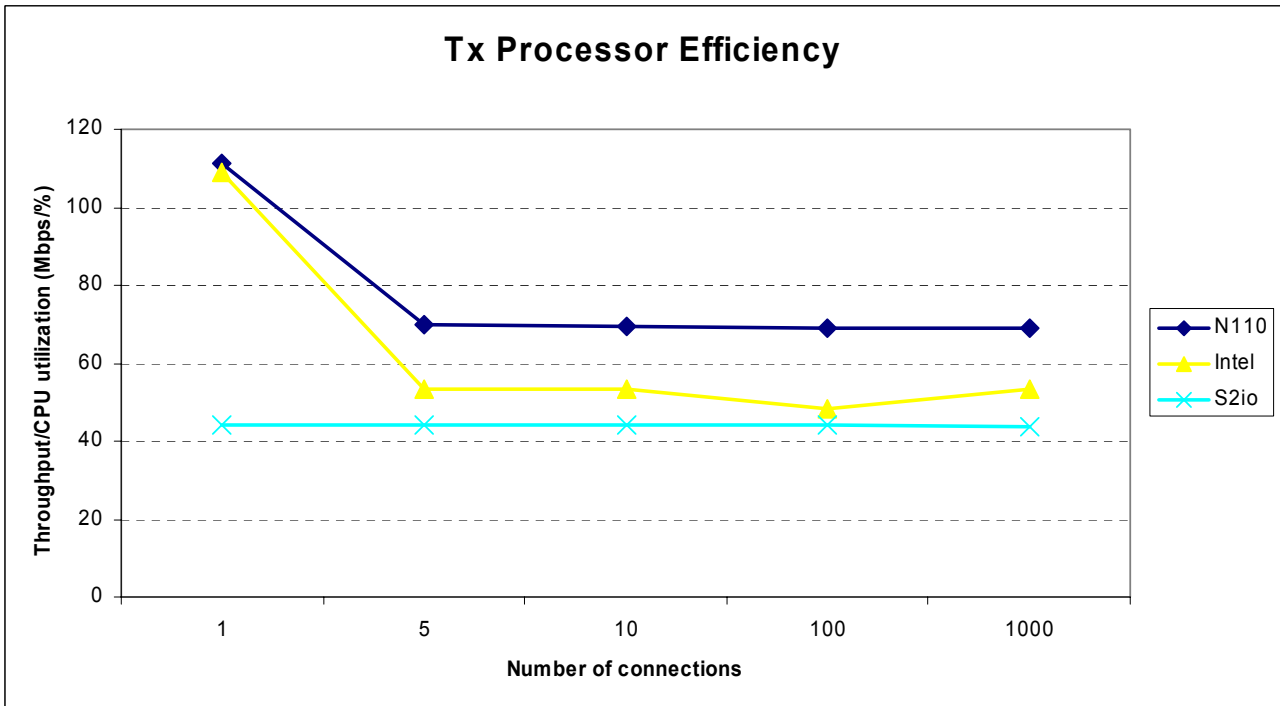


Figure 2b: Tx CPU efficiency versus number of connections (9,000 byte frames)

As shown in Figures 2a and 2b above, the Chelsio Communications N110 10Gbit Server Adapter had the highest transmit goodput while maintaining the best CPU efficiency index across all number of connections from 1 to 1000 for tests with 9000 byte frames.

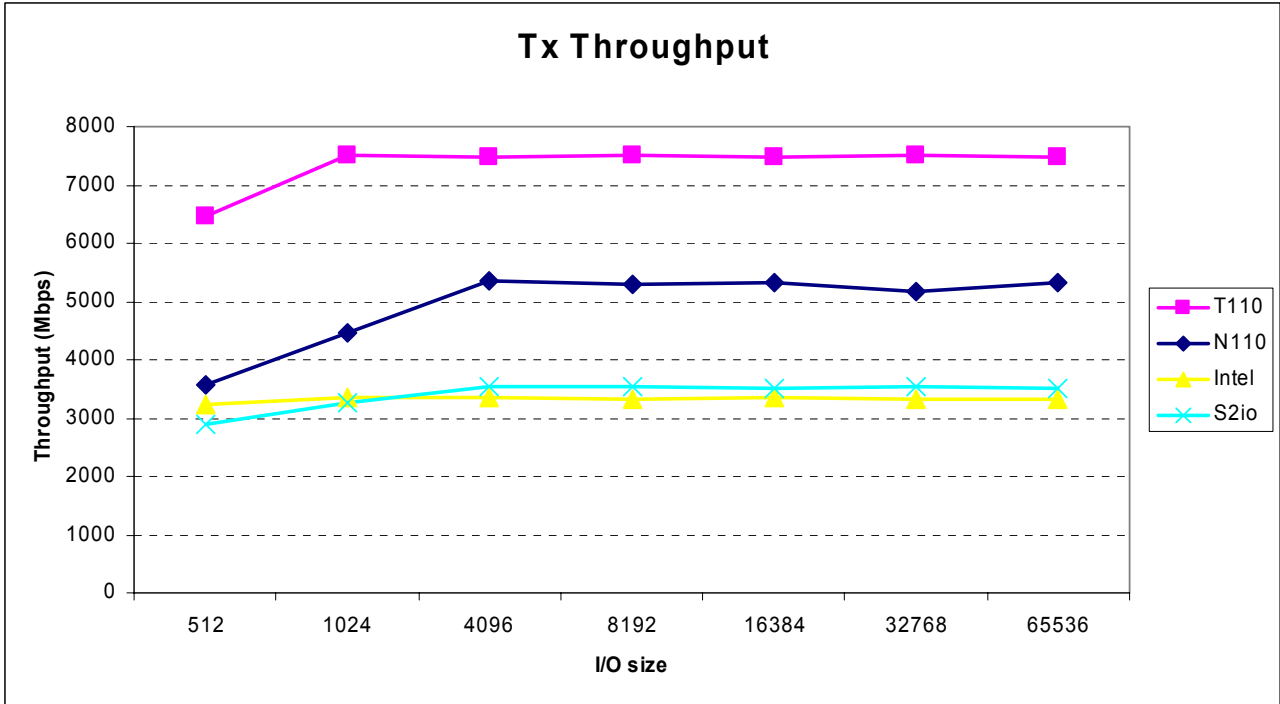


Figure 3a: Tx TCP goodput versus I/O size (1,500 byte frames)

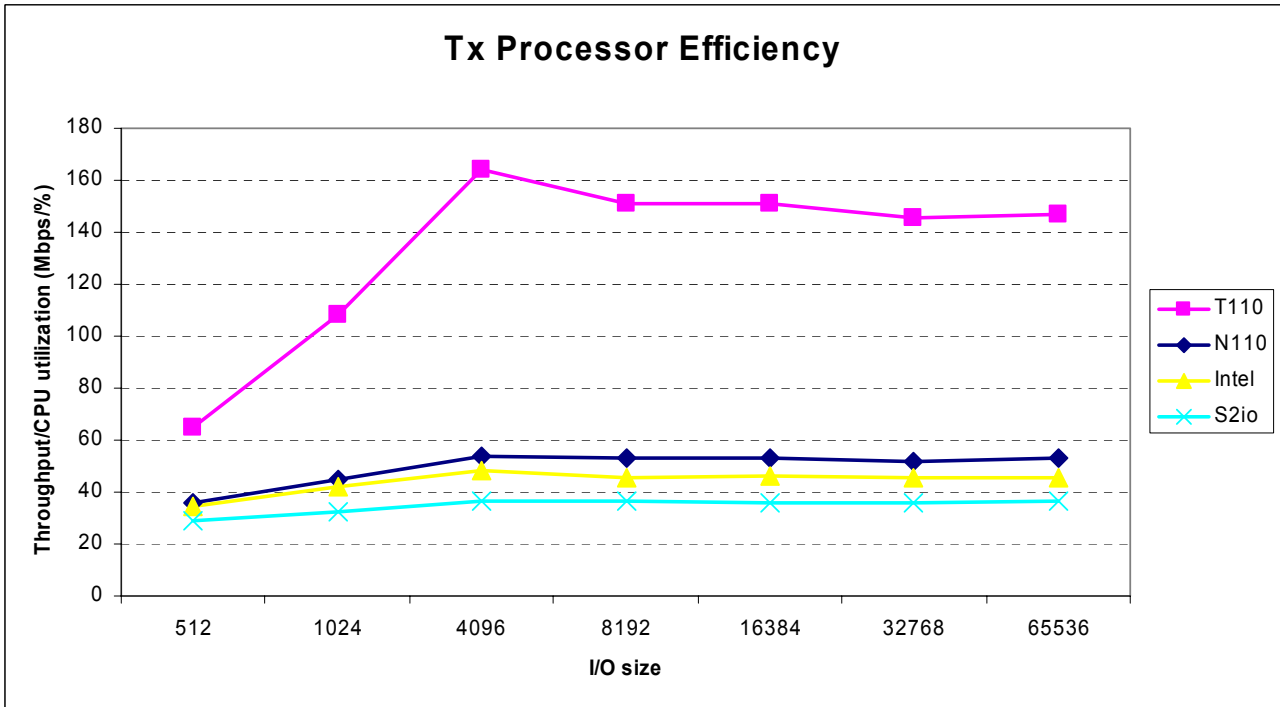


Figure 3b: Tx CPU efficiency versus I/O size (1,500 byte frames)

As shown in Figures 3a and 3b above, the Chelsio Communications T110 10Gbit Protocol Engine had the highest transmit goodput and the best CPU efficiency index across all I/O sizes from 512 bytes to 64KB for these tests with 1,500 byte frames. The T110's CPU efficiency on average was 3.4 times that of the Intel adapter and 4.3 times that of the S2io adapter. Among the non-offload adapters, Chelsio's N110 server adapter again had the best performance and highest CPU efficiency.

Receive Throughput Measurements

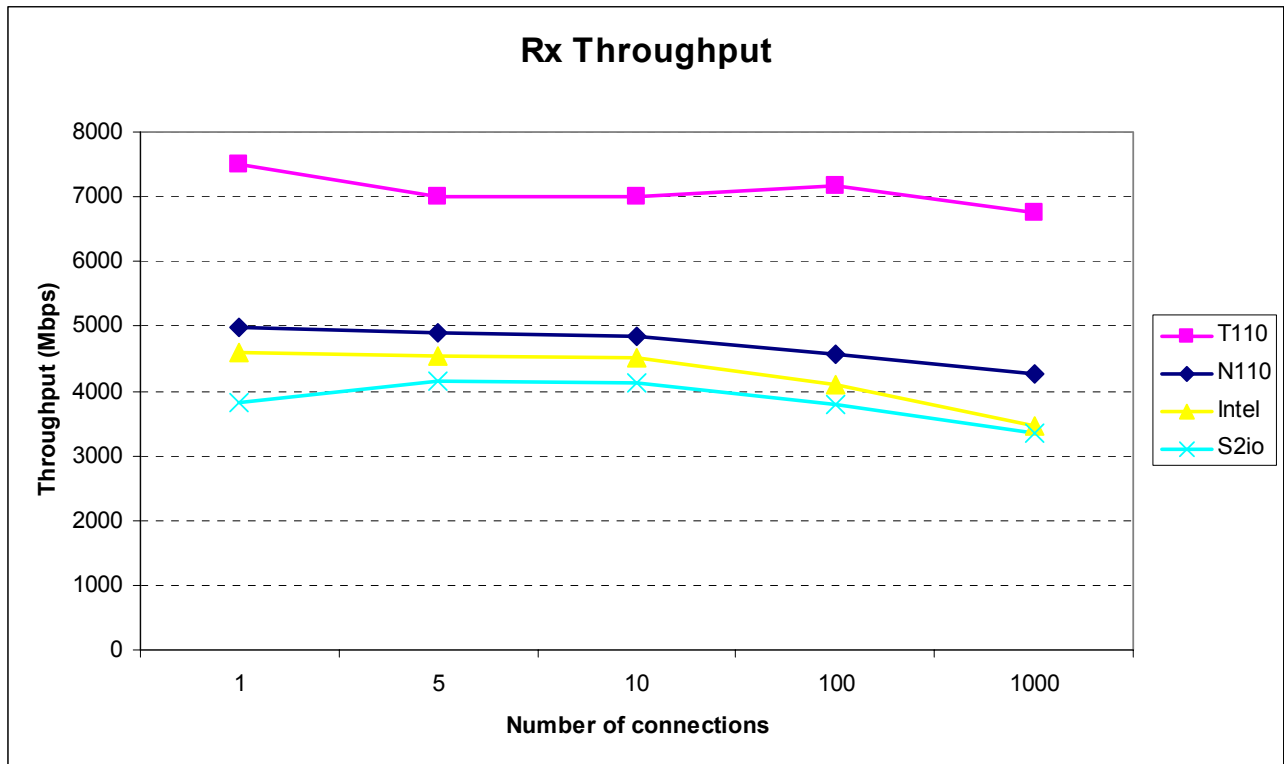


Figure 4a: Rx TCP goodput versus number of connections (1,500 byte frames)

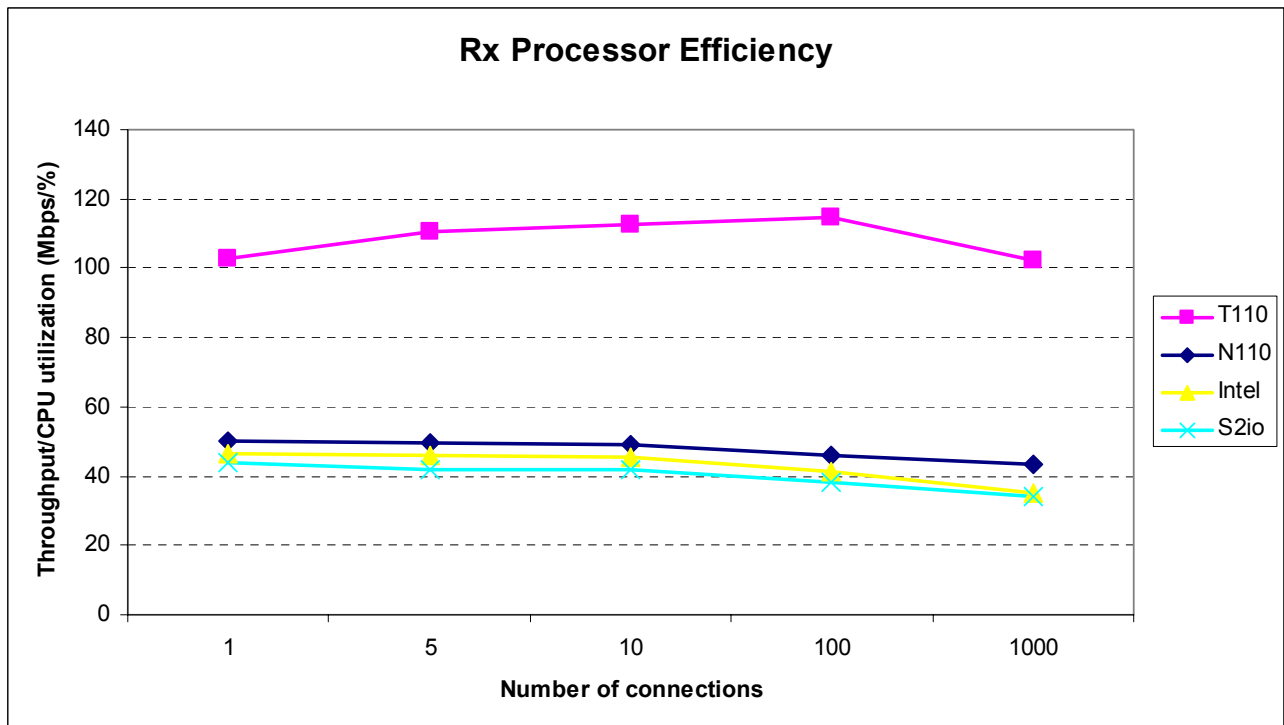


Figure 4b: Rx processor efficiency versus number of connections (1,500 byte frames)

As shown in Figures 4a and 4b above, the Chelsio Communications T110 10Gbit Protocol Engine had the greatest receive goodput across all number of connections from 1 to 1000. It also provided on average 2.7 times the CPU efficiency (Mbps per %CPU) of the other adapters. Similar to the Tx tests, among the non-offload adapters, Chelsio's N110 server adapter had the best performance and highest CPU efficiency.

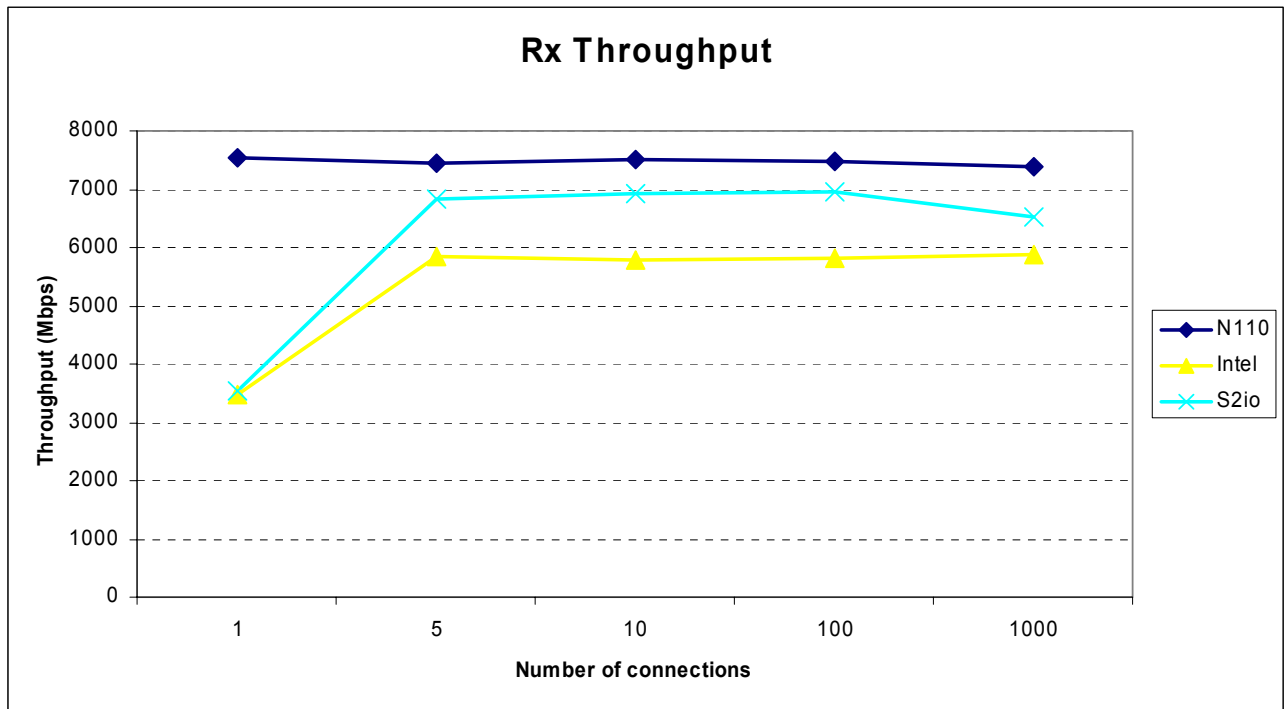


Figure 5a: Rx TCP goodput versus number of connections (9,000 byte frames)

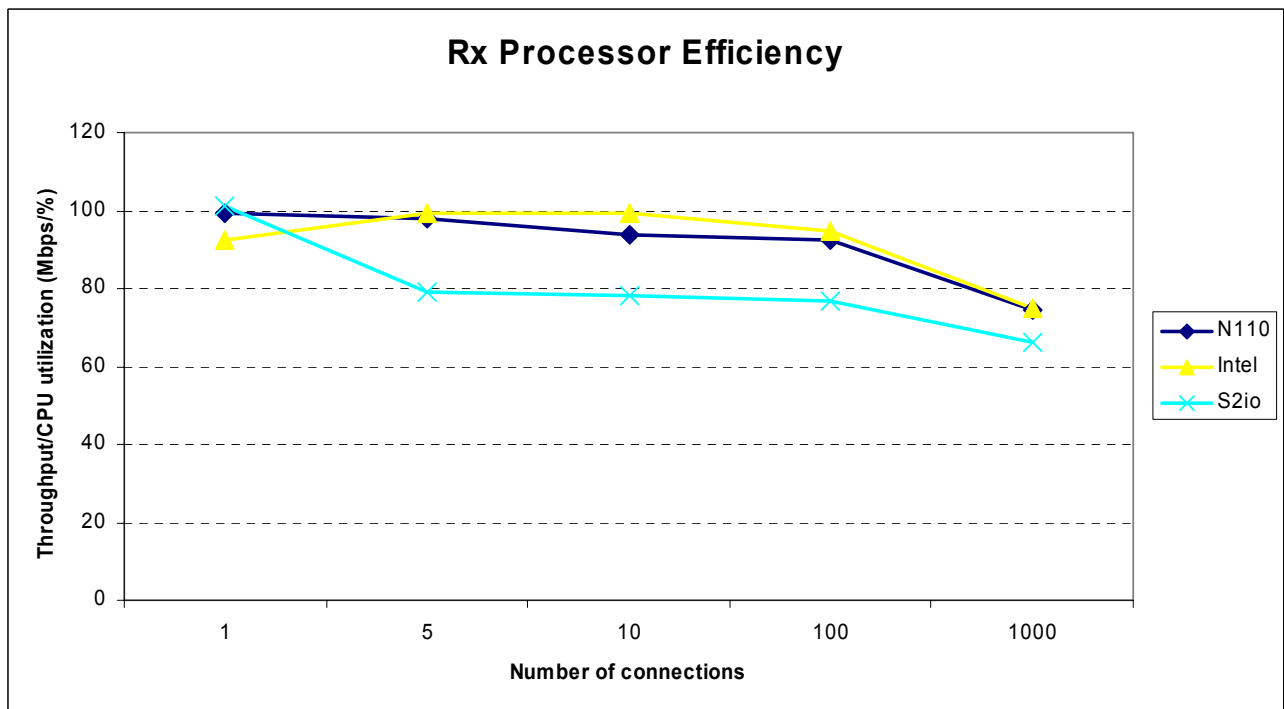


Figure 5b: Rx CPU efficiency versus number of connections (9,000 byte frames)

As shown in Figures 5a and 5b, the Chelsio Communications N110 10Gbit Server Adapter had the highest receive goodput and along with the Intel adapter the best CPU efficiency index across all number of connections from 1 to 1000 for tests with 9000 byte frames. However, the Intel adapter averages only 71% of the TCP goodput obtained by the N110.

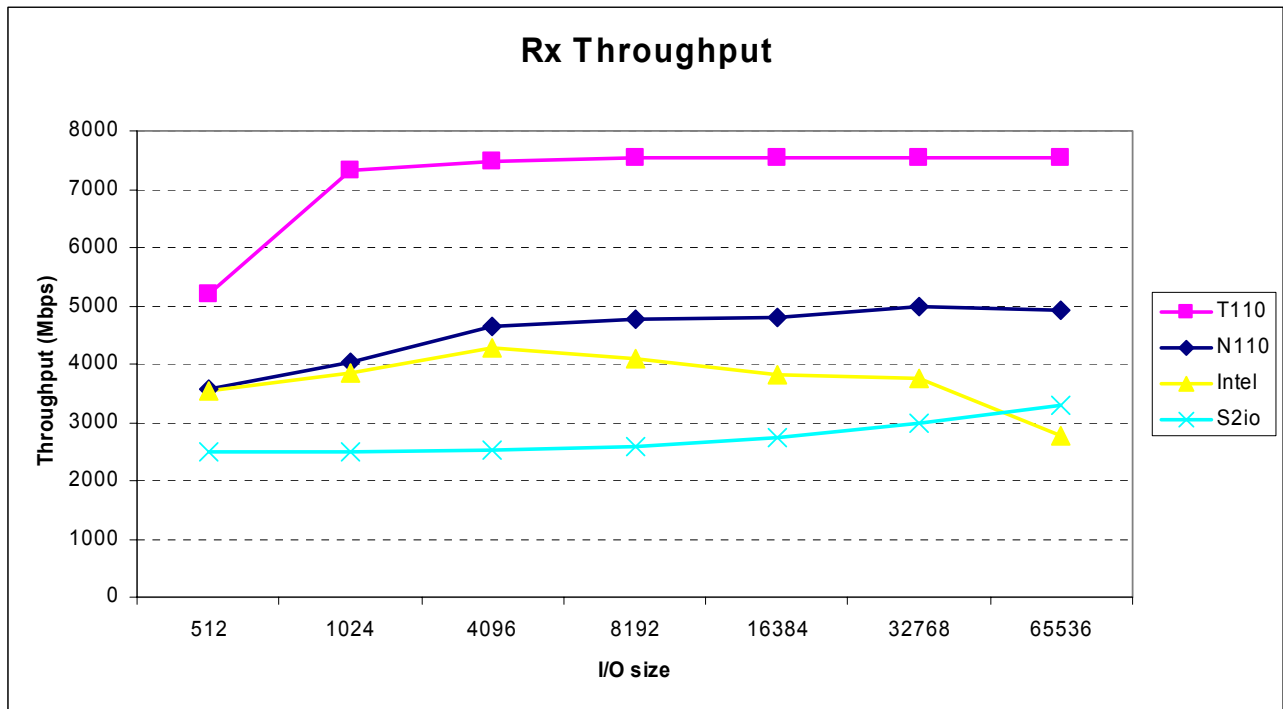


Figure 6a: Rx TCP goodput versus I/O size (1,500 byte frames)

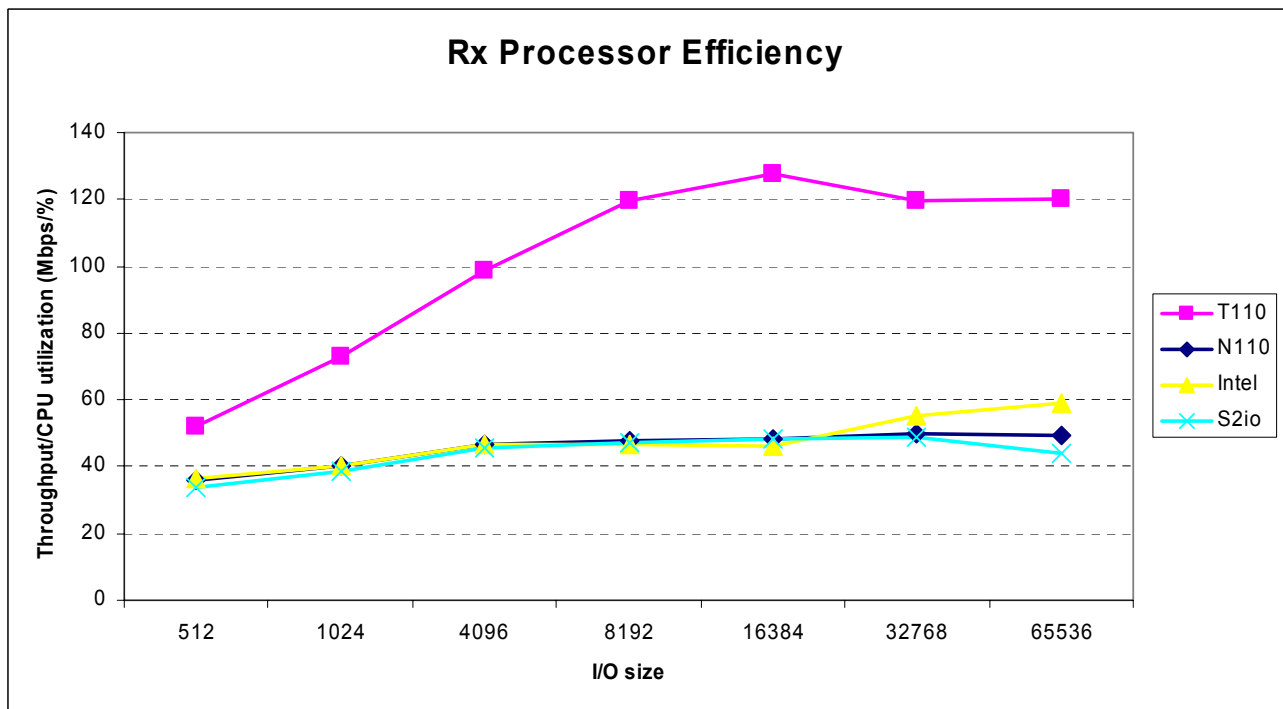


Figure 6b: Rx CPU efficiency versus I/O size

As shown in Figures 6a and 6b, the Chelsio Communications T110 10Gbit Protocol Engine had the highest receive goodput and the best CPU efficiency index across all I/O sizes from 512 bytes to 64KB for these tests with 1,500 byte frames. The T110's CPU efficiency was on average 2.5 times that of the other adapters. Among the non-offload adapters, Chelsio's N110 server adapter had the best performance.

Latency Measurements

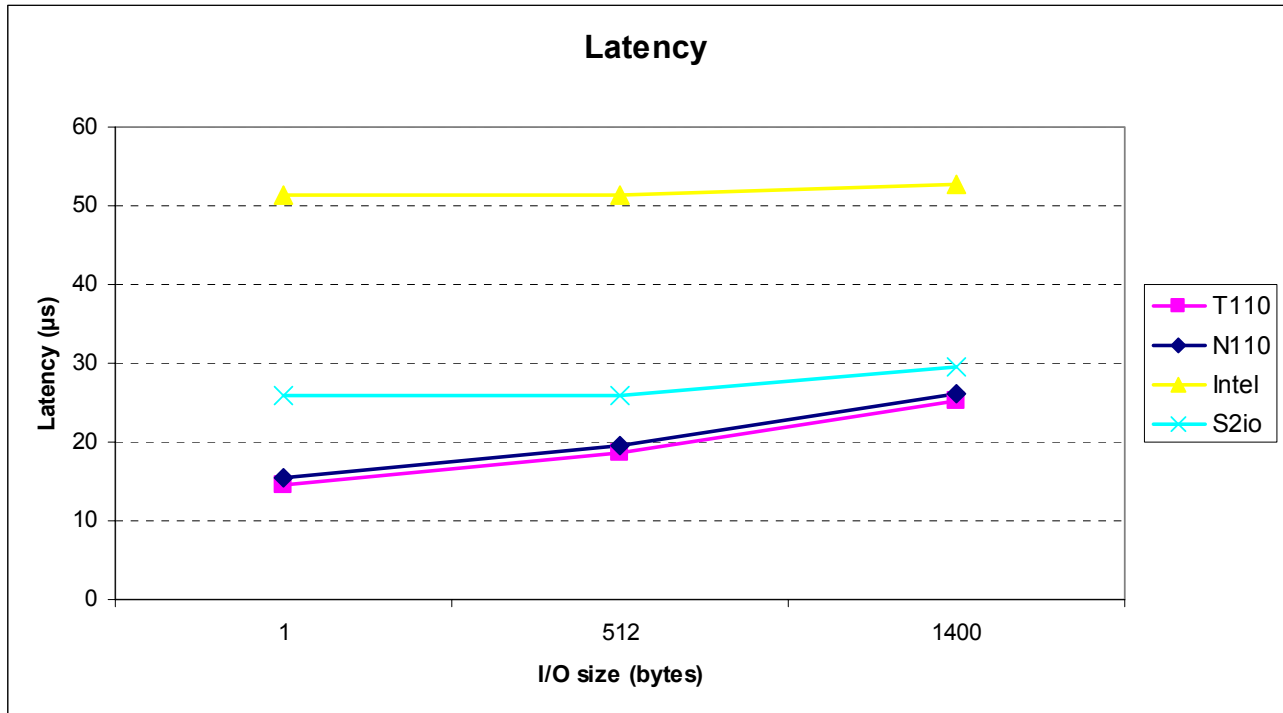


Figure 7: Latency versus I/O size (lower latency is better)

For measuring latency, all products were installed and tested in their default configuration. As shown in Figure 7, the T110 had the shortest latency at 14.6µs with 1 byte I/O size, and is the best performing adapter across the studied range. The next best adapter was the N110, with a 1 byte latency of 15.5µs and consistently the second best performance across the range. The S2io adapter had a 1 byte latency of 25.8µs and the Intel adapter ranked last at 51.3µs.

Testing methodology

Chelsio commissioned VeriTest to verify a performance study of the following four 10GbE network adapters:

- Chelsio Communications T110-SR 10Gbit Protocol Engine
- Chelsio Communications N110-SR 10Gbit Server Adapter
- Intel PRO/10GbE SR Server Adapter
- S2io Technologies Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter

To conduct this performance test audit, VeriTest sent a Test Engineer to the Chelsio facility in Sunnyvale, California. The VeriTest Test Engineer verified the testing environment, tool, product and server configurations, and validated the performance results.

The network benchmarking tool Netperf v2.2pl4 was used to measure latency and Iperf 1.7.0 was used to measure the TCP throughput (goodput) and CPU utilization, while changing network parameters, such as the number of network connections, the I/O size, and frame size. Additional information on Netperf is available at <http://www.netperf.org/netperf/NetperfPage.html>. Additional information on Iperf is available at <http://dast.nlanr.net/projects/Iperf>.

Both tools have their strengths and weaknesses and are therefore needed to obtain the performance measures of interest here. Netperf provides the transaction test feature required for latency measurements but scales poorly as the number of connections is increased because it spawns a process for each connection. Iperf scales better because it spawns somewhat lighter weight threads instead but does not provide the transaction test feature.

The following equipment was used for this performance study:

- One Chelsio T110-SR 10Gbit Protocol Engine
- One Chelsio N110-SR 10Gbit Server Adapter
- One Intel PRO/10GbE SR Server Adapter
- One S2io Technologies Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter
- Two identical single processor systems, configured with one AMD Opteron 248 2.2GHz, 1GB PC3200 RAM, one 80GB IDE Hard Drive, and one 18GB Ultra-SCSI Hard Drive.
- One Foundry Networks Netlron 40G switch

Test bed Configuration

For the test servers, Chelsio provided two identical single processor AMD Opteron 248 2.2GHz server systems configured with 1GB of PC3200 RAM, an 80GB IDE hard disk drive running RedHat 9.0 (2.6.6). We used one system to house each of the network adapters as testing was performed, and equipped the other with a T110 Protocol Engine to serve as a testing peer. The two servers were connected through the Foundry Networks Netlron 40G switch.

Prior to the start of testing, VeriTest verified that the Linux kernel 2.6.6 was directly downloaded from <http://www.kernel.org>. The driver source code was copied directly into specific build folders within the kernel source tree. Every attempt was made to ensure that the testing be done in as fair manner as possible, including the use of the latest driver versions available at the time of this performance study. Namely, the Intel v1.0.65 (4/30/04), S2io v1.154 (7/15/04), and Chelsio v1.23 (7/23/04) drivers were used respectively. During the kernel build process, Chelsio enabled support for the following network hardware:

- Intel PRO/10GbE support
- S2io 10Gbe XFrame NIC
- Chelsio Communication T1 support

The kernel was compiled for a 64-bit system and copied into to the target host server.

Performance Tuning

Since we were auditing and verifying a competitive analysis, we placed special emphasis on ensuring that all of the products under test were tuned as recommended by their respective vendor. The following section describes the performance tuning tasks that we verified were completed prior to testing each product.

For all cards, the following tuning modifications were performed:

```
### IPV4 specific settings
net.ipv4.tcp_timestamps = 0 # turns TCP timestamp support off, default 1,
reduces CPU use
net.ipv4.tcp_sack = 0 # turn SACK support off, default on
# on systems with a VERY fast bus -> memory interface this is the big gainer
net.ipv4.tcp_rmem = 10000000 10000000 10000000 # sets min/default/max TCP read
buffer, default 4096 87380 174760
net.ipv4.tcp_wmem = 10000000 10000000 10000000 # sets min/pressure/max TCP write
buffer, default 4096 16384 131072
net.ipv4.tcp_mem = 10000000 10000000 10000000 # sets min/pressure/max TCP buffer
space, default 31744 32256 32768

### CORE settings (mostly for socket and UDP effect)
net.core.rmem_max = 524287 # maximum receive socket buffer size, default 131071
net.core.wmem_max = 524287 # maximum send socket buffer size, default 131071
net.core.rmem_default = 524287 # default receive socket buffer size, default
65535
net.core.wmem_default = 524287 # default send socket buffer size, default 65535
net.core.optmem_max = 524287 # maximum amount of option memory buffers, default
10240
net.core.netdev_max_backlog = 300000 # number of unprocessed input packets
before kernel starts dropping them, default 300
- END sysctl_ixgb.conf
```

The maximum performance settings were used as recommended by each vendor. For the T110 and N110 products, the configuration settings are described in Appendix B. For the Intel product, the recommended configuration settings were followed as described in Appendix C. For the S2io product, the recommended configuration settings were followed as described in Appendix D.

Test Setup

The four server-to-server test configurations used were as follows:

1. Server 1 (102.50.50.66): Chelsio T110-SR
Server 2 (102.50.50.67): Chelsio T110-SR
2. Server 1 (102.50.50.66): Chelsio T110-SR
Server 2 (102.50.50.67): Chelsio N110-SR
3. Server 1 (102.50.50.66): Chelsio T110-SR
Server 2 (102.50.50.67): Intel PRO/10GbE SR Server Adapter
4. Server 1 (102.50.50.66): Chelsio T110-SR
Server 2 (102.50.50.67): S2io Technologies Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter

Given the full-duplex line rate capabilities of the T110, using it in the peer server ensures that the system under test is not hindered by the performance of the peer. Using two identical network adapter products in the two servers would prevent independent testing of each direction since the performance figures for each

adapter would be limited by its slower direction. This also means that the latency figures reported here benefit from the fact that there is only one pass through the slower adapters.

The testbed configuration used is shown in Figure 10.

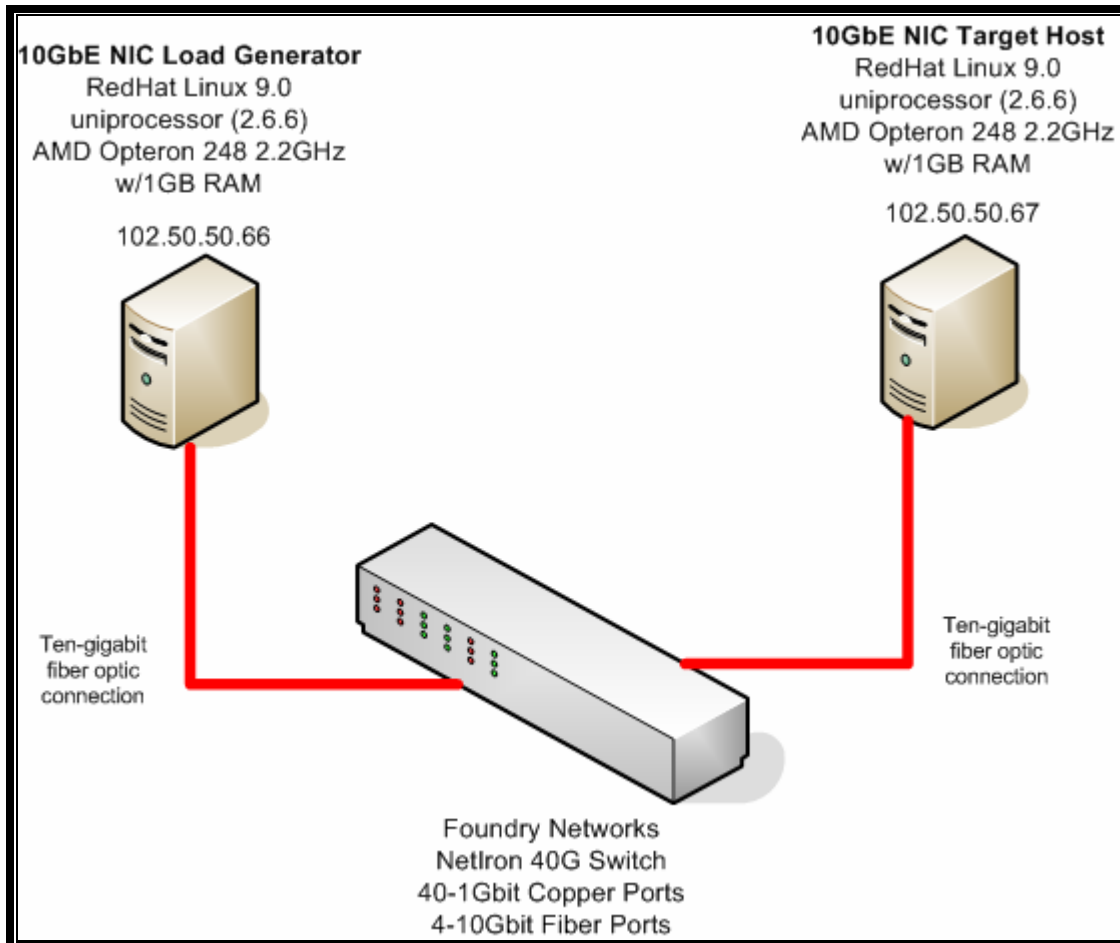


Figure 8: Testbed configuration

Testing procedure

Chelsio created a shell script to automate the testing. This script initiated each test run and returned results to a .txt file which was imported into Excel. Appendix F contains more information regarding this process.

The following process was used to drive the automated scripts from both tools to perform the following operations:

1. Install the network adapter to be tested.
2. Reboot the two participating computer systems.
3. Verify the systems are available.
4. Tune the network adapter as recommended by the respective vendor.
5. Start the Iperf server on the system that will be receiving data.
6. Run the appropriate test script to perform test. (Data is redirected to a text file for later importing into Excel for data analysis.)

Prior to testing, we verified that the results generated by the automated test scripts matched the results generated when we manually executed the tests. Additionally, during the test, we spot checked the displayed

results from the console windows. At the completion of each test that we spot checked, we compared the two values.

We monitored the target host server processor utilization statistics using TOP and matched those with the results reported by the benchmarking tool at the end of the tests. We executed two runs for each test configuration and ensured that results repeatability was within 5% from run-to-run.

We reported throughput in terms of megabytes per second, CPU utilization as a percentage, and the latency in microseconds. The latency figure was obtained by running a Netperf TCP Request Response (TCP_RR) transaction test on a single connection, and the one-way latency was calculated by using the following formula:

$$\text{Latency} = (1 / \text{number of transactions per second}) / 2$$

The throughput and CPU utilization numbers were used to compute a **Processor (CPU) Efficiency** measure, defined as the ratio of the throughput divided by the CPU utilization. The resulting index, in units of *Mbps per %CPU*, allows a normalized CPU performance measure to be obtained for all adapters regardless of the throughput achieved by each. The combination of the throughput achieved and the processor efficiency gives well rounded picture of the overall performance of each adapter.

Test Results

The section provides the complete results for all the conducted tests. Please refer to the Test Methodology section for the full details on how these tests were conducted.

Chelsio commissioned VeriTest to audit a performance study of the following four 10GbE network adapters:

- Chelsio Communications T110-SR 10Gbit Protocol Engine
- Chelsio Communications N110-SR 10Gbit Server Adapter
- Intel PRO/10GbE SR Server Adapter
- S2io Technologies Xframe 10 Gigabit Ethernet PCI-X Server/Storage Adapter

For each network adapter the following tests were ran to collect throughput, latency, and CPU utilization data:

Throughput and CPU utilization tests – Varying the number of connections

- Server in transmit mode using 1 connection.
- Server in transmit mode using 5 connections.
- Server in transmit mode using 10 connections.
- Server in transmit mode using 100 connections.
- Server in transmit mode using 1000 connections.
- Server in receive mode using 1 connection.
- Server in receive mode using 5 connections.
- Server in receive mode using 10 connections.
- Server in receive mode using 100 connections.
- Server in receive mode using 1000 connections.

The tests above were performed for 1500 byte and 9000 byte Ethernet frames.

Throughput and CPU utilization tests – Varying the I/O size

- Server in transmit mode using 1 connection and I/O size of 512 bytes.
- Server in transmit mode using 1 connection and I/O size of 1024 bytes.
- Server in transmit mode using 1 connection and I/O size of 4096 bytes.
- Server in transmit mode using 1 connection and I/O size of 8KB
- Server in transmit mode using 1 connection and I/O size of 16KB
- Server in transmit mode using 1 connection and I/O size of 32KB
- Server in transmit mode using 1 connection and I/O size of 64KB
- Server in receive mode using 1 connection and I/O size of 512 bytes.
- Server in receive mode using 1 connection and I/O size of 1024 bytes.
- Server in receive mode using 1 connection and I/O size of 4096 bytes.
- Server in receive mode using 1 connection and I/O size of 8KB
- Server in receive mode using 1 connection and I/O size of 16KB
- Server in receive mode using 1 connection and I/O size of 32KB
- Server in receive mode using 1 connection and I/O size of 64KB

The tests above were performed for 1500 byte frames.

Latency tests

- Server using 1 connection with a 1 byte I/O size.
- Server using 1 connection with a 512 byte I/O size.
- Server using 1 connection with a 1400 byte I/O size.

All tests were run with the following test parameters, except where noted above:

Block size = 256KB

I/O Size = 64KB

Test length = 120 seconds

Chelsio T110 network adapter configuration = Offload mode enabled

The test setup was:

Host target server (102.50.50.67)

Load generating server (102.50.50.66) : contained Chelsio T110-SR 10Gbit Protocol Engine for all tests

Transmit Test Results

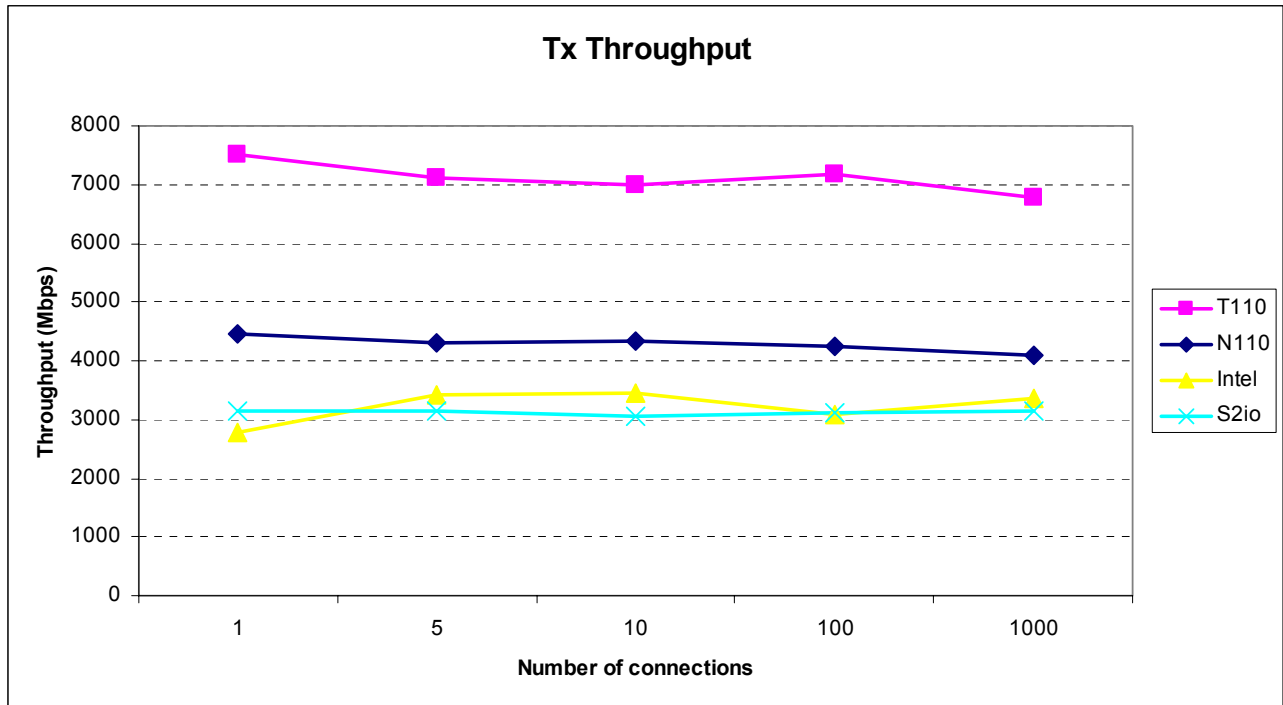


Figure 9a: Tx throughput versus number of connections (1,500 byte frames)

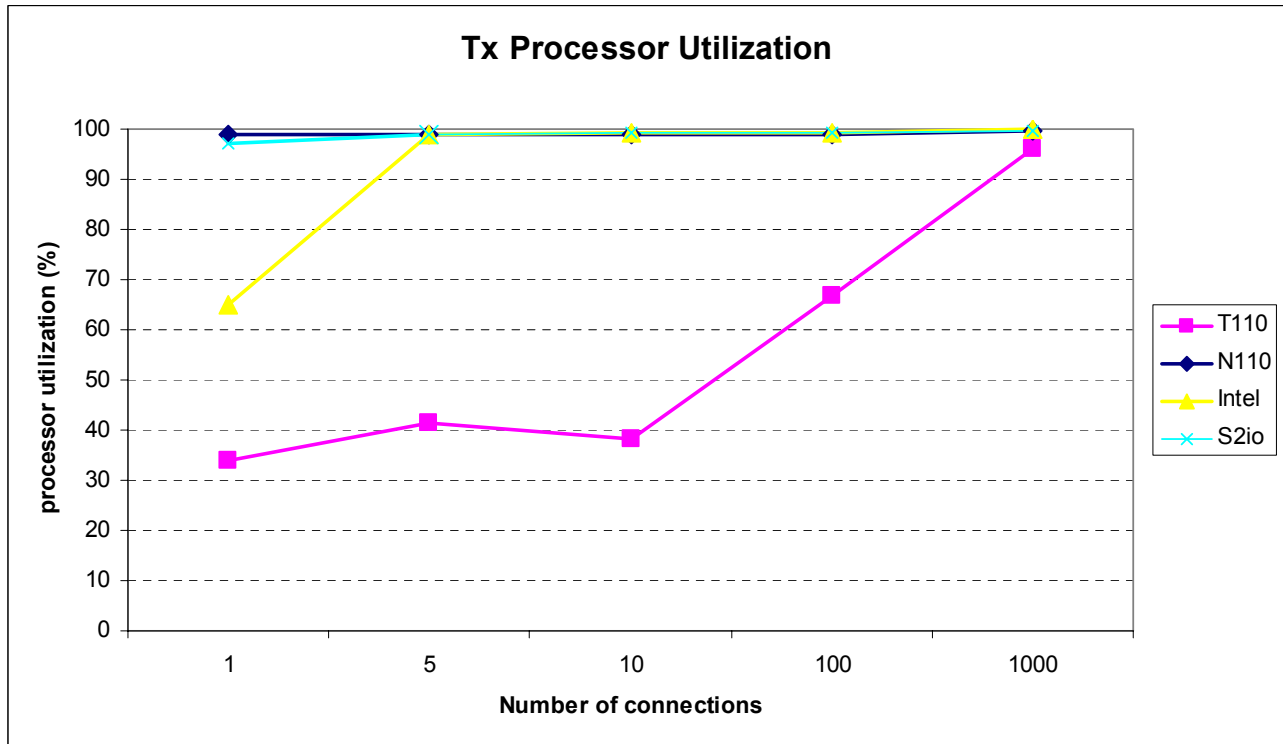


Figure 9b: Tx processor utilization versus number of connections (1,500 byte frames)

Figure 9b shows that, in contrast to the other adapters which fully consume the CPU as early as 5 connections, Chelsio's T110 CPU utilization slowly increases as the number of connections increases. The

increase in CPU utilization for the T110 at large number of connections is attributed to the software overhead in managing the sending threads, and cache contention effects. The T110's CPU utilization curve would have stayed flat if the spawning of that many threads was replaced by a more efficient connection handling.

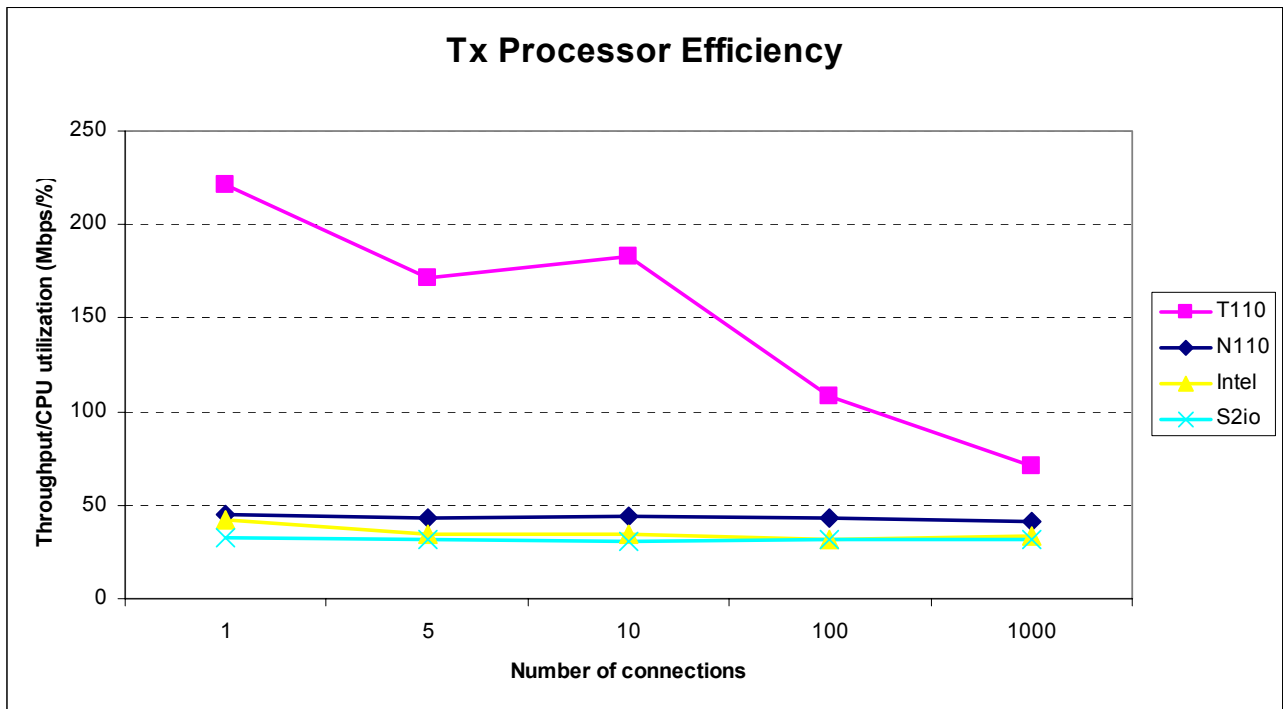


Figure 9c: Tx processor efficiency versus number of connections (1,500 byte frames)

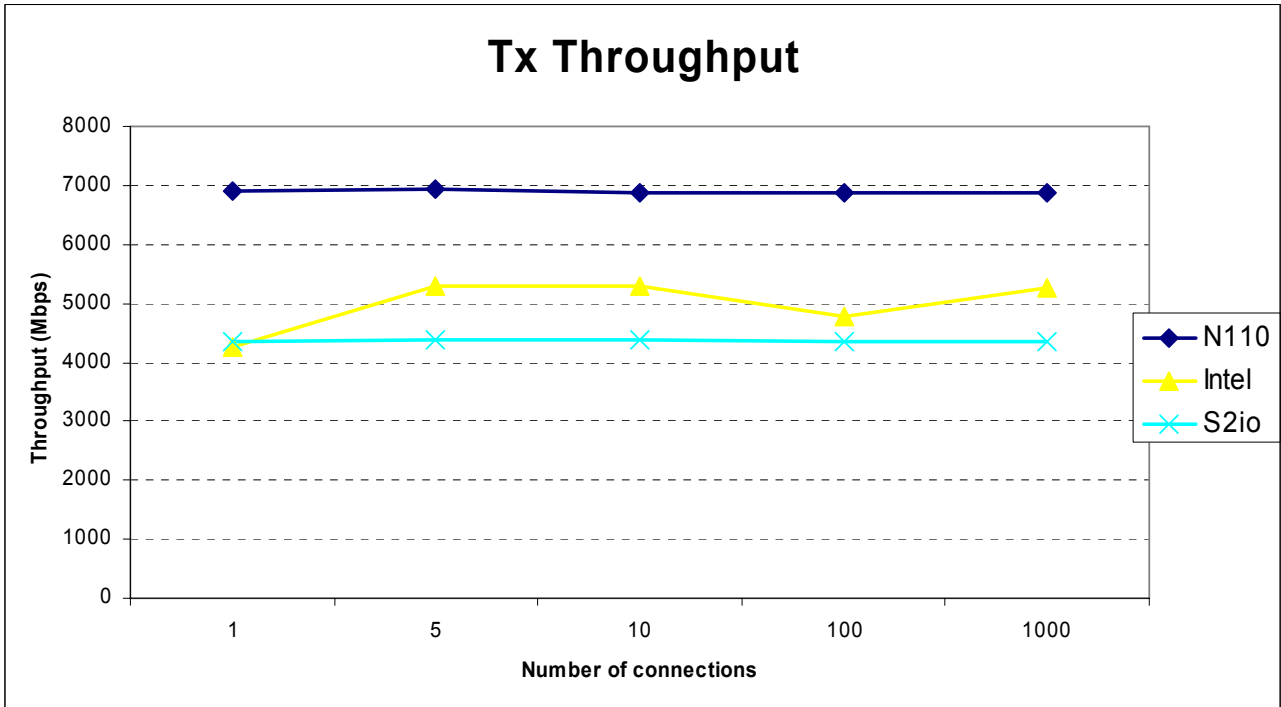


Figure 10a: Tx throughput versus number of connections (9000 byte frames)

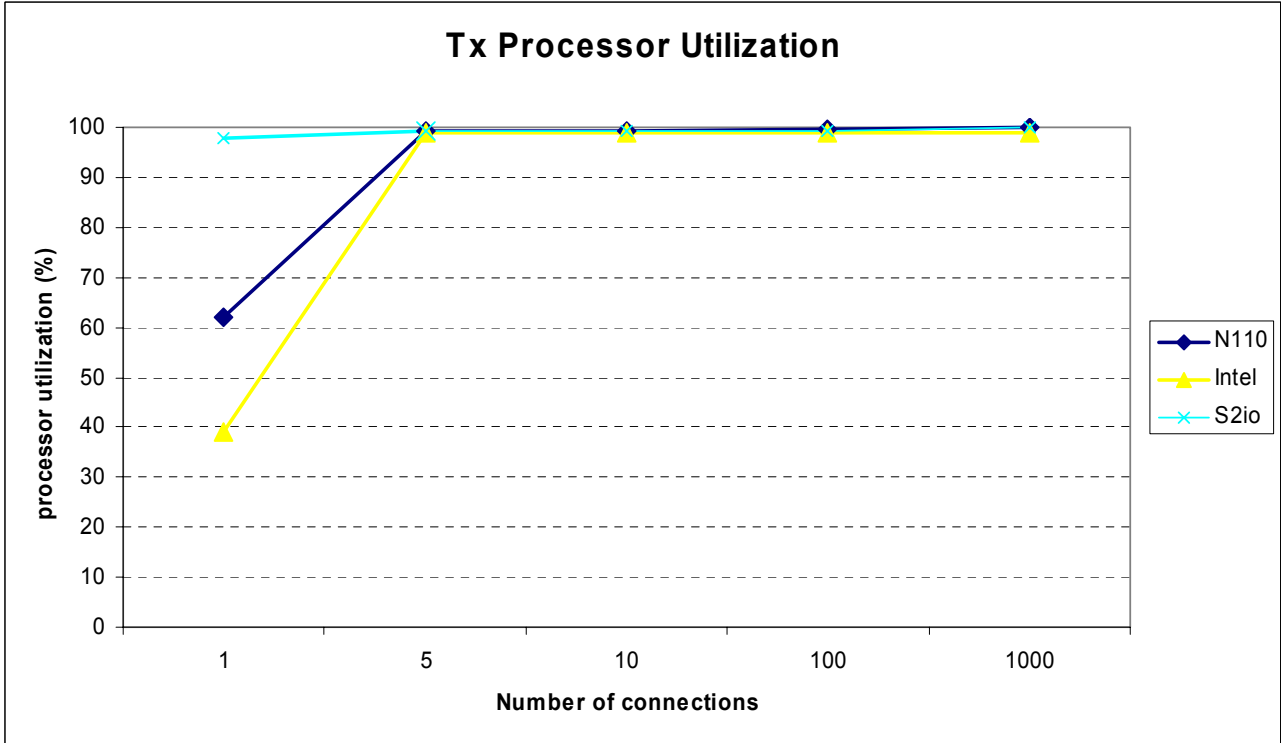


Figure 10b: Tx processor utilizations versus number of connections (9000 byte frames)

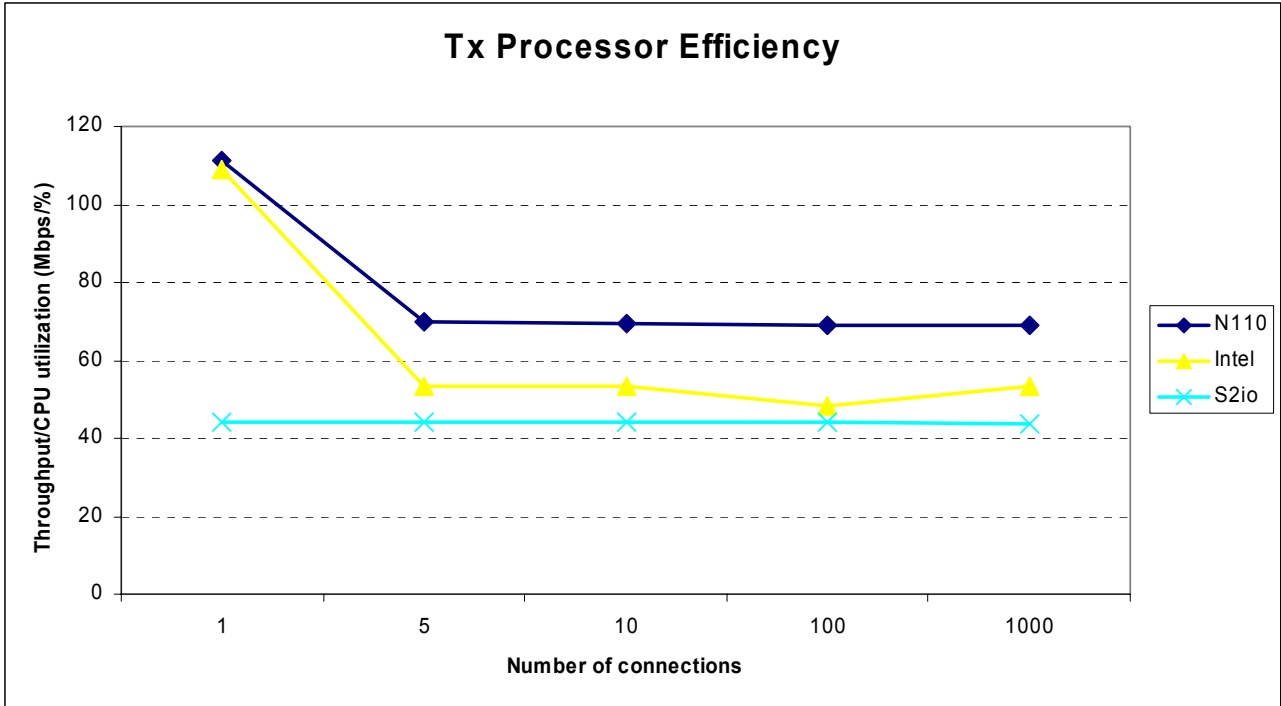


Figure 10c: Tx processor efficiency versus number of connections (9000 byte frames)

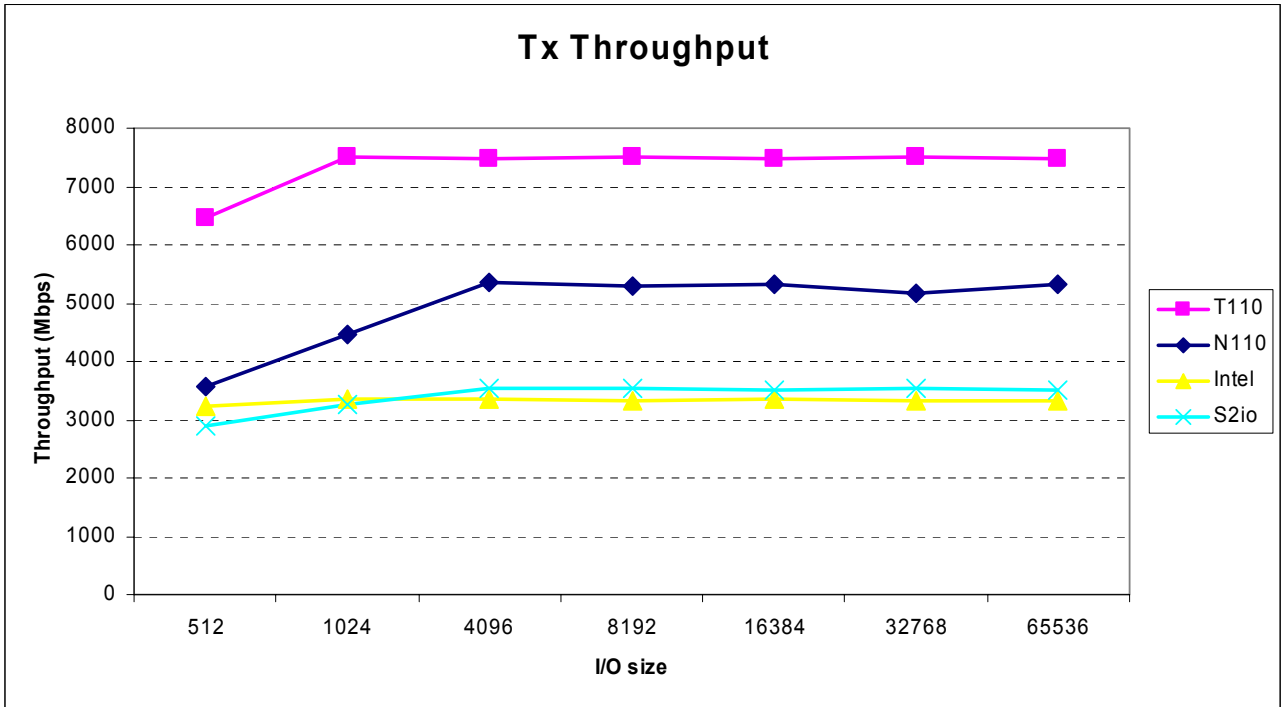


Figure 11a: Tx throughput versus I/O size (1,500 byte frames)

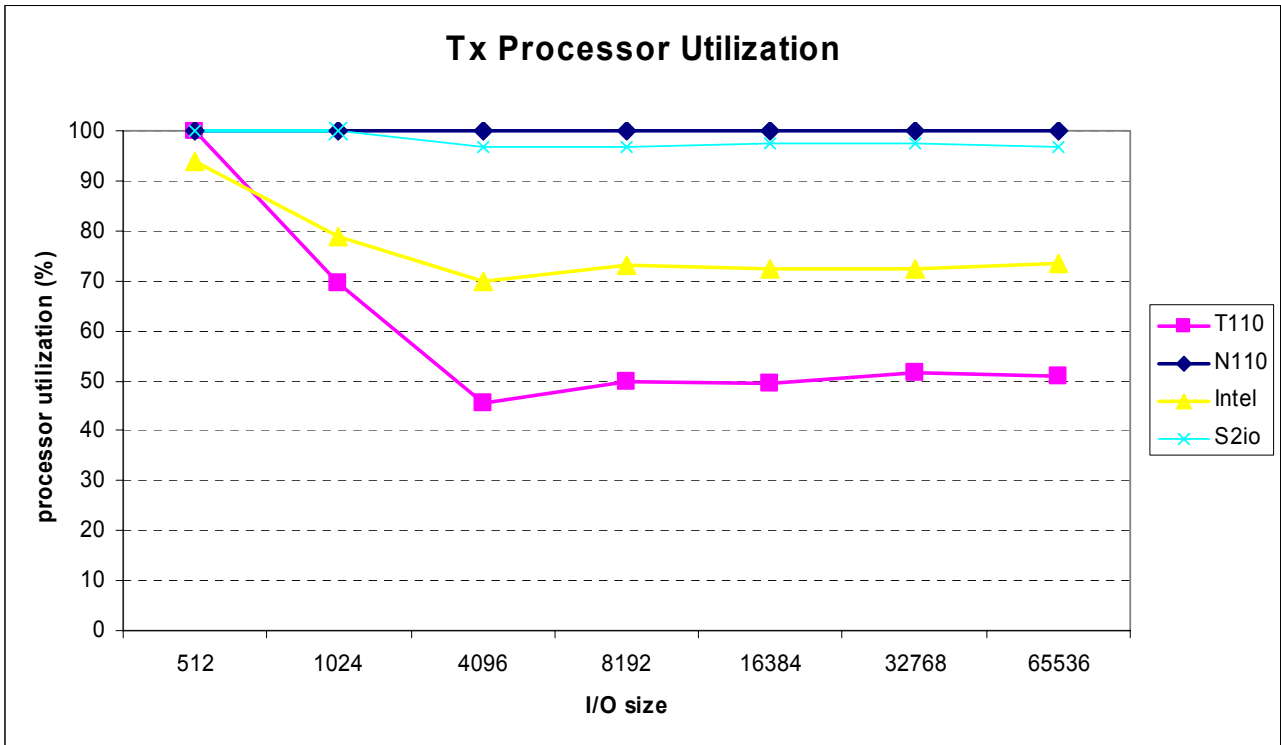


Figure 11b: Tx processor utilization versus I/O size (1,500 byte frames)

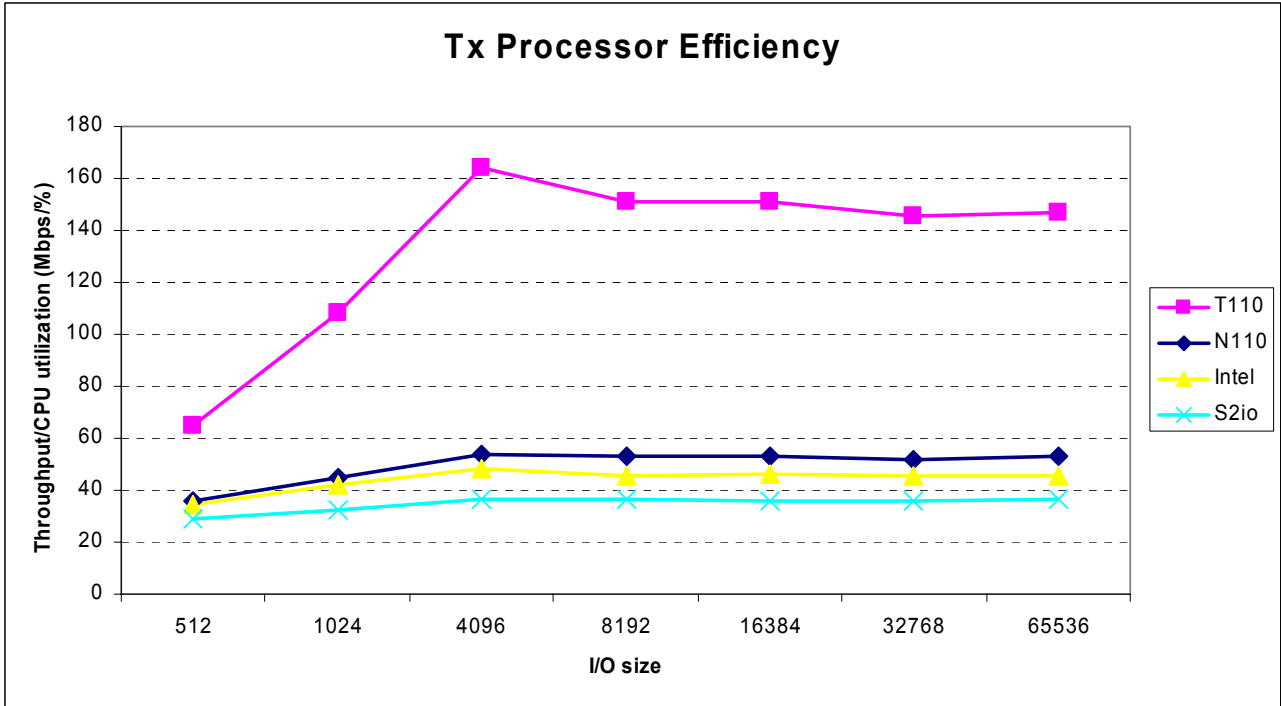


Figure 11c: Tx processor efficiency versus I/O size (1,500 byte frames)

Receive Test Results

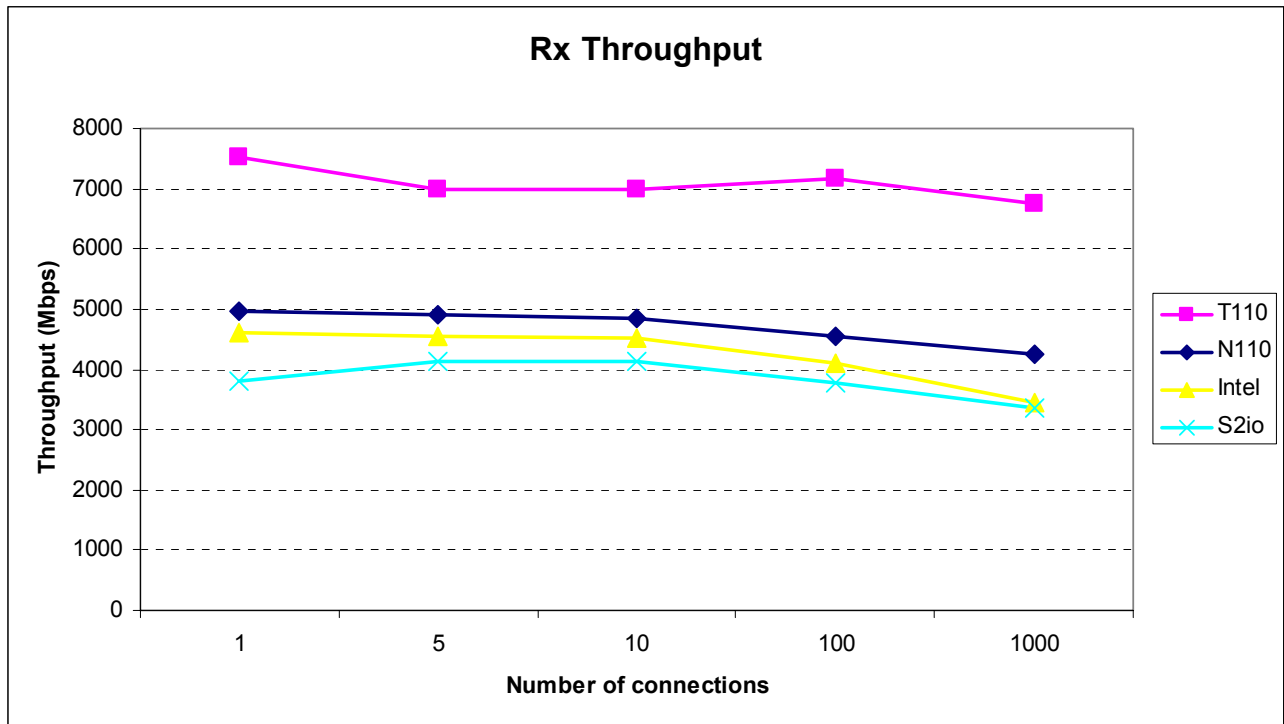


Figure 12a: Rx throughput versus number of connections (1,500 byte frames)

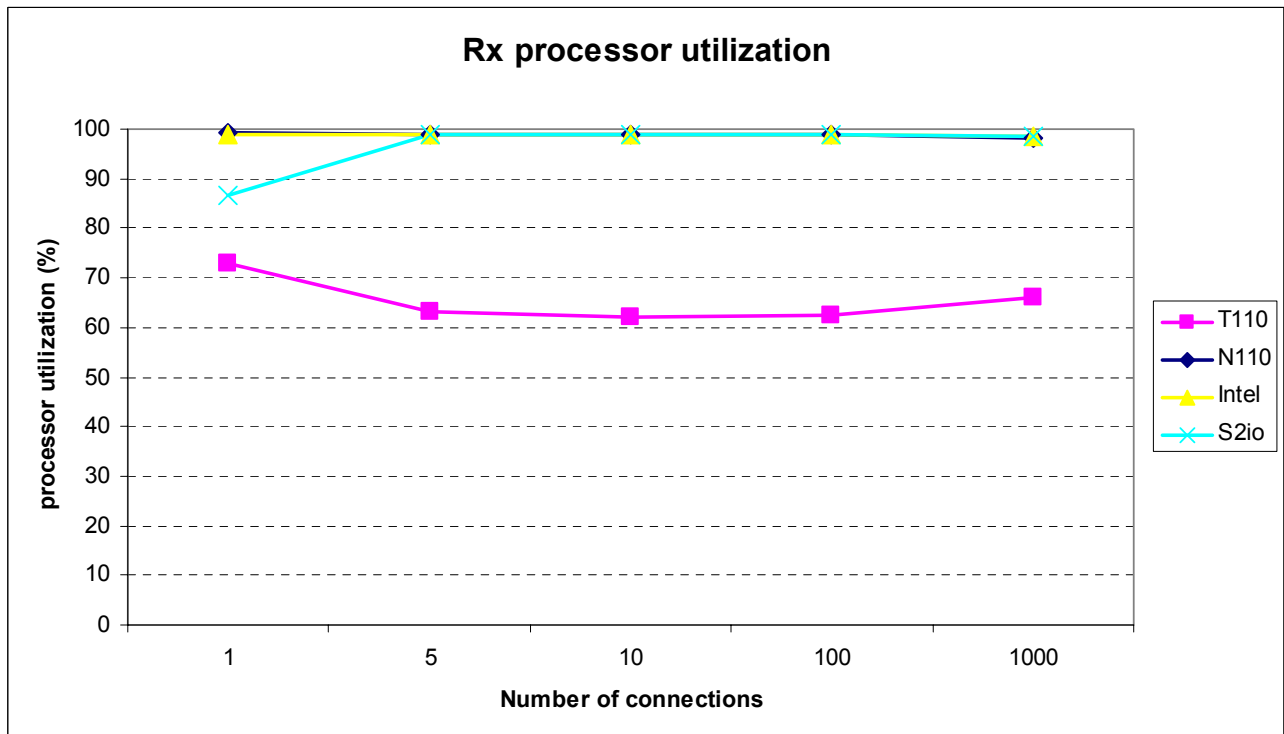


Figure 12b: Rx processor utilization versus number of connections (1,500 byte frames)

Figure 12b shows that, in contrast to the other adapters which fully consume the CPU as early as 5 connections, Chelsio's T110 CPU utilization remains flat at about 65% as the number of connections increases while maintaining a goodput figure of about 7Gbps.

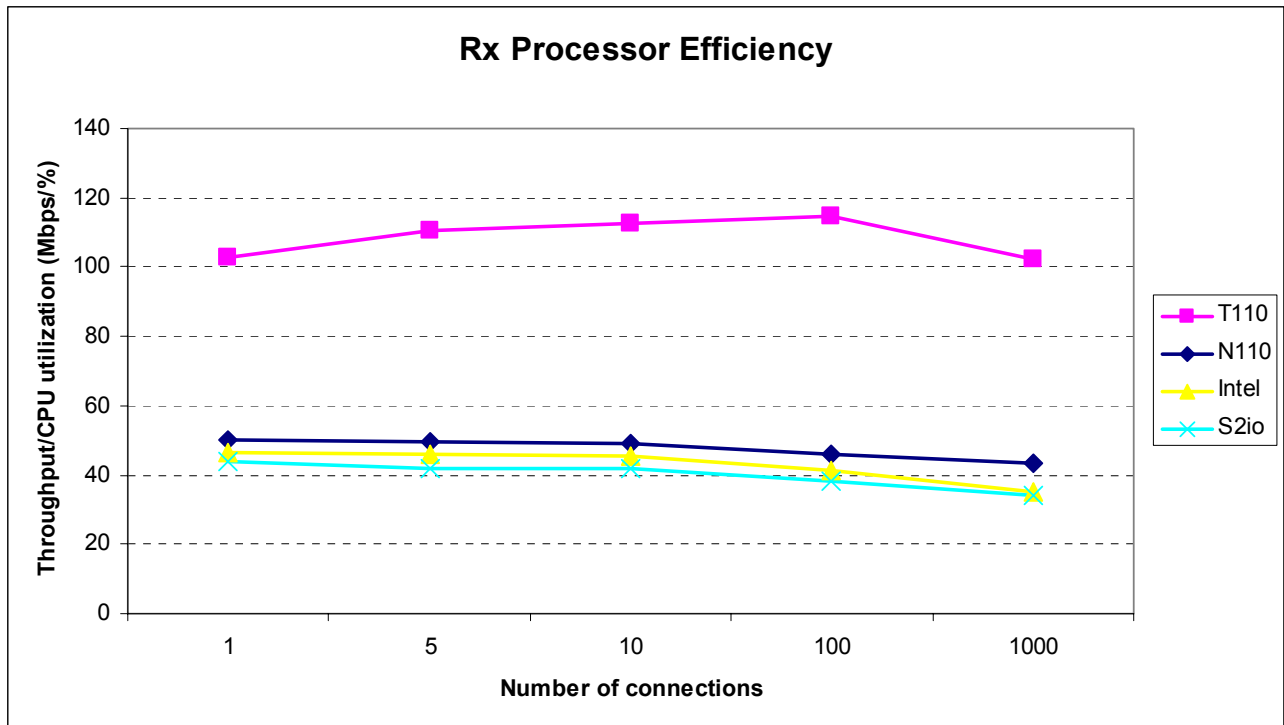


Figure 12c: Rx processor efficiency versus number of connections (1,500 byte frames)

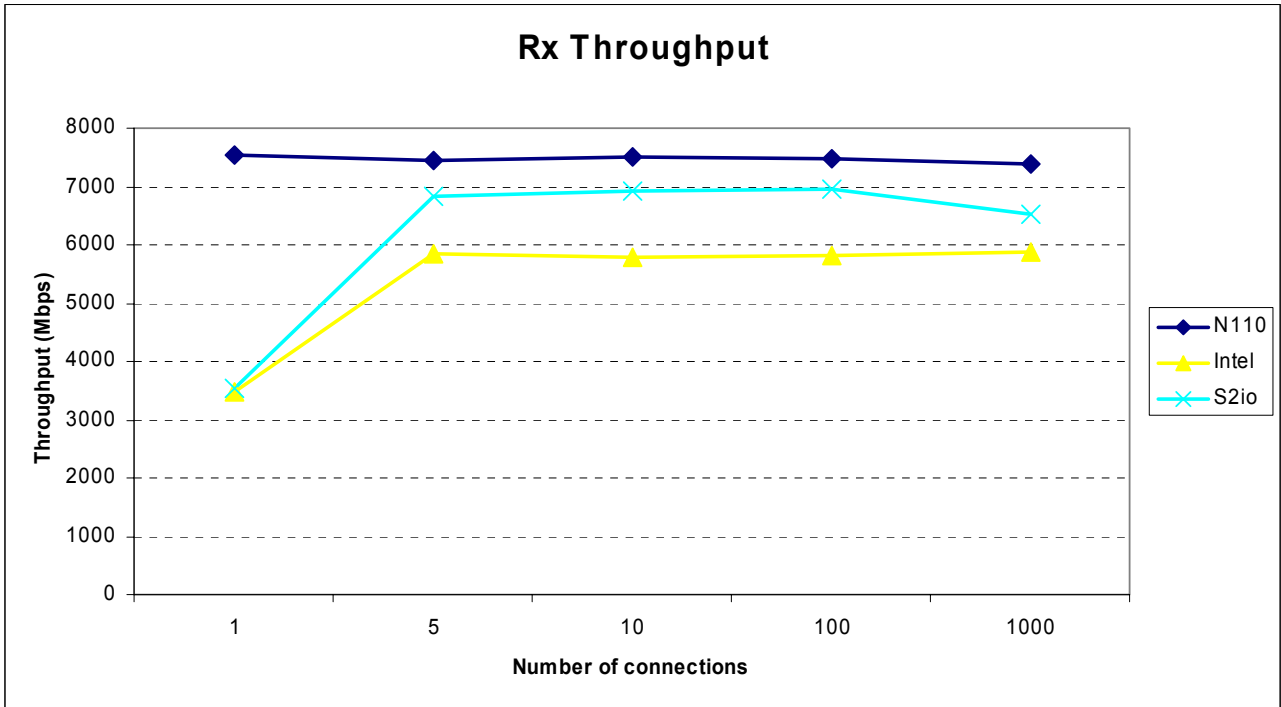


Figure 13a: Rx throughput versus number of connections (9000 byte frames)

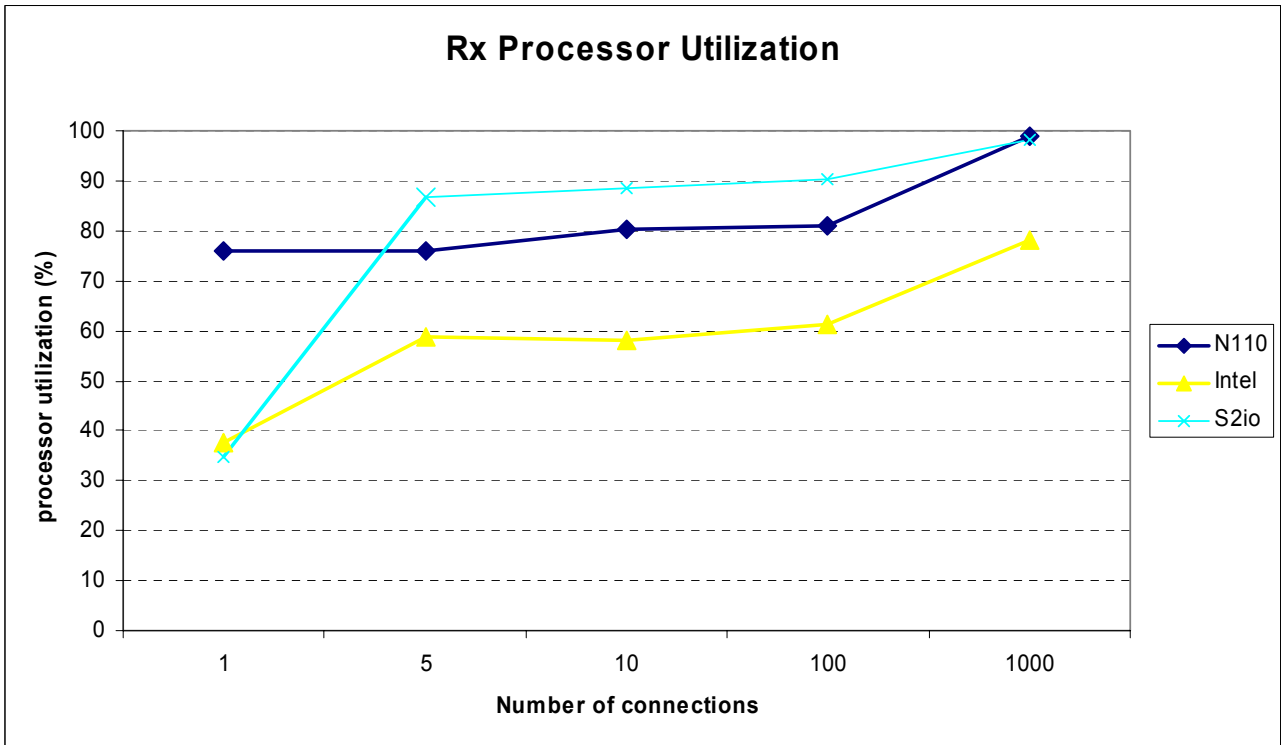


Figure 13b: Rx processor utilizations versus number of connections (9000 byte frames)

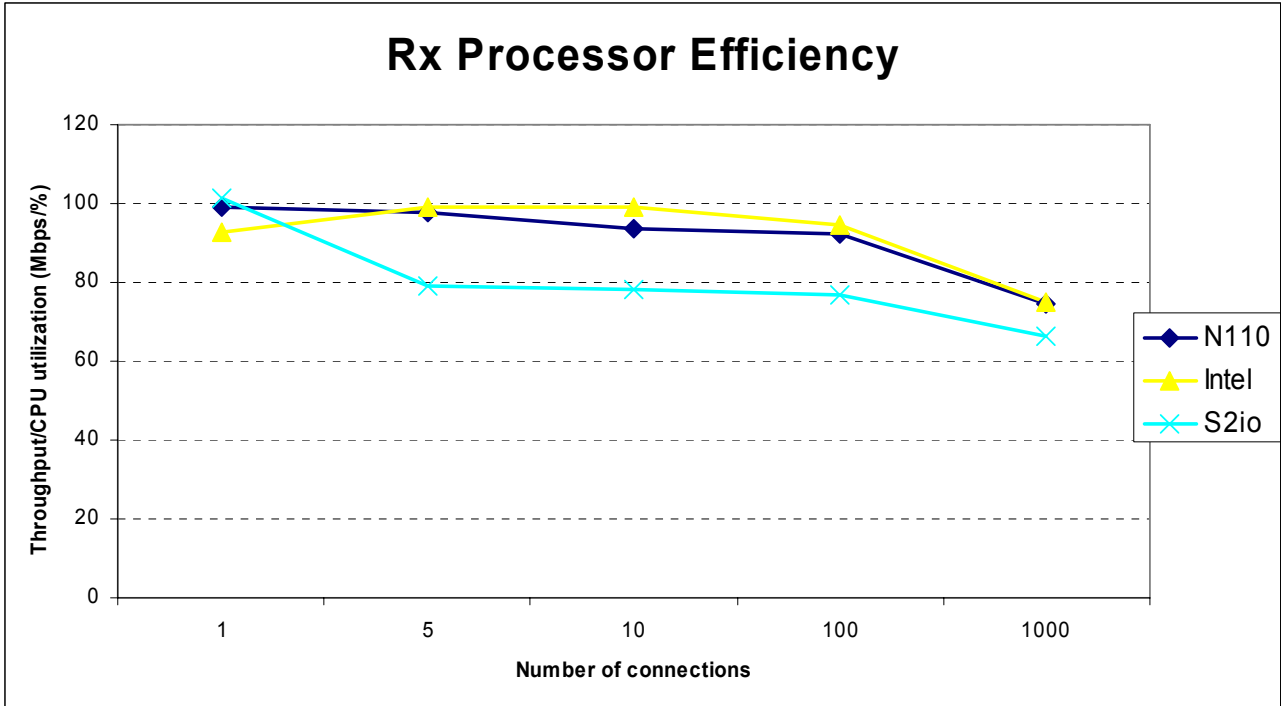


Figure 13c: Rx processor efficiency versus number of connections (9000 byte frames)

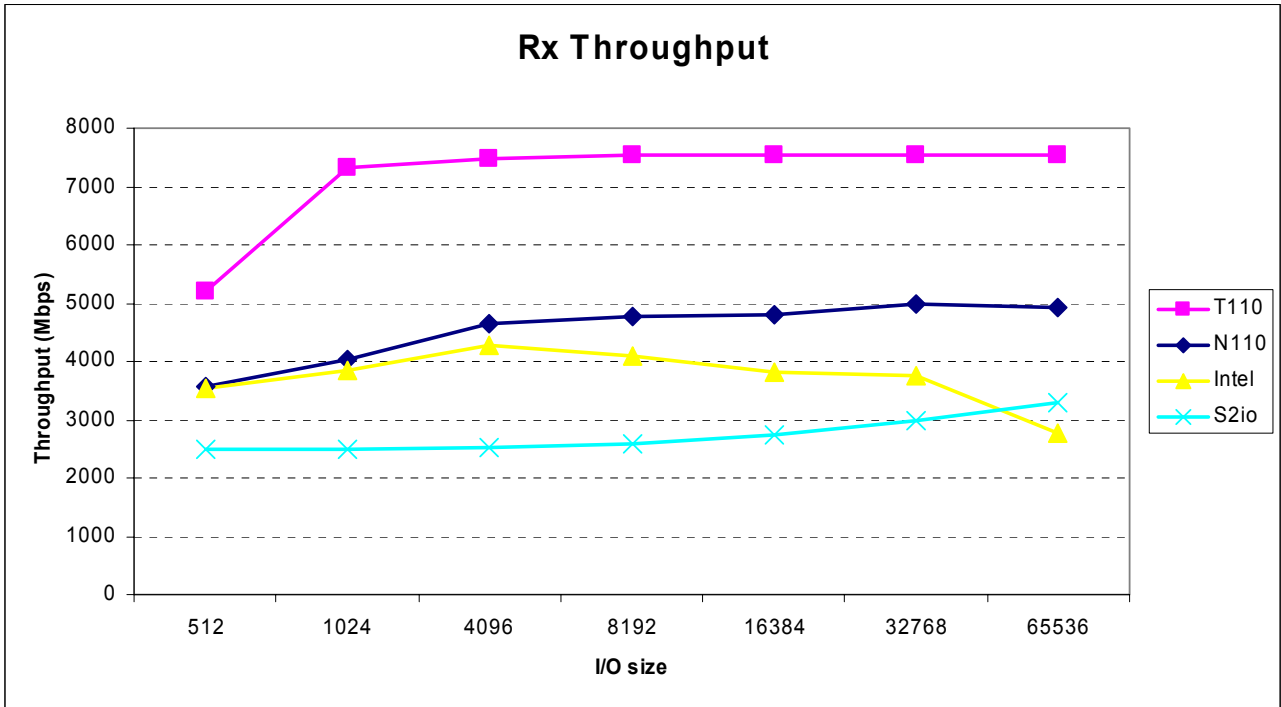


Figure 14a: Rx throughput versus I/O size (1,500 byte frames)

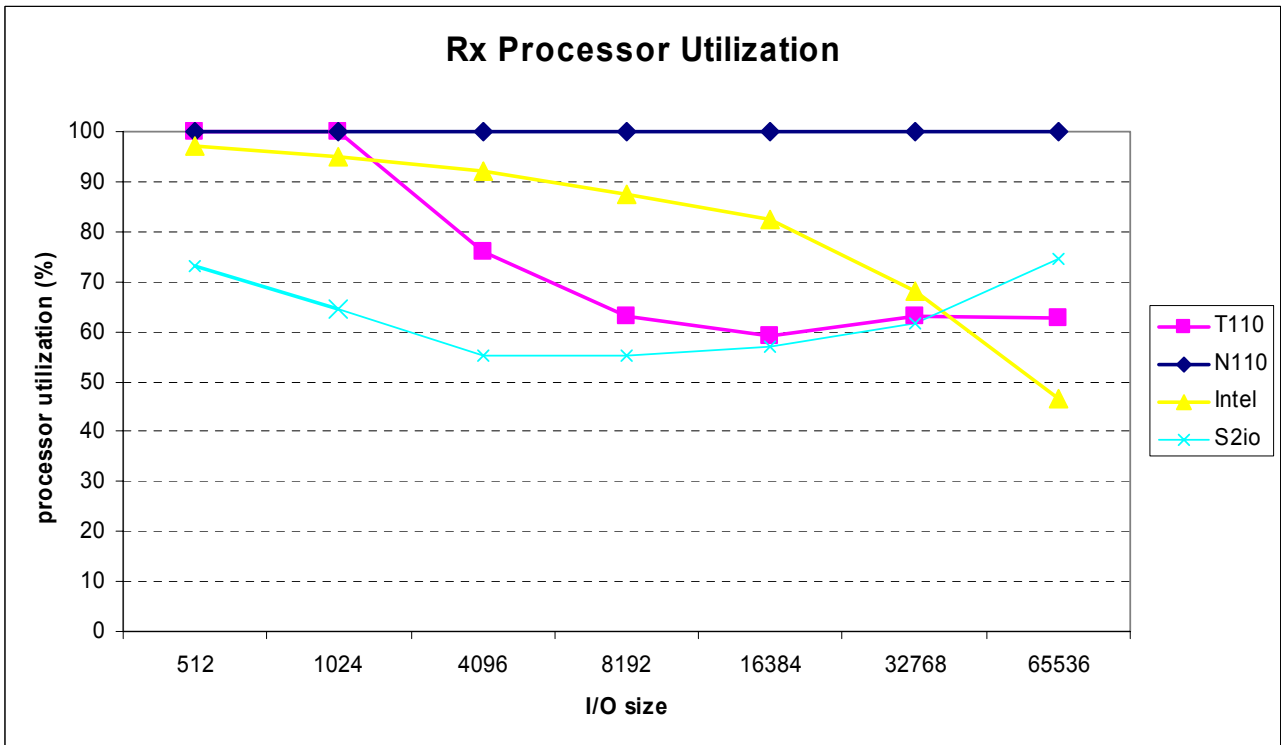


Figure 14b: Rx processor utilization versus I/O size (1,500 byte frames)

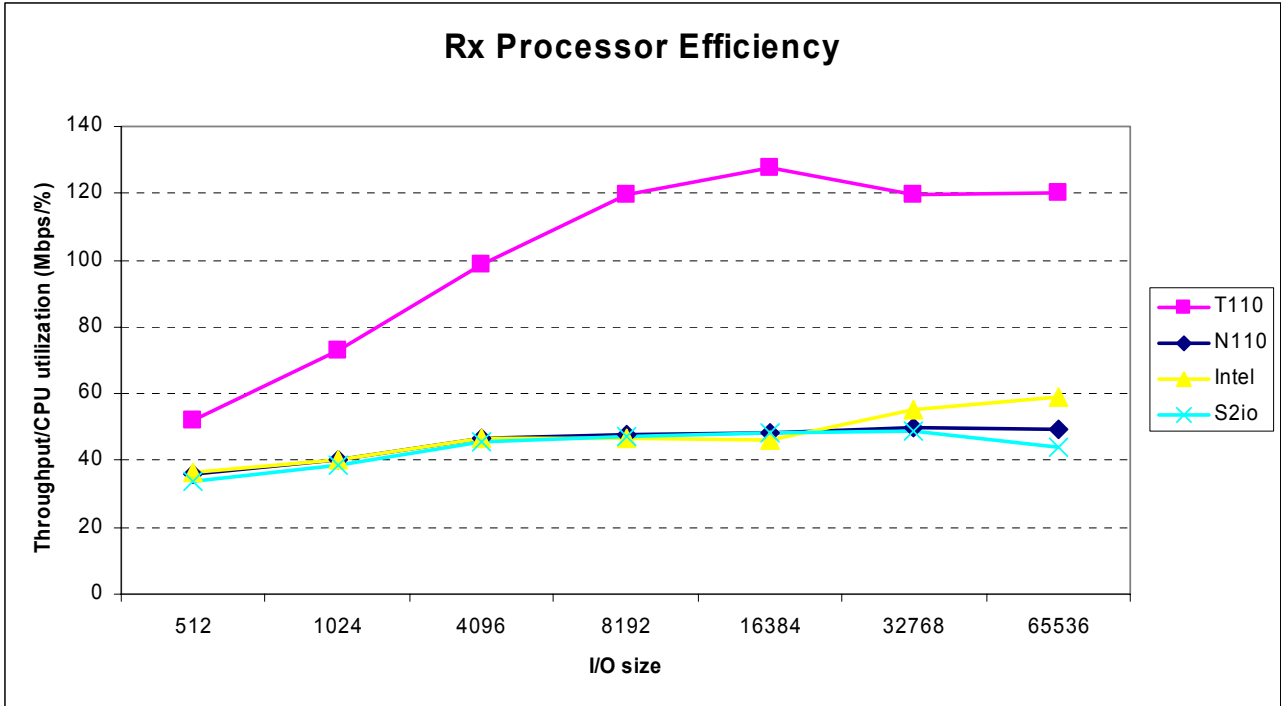


Figure 14c: Rx processor efficiency versus I/O size (1,500 byte frames)

Latency Test Results

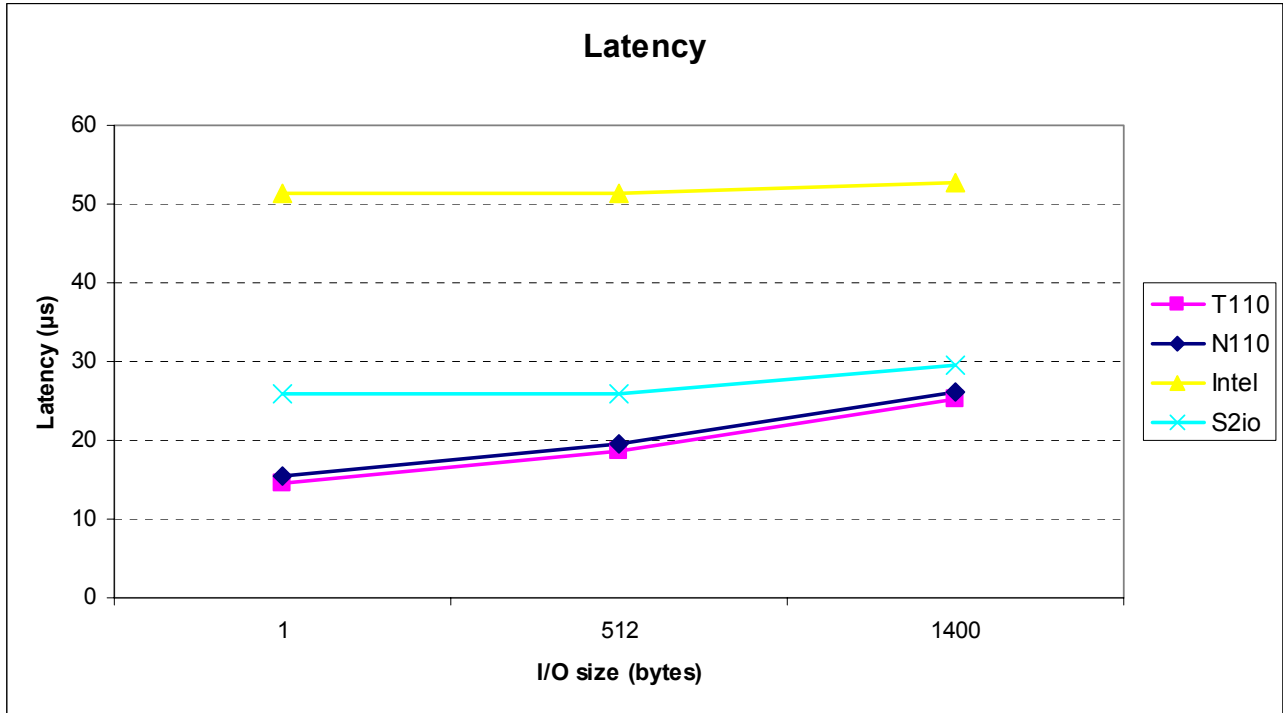


Figure 15: Latency versus I/O size (lower latency is better)

Appendix A. Systems Disclosure Information

Server Systems	
CPU Type	Single AMD Opteron 248 2.2GHz
Motherboard Name	RioWorks HDAMA Version 1.82
Motherboard Chipset	AMD-8000
System Memory	1024 MB (PC3200)
BIOS Type	Phoenix BIOS 3 Release 6.0
Video Adapter	ATI 3D-Rage XL
Floppy Drive	1440 KB
Hard Disk Capacity (BIOS)	80GB IDE, Western Digital WD800JB-00ETA0 (OS kernel)
Hard Disk Capacity (BIOS)	18.2GB Ultra-SCSI, Seagate ST318453LW
Network Adapter	2 Broadcom BCM5702 Dual Port Server Adapter
Network Adapter	Chelsio T110 10Gbit Network Adapter
Network Adapter Driver	Chelsio v1.0
Operating System	RedHat 9.0 Uniprocessor (kernel v.2.6.6)

Figure A1: Host and Peer Servers

Foundry Networks NetIron 40G Switch	
Foundry Operating System	IronWare 1.1.2
Module Processor	400MHz PowerPC 440GP
Line Card 2	4 – 10G ports
Line Card 3	40 – 1G ports
Jumbo Capable	Yes

Figure A2: Foundry Network NetIron 40G Switch Configuration

Appendix B. Chelsio Driver Version, installation, and tuning parameters

DRIVER VERSION: Chelsio v1.0


DRIVER INFO AND CONFIGURATION:

Following boot up and after the driver is loaded, but before the test, run the script “pet110init.tcl”, which sets the Chelsio Terminator ASIC core clock to the appropriate value for 10Gbit operation.

Appendix C. Intel Driver Version, Installation, and Tuning Parameters


Improving Performance

With the Intel PRO/10 GbE adapter, the default Linux configuration will very likely limit the total available throughput artificially. There is a set of things that when applied together increase the ability of Linux to transmit and receive data. The following enhancements were originally acquired from settings published at <http://www.spec.org/web99/> for various submitted results using Linux.

 **NOTE:** These changes are only suggestions, and serve as a starting point for tuning your network performance.

The changes are made in three major ways, listed in order of greatest effect:

- Use `ifconfig` to modify the `mtu` (maximum transmission unit) and the `txqueuelen` parameter.
- Use `sysctl` to modify `/proc` parameters (essentially kernel tuning)
- Use `setpci` to modify the MMRBC field in PCI-X configuration space to increase transmit burst lengths on the bus.

 **NOTE:** `setpci` modifies the adapter's configuration registers to allow it to read up to 4k bytes at a time (for transmits). However, for some systems the behavior after modifying this register may be undefined (possibly errors of some kind). A power-cycle, hard reset or explicitly setting the `e6` register back to 22 (`setpci -d 8086:1a48 e6.b=22`) may be required to get back to a stable configuration.

- COPY these lines and paste them into `ixgb_perf.sh`:

```
#!/bin/bash
echo "configuring network performance , edit this file to change the interface
or device ID of 10GbE card"
# set mmrbc to 4k reads, modify only Intel 10GbE device IDs
# replace 1a48 with appropriate 10GbE device's ID installed on the system, if
needed.
setpci -d 8086:1a48 e6.b=2e
# set the MTU (max transmission unit) - it requires your switch and clients to
change too!
# set the txqueuelen
# your ixgb adapter should be loaded as eth1 for this to work, change if needed
ifconfig eth1 mtu 9000 txqueuelen 1000 up
# call the sysctl utility to modify /proc/sys entries
sysctl -p ./sysctl_ixgb.conf
- END ixgb_perf.sh
```

- COPY these lines and paste them into `sysctl_ixgb.conf`:

```
# some of the defaults may be different for your kernel
# call this file with sysctl -p <this file>
# these are just suggested values that worked well to increase throughput in
# several network benchmark tests, your mileage may vary

### IPV4 specific settings
net.ipv4.tcp_timestamps = 0 # turns TCP timestamp support off, default 1, reduces
CPU use
net.ipv4.tcp_sack = 0 # turn SACK support off, default on
# on systems with a VERY fast bus -> memory interface this is the big gainer
net.ipv4.tcp_rmem = 10000000 10000000 10000000 # sets min/default/max TCP read
buffer, default 4096 87380 174760
net.ipv4.tcp_wmem = 10000000 10000000 10000000 # sets min/pressure/max TCP write
buffer, default 4096 16384 131072
```

```
net.ipv4.tcp_mem = 10000000 10000000 10000000 # sets min/pressure/max TCP buffer
space, default 31744 32256 32768

net.ipv4.tcp_max_syn_backlog = 1000

### CORE settings (mostly for socket and UDP effect)
net.core.rmem_max = 524287 # maximum receive socket buffer size, default 131071
net.core.wmem_max = 524287 # maximum send socket buffer size, default 131071
net.core.rmem_default = 524287 # default receive socket buffer size, default
65535
net.core.wmem_default = 524287 # default send socket buffer size, default 65535
net.core.optmem_max = 524287 # maximum amount of option memory buffers, default
10240
net.core.netdev_max_backlog = 300000 # number of unprocessed input packets before
kernel starts dropping them, default 300
- END sysctl_ixgb.conf
```

Edit the ixgb_perf.sh script if necessary to change eth1 to whatever interface your ixgb driver is using and/or replace '1a48' with appropriate 10GbE device's ID installed on the system.



NOTE: Unless these scripts are added to the boot process, these changes will only last only until the next system reboot.

Appendix D. S2io Performance Tuning Requirements

Windows Registry name: **MTU**

Windows Advanced Tab Name: MTU

Windows Default: 1500

Linux name: **mtu**

Linux Default: 1500

Range: 1500 – 9600

Description: The maximum number of bytes of data in an Ethernet frame.. IF jumbo frames are being used with MTU greater than 1500 bytes, coalescing interrupts must be varied accordingly to the bandwidth being used in both the transmit and receive sides, as well as the number of packets to be processed per DCP.

Jumbo frame (since 9600 bytes) is recommended to achieve maximum performance.

Windows Usage: This value can be changed in registry.

Linux Usage: This value can be changed using “ifconfig”.

Windows Registry name: **lso_enable**

Windows Advanced Tab name: LSO

Windows default: Enable (0x1)

Linux name: **tso**

Linux Default: Enable (0x1)

Range: Enable or Disable

Description: Large send operation to be offloaded to hardware.

Windows Usage: This value can be changed in registry or via “Advanced” tab.

Linux usage: This value can be changed using “ethtool”.

Windows Registry name: **cksum_offload_enable**

Windows Default: Enable (0x1)

Linux name: **[tx|rx]**

Linux Default: Enable (0x1)

Range: Enable or Disable

Description: Checksum (Tx/Rx) operation to be offloaded to hardware.

Windows Usage: This value can be changed in registry.

Linux Usage: This value can be changed using “ethtool”.

Windows Registry name: **max_read_byte_cnt**

Windows Default: 3 (0x3)

Linux name: **max_read_byte_cnt**

Linux Default: 3 (0x3)

Range: 0 – 3

Description: This is a PCI-X command. It sets the maximum byte count the device uses when initiating a sequence with one of the burst memory read commands.

Windows Usage: This value can be changed in registry.

Linux usage: This value can be entered individually as part of the “insmod” command line argument.

Note: S2io provided other notes on performance tuning; however, Chelsio illustrated to us how these additional parameter settings would not apply to the netperf test results.

Appendix E. Server Disclosure RH Linux 9.0 (2.6.6)

OS Info

Linux warpl1 2.6.6 #4 Wed Jul 28 09:54:16 PDT 2004 x86_64 x86_64 x86_64
GNU/Linux

CPU Info

processor : 0
vendor_id : AuthenticAMD
cpu family : 15
model : 5
model name : AMD Opteron(tm) Processor 248
stepping : 8
cpu MHz : 2205.045
cache size : 1024 KB
fpu : yes
fpu_exception : yes
cpuid level : 1
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36 clflush mmx fxsr sse sse2 syscall nx mmxext lm 3dnowext 3dnow
bogomips : 4325.37
TLB size : 1088 4K pages
clflush size : 64
cache_alignment : 64
address sizes : 40 bits physical, 48 bits virtual
power management: ts ttp

Memory Info

MemTotal: 1029744 kB
MemFree: 978692 kB
Buffers: 4128 kB
Cached: 13316 kB
SwapCached: 0 kB
Active: 10576 kB
Inactive: 8884 kB
HighTotal: 0 kB
HighFree: 0 kB
LowTotal: 1029744 kB
LowFree: 978692 kB
SwapTotal: 2096472 kB
SwapFree: 2096472 kB
Dirty: 80 kB
Writeback: 0 kB
Mapped: 4880 kB
Slab: 24948 kB
Committed_AS: 14904 kB
PageTables: 304 kB
VmallocTotal: 536870911 kB
VmallocUsed: 1256 kB
VmallocChunk: 536869655 kB
HugePages_Total: 0
HugePages_Free: 0
Hugepagesize: 2048 kB

Interface Info

```
eth0      Link encap:Ethernet  HWaddr 00:07:43:00:14:16
          inet addr:102.50.50.67  Bcast:102.50.50.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5  errors:0  dropped:0  overruns:0  frame:0
          TX packets:4  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:699 (699.0 b)  TX bytes:184 (184.0 b)
          Interrupt:29  Memory:fe200000-fe200fff

eth1      Link encap:Ethernet  HWaddr 00:50:45:01:0F:6C
          inet addr:10.192.166.67  Bcast:10.192.175.255  Mask:255.255.240.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:618  errors:0  dropped:0  overruns:0  frame:0
          TX packets:174  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:117276 (114.5 Kb)  TX bytes:12376 (12.0 Kb)
          Interrupt:27

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:5  errors:0  dropped:0  overruns:0  frame:0
          TX packets:5  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:471 (471.0 b)  TX bytes:471 (471.0 b)
```

Boot up Info

```
Bootdata ok (command line is ro root=/dev/hda2)
Linux version 2.6.6 (root@cleopatra.asicdesigners.com) (gcc version 3.2.3
20030502 (Red Hat Linux 3.2.3-34)) #4 Wed Jul 28 09:54:16 PDT 2004
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009b800 (usable)
 BIOS-e820: 000000000009b800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000cc000 - 00000000000100000 (reserved)
 BIOS-e820: 00000000000100000 - 000000000003ff70000 (usable)
 BIOS-e820: 000000000003ff70000 - 000000000003ff76000 (ACPI data)
 BIOS-e820: 000000000003ff76000 - 000000000003ff80000 (ACPI NVS)
 BIOS-e820: 000000000003ff80000 - 0000000000040000000 (reserved)
 BIOS-e820: 00000000000fec00000 - 00000000000fec00400 (reserved)
 BIOS-e820: 00000000000fee00000 - 00000000000fee01000 (reserved)
 BIOS-e820: 00000000000fff80000 - 00000000000100000000 (reserved)
ACPI: have wakeup address 0x10000001000
No mptable found.
No mptable found.
On node 0 totalpages: 262000
  DMA zone: 4096 pages, LIFO batch:1
  Normal zone: 257904 pages, LIFO batch:16
  HighMem zone: 0 pages, LIFO batch:1
ACPI: RSDP (v002 PTLTD ) @
0x00000000000000f6920
ACPI: XSDT (v001 PTLTD XSDT 0x06040000 LTP 0x00000000) @
0x000000000003ff734d8
ACPI: FADT (v003 AMD HAMMER 0x06040000 PTEC 0x000f4240) @
0x000000000003ff75e46
```

```

ACPI: MADT (v001 PTLTD          APIC    0x06040000  LTP 0x00000000) @
0x0000000003ff75f3a
ACPI: SPCR (v001 PTLTD  $UCRTBL$ 0x06040000  PTL  0x00000001) @
0x0000000003ff75fb0
ACPI: DSDT (v001 AMD-K8  AMDACPI 0x06040000  MSFT 0x0100000e) @
0x0000000000000000
ACPI: Local APIC address 0xfee00000
ACPI: LAPIC (acpi_id[0x00] lapic_id[0x00] enabled)
Processor #0 15:5 APIC version 16
ACPI: LAPIC (acpi_id[0x01] lapic_id[0x01] enabled)
Processor #1 15:5 APIC version 16
WARNING: NR_CPUS limit of 1 reached. Processor ignored.
ACPI: LAPIC_NMI (acpi_id[0x00] high edge lint[0x1])
ACPI: LAPIC_NMI (acpi_id[0x01] high edge lint[0x1])
ACPI: IOAPIC (id[0x02] address[0xfec00000] global_irq_base[0x0])
IOAPIC[0]: Assigned apic_id 2
IOAPIC[0]: apic_id 2, version 17, address 0xfec00000, GSI 0-23
ACPI: IOAPIC (id[0x03] address[0xfe500000] global_irq_base[0x18])
IOAPIC[1]: Assigned apic_id 3
IOAPIC[1]: apic_id 3, version 17, address 0xfe500000, GSI 24-27
ACPI: IOAPIC (id[0x04] address[0xfe501000] global_irq_base[0x1c])
IOAPIC[2]: Assigned apic_id 4
IOAPIC[2]: apic_id 4, version 17, address 0xfe501000, GSI 28-31
ACPI: INT_SRC_OVR (bus 0 bus_irq 0 global_irq 2 high edge)
ACPI: IRQ0 used by override.
ACPI: IRQ2 used by override.
ACPI: IRQ9 used by override.
Using ACPI (MADT) for SMP configuration information
Checking aperture...
CPU 0: aperture @ 0 size 32 MB
No AGP bridge found
Built 1 zonelists
Kernel command line: ro root=/dev/hda2 console=tty0
Initializing CPU#0
PID hash table entries: 16 (order 4: 256 bytes)
time.c: Using 1.193182 MHz PIT timer.
time.c: Detected 2205.045 MHz processor.
Console: colour VGA+ 80x25
Memory: 1029240k/1048000k available (2091k kernel code, 18008k reserved, 1085k
data, 156k init)
Calibrating delay loop... 4325.37 BogomIPS
Dentry cache hash table entries: 131072 (order: 8, 1048576 bytes)
Inode-cache hash table entries: 65536 (order: 7, 524288 bytes)
Mount-cache hash table entries: 256 (order: 0, 4096 bytes)
CPU: L1 I Cache: 64K (64 bytes/line), D cache 64K (64 bytes/line)
CPU: L2 Cache: 1024K (64 bytes/line)
CPU: AMD Opteron(tm) Processor 248 stepping 08
POSIX conformance testing by UNIFIX
Using local APIC NMI watchdog using perfctr0
ENABLING IO-APIC IRQs
init IO_APIC IRQs
 IO-APIC (apicid-pin) 2-0, 2-16, 2-17, 2-18, 2-19, 2-20, 2-21, 2-22, 2-23, 3-0,
3-1, 3-2, 3-3, 4-0, 4-1, 4-2, 4-3 not connected.
..TIMER: vector=0x31 pin1=2 pin2=-1
Using local APIC timer interrupts.
Detected 12.528 MHz APIC timer.
NET: Registered protocol family 16
PCI: Using configuration type 1

```

```

mtrr: v2.0 (20020519)
ACPI: Subsystem revision 20040326
  tbxface-0117 [03] acpi_load_tables      : ACPI Tables successfully acquired
Parsing all Control
Methods:.....
Table [DSDT](id F004) - 297 Objects with 30 Devices 60 Methods 30 Regions
ACPI Namespace successfully loaded at root ffffffff80431ee0
evxfevnt-0093 [04] acpi_enable           : Transition to ACPI mode successful
evgpeblk-0867 [06] ev_create_gpe_block   : GPE 00 to 15 [_GPE] 2 regs at
00000000000008020 on int 9
evgpeblk-0925 [06] ev_create_gpe_block   : Found 0 Wake, Enabled 2 Runtime GPEs
in this block
evgpeblk-0867 [06] ev_create_gpe_block   : GPE 176 to 207 [_GPE] 4 regs at
000000000000080B0 on int 9
evgpeblk-0925 [06] ev_create_gpe_block   : Found 0 Wake, Enabled 0 Runtime GPEs
in this block
Completing Region/Field/Buffer/Package
initialization:.....
Initialized 30/30 Regions 0/0 Fields 15/15 Buffers 17/17 Packages (306 nodes)
Executing all Device _STA and _INI methods:.....
33 Devices found containing: 33 _STA, 0 _INI methods
ACPI: Interpreter enabled
ACPI: Using IOAPIC for interrupt routing
ACPI: PCI Root Bridge [PCI0] (00:00)
PCI: Probing PCI hardware (bus 00)
ACPI: PCI Interrupt Link [LNKA] (IRQs 3 5 10 11) *0, disabled.
ACPI: PCI Interrupt Link [LNKB] (IRQs 3 *5 10 11)
ACPI: PCI Interrupt Link [LNKC] (IRQs 3 5 *10 11)
ACPI: PCI Interrupt Link [LNKD] (IRQs 3 5 10 *11)
ACPI: PCI Interrupt Routing Table [\_SB_.PCI0.TP2P._PRT]
ACPI: PCI Interrupt Routing Table [\_SB_.PCI0.G0PA._PRT]
ACPI: PCI Interrupt Routing Table [\_SB_.PCI0.G0PB._PRT]
IOAPIC[0]: Set PCI routing entry (2-16 -> 0xa9 -> IRQ 16 Mode:1 Active:1)
00:01:00[A] -> 2-16 -> IRQ 16
IOAPIC[0]: Set PCI routing entry (2-17 -> 0xb1 -> IRQ 17 Mode:1 Active:1)
00:01:00[B] -> 2-17 -> IRQ 17
IOAPIC[0]: Set PCI routing entry (2-18 -> 0xb9 -> IRQ 18 Mode:1 Active:1)
00:01:00[C] -> 2-18 -> IRQ 18
IOAPIC[0]: Set PCI routing entry (2-19 -> 0xc1 -> IRQ 19 Mode:1 Active:1)
00:01:00[D] -> 2-19 -> IRQ 19
IOAPIC[1]: Set PCI routing entry (3-1 -> 0xc9 -> IRQ 25 Mode:1 Active:1)
00:02:01[A] -> 3-1 -> IRQ 25
IOAPIC[1]: Set PCI routing entry (3-2 -> 0xd1 -> IRQ 26 Mode:1 Active:1)
00:02:01[B] -> 3-2 -> IRQ 26
IOAPIC[1]: Set PCI routing entry (3-3 -> 0xd9 -> IRQ 27 Mode:1 Active:1)
00:02:01[C] -> 3-3 -> IRQ 27
IOAPIC[1]: Set PCI routing entry (3-0 -> 0xe1 -> IRQ 24 Mode:1 Active:1)
00:02:01[D] -> 3-0 -> IRQ 24
IOAPIC[2]: Set PCI routing entry (4-1 -> 0xe9 -> IRQ 29 Mode:1 Active:1)
00:03:01[A] -> 4-1 -> IRQ 29
IOAPIC[2]: Set PCI routing entry (4-2 -> 0x32 -> IRQ 30 Mode:1 Active:1)
00:03:01[B] -> 4-2 -> IRQ 30
IOAPIC[2]: Set PCI routing entry (4-3 -> 0x3a -> IRQ 31 Mode:1 Active:1)
00:03:01[C] -> 4-3 -> IRQ 31
IOAPIC[2]: Set PCI routing entry (4-0 -> 0x42 -> IRQ 28 Mode:1 Active:1)
00:03:01[D] -> 4-0 -> IRQ 28
number of MP IRQ sources: 15.
number of IO-APIC #2 registers: 24.

```

number of IO-APIC #3 registers: 4.
 number of IO-APIC #4 registers: 4.
 testing the IO APIC.....

IO APIC #2.....

.... register #00: 02000000
 : physical APIC id: 02
 register #01: 00170011
 : max redirection entries: 0017
 : PRQ implemented: 0
 : IO APIC version: 0011
 register #02: 02000000
 : arbitration: 02
 IRQ redirection table:

NR	Log	Phy	Mask	Trig	IRR	Pol	Stat	Dest	Deli	Vect:
00	000	00	1	0	0	0	0	0	0	00
01	001	01	0	0	0	0	0	1	1	39
02	001	01	0	0	0	0	0	1	1	31
03	001	01	0	0	0	0	0	1	1	41
04	001	01	0	0	0	0	0	1	1	49
05	001	01	0	0	0	0	0	1	1	51
06	001	01	0	0	0	0	0	1	1	59
07	001	01	0	0	0	0	0	1	1	61
08	001	01	0	0	0	0	0	1	1	69
09	001	01	0	1	0	1	0	1	1	71
0a	001	01	0	0	0	0	0	1	1	79
0b	001	01	0	0	0	0	0	1	1	81
0c	001	01	0	0	0	0	0	1	1	89
0d	001	01	0	0	0	0	0	1	1	91
0e	001	01	0	0	0	0	0	1	1	99
0f	001	01	0	0	0	0	0	1	1	A1
10	001	01	1	1	0	1	0	1	1	A9
11	001	01	1	1	0	1	0	1	1	B1
12	001	01	1	1	0	1	0	1	1	B9
13	001	01	1	1	0	1	0	1	1	C1
14	000	00	1	0	0	0	0	0	0	00
15	000	00	1	0	0	0	0	0	0	00
16	000	00	1	0	0	0	0	0	0	00
17	000	00	1	0	0	0	0	0	0	00

IO APIC #3.....

.... register #00: 03000000
 : physical APIC id: 03
 register #01: 00030011
 : max redirection entries: 0003
 : PRQ implemented: 0
 : IO APIC version: 0011
 register #02: 00000000
 : arbitration: 00
 IRQ redirection table:

NR	Log	Phy	Mask	Trig	IRR	Pol	Stat	Dest	Deli	Vect:
00	001	01	1	1	0	1	0	1	1	E1
01	001	01	1	1	0	1	0	1	1	C9
02	001	01	1	1	0	1	0	1	1	D1
03	001	01	1	1	0	1	0	1	1	D9

IO APIC #4.....

.... register #00: 04000000

```

.....      : physical APIC id: 04
.... register #01: 00030011
.....      : max redirection entries: 0003
.....      : PRQ implemented: 0
.....      : IO APIC version: 0011
.... register #02: 00000000
.....      : arbitration: 00
.... IRQ redirection table:
  NR Log Phy Mask Trig IRR Pol Stat Dest Deli Vect:
  00 001 01 1 1 0 1 0 1 1 42
  01 001 01 1 1 0 1 0 1 1 E9
  02 001 01 1 1 0 1 0 1 1 32
  03 001 01 1 1 0 1 0 1 1 3A
IRQ to pin mappings:
IRQ0 -> 0:2
IRQ1 -> 0:1
IRQ3 -> 0:3
IRQ4 -> 0:4
IRQ5 -> 0:5
IRQ6 -> 0:6
IRQ7 -> 0:7
IRQ8 -> 0:8
IRQ9 -> 0:9
IRQ10 -> 0:10
IRQ11 -> 0:11
IRQ12 -> 0:12
IRQ13 -> 0:13
IRQ14 -> 0:14
IRQ15 -> 0:15
IRQ16 -> 0:16
IRQ17 -> 0:17
IRQ18 -> 0:18
IRQ19 -> 0:19
IRQ24 -> 1:0
IRQ25 -> 1:1
IRQ26 -> 1:2
IRQ27 -> 1:3
IRQ28 -> 2:0
IRQ29 -> 2:1
IRQ30 -> 2:2
IRQ31 -> 2:3
..... done.
pci_irq-0316 [11] acpi_pci_irq_derive : Unable to derive IRQ for device
0000:00:07.2
ACPI: No IRQ known for interrupt pin D of device 0000:00:07.2
PCI: Using ACPI for IRQ routing
PCI-DMA: Disabling IOMMU.
IA32 emulation $Id: sys_ia32.c,v 1.32 2002/03/24 13:02:28 ak Exp $
Total HugeTLB memory allocated, 0
VFS: Disk quotas dquot_6.5.1
Dquot-cache hash table entries: 512 (order 0, 4096 bytes)
Initializing Cryptographic API
Real Time Clock Driver v1.12
Linux agpgart interface v0.100 (c) Dave Jones
RAMDISK driver initialized: 16 RAM disks of 8192K size 1024 blocksize
Intel(R) PRO/10GbE Network Driver - version 1.0.65
Copyright (c) 2001-2004 Intel Corporation.
Chelsio TOE Network Driver - version 1.0

```

```
0000:03:01.0: PM3393 HW RESET: 0 pl4_reset(0x0040) val(0x0268)
is_pl4_outof_lock(0x0000) xau_i_locked(0x0200)
divert: allocating divert_blk for eth0
eth0: Chelsio T110 1x1000BaseX TOE (rev 1), PCIX 133MHz/64-bit
tg3.c:v3.3 (April 27, 2004)
divert: allocating divert_blk for eth1
eth1: Tigon3 [partno(BCM95702A20) rev 1002 PHY(5703)] (PCI:66MHz:32-bit)
10/100/1000BaseT Ethernet 00:50:45:01:0f:6c
eth1: HostTXDS[0] RXcsums[1] LinkChgREG[0] MIirq[0] ASF[0] Split[0] WireSpeed[1]
TSOcap[1]
divert: allocating divert_blk for eth2
eth2: Tigon3 [partno(BCM95702A20) rev 1002 PHY(5703)] (PCI:66MHz:32-bit)
10/100/1000BaseT Ethernet 00:50:45:01:0f:6d
eth2: HostTXDS[0] RXcsums[1] LinkChgREG[0] MIirq[0] ASF[0] Split[0] WireSpeed[1]
TSOcap[1]
divert: not allocating divert_blk for non-ethernet device lo
Uniform Multi-Platform E-IDE driver Revision: 7.00alpha2
ide: Assuming 33MHz system bus speed for PIO modes; override with idebus=xx
AMD8111: IDE controller at PCI slot 0000:00:07.1
AMD8111: chipset revision 3
AMD8111: not 100% native mode: will probe irqs later
AMD8111: 0000:00:07.1 (rev 03) UDMA133 controller
    ide0: BM-DMA at 0x1020-0x1027, BIOS settings: hda:DMA, hdb:DMA
    ide1: BM-DMA at 0x1028-0x102f, BIOS settings: hdc:pio, hdd:pio
hda: WDC WD800JB-00FMA0, ATA DISK drive
hdb: WDC WD800JB-00FSA0, ATA DISK drive
Using anticipatory io scheduler
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hda: max request size: 128KiB
hda: 156301488 sectors (80026 MB) w/8192KiB Cache, CHS=65535/16/63, UDMA(100)
    hda: hda1 hda2 hda3
hdb: max request size: 1024KiB
hdb: 156301488 sectors (80026 MB) w/8192KiB Cache, CHS=16383/255/63, UDMA(100)
    hdb: hdb1 hdb2 hdb3
ide-floppy driver 0.99.newide
mice: PS/2 mouse device common for all mice
serio: i8042 AUX port at 0x60,0x64 irq 12
serio: i8042 KBD port at 0x60,0x64 irq 1
input: AT Translated Set 2 keyboard on isa0060/serio0
md: md driver 0.90.0 MAX_MD_DEVS=256, MD_SB_DISKS=27
NET: Registered protocol family 2
IP: routing cache hash table of 8192 buckets, 64Kbytes
TCP: Hash tables configured (established 262144 bind 65536)
Initializing IPsec netlink socket
NET: Registered protocol family 1
NET: Registered protocol family 17
NET: Registered protocol family 8
NET: Registered protocol family 20
ACPI: (supports S0 S1 S4 S5)
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
kjournald starting. Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Freeing unused kernel memory: 156k freed
EXT3 FS on hda2, internal journal
Adding 2096472k swap on /dev/hda3. Priority:-1 extents:1
```

```
kjournald starting. Commit interval 5 seconds
EXT3 FS on hda1, internal journal
EXT3-fs: mounted filesystem with ordered data mode.
eth0: link is up at 10Gb/s, full duplex
tg3: eth1: Link is up at 100 Mbps, full duplex.
tg3: eth1: Flow control is off for TX and off for RX.
```

Appendix F. Benchmark Script

A shell script was written to automate the test initiation, monitoring and result collection processes. The script takes as input the test parameters which are varied, the various Iperf command line options as well as other useful settings such as test length, paths to the script and results directories and server information.

To collect throughput information for multiple connection tests, the script calls Iperf, and to collect transaction rate information for latency tests the script calls Netperf.

Netperf is used with the following command line options:

- “H”: to specify the host running the netserver
- “t”: to specify TCP_STREAM or TCP_RR
- “c” and “C”: to measure local and remote CPU utilization
- “l”: to specify how long to run the test
- “s” and “S”: to specify local and remote socket buffer size
- “m” and “M” to specify local and remote I/O size
- “f”: to specify the output format

Iperf is used with the following command line options:

- “C”: to specify the host running the iperf server
- “P”: to specify the number of connections
- “l”: to specify the I/O size
- “t”: to specify how long to run the test
- “w”: to specify the socket buffer size

Depending on the type of test required, the script calls either tool with the appropriate values in the command line options.

The script uses mpstat to measure the CPU utilization while the run is in progress.

At the end of each run, the script parses the tool’s output and retrieves the relevant performance measure such as achieved throughput or transaction rate, appends it to the collected CPU utilization metrics and saves it in the test suite’s log file before proceeding to the next run.

VeriTest (www.veritest.com), the testing division of Lionbridge Technologies, Inc., provides outsourced testing solutions that maximize revenue and reduce costs for our clients. For companies who use high-tech products as well as those who produce them, smoothly functioning technology is essential to business success. VeriTest helps our clients identify and correct technology problems in their products and in their line of business applications by providing the widest range of testing services available.

VeriTest created the suite of industry-standard benchmark software that includes WebBench, NetBench, Winstone, and WinBench. We've distributed over 20 million copies of these tools, which are in use at every one of the 2001 Fortune 100 companies. Our Internet BenchMark service provides the definitive ratings for Internet Service Providers in the US, Canada, and the UK.

Under our former names of ZD Labs and eTesting Labs, and as part of VeriTest since July of 2002, we have delivered rigorous, objective, independent testing and analysis for over a decade. With the most knowledgeable staff in the business, testing facilities around the world, and almost 1,600 dedicated network PCs, VeriTest offers our clients the expertise and equipment necessary to meet all their testing needs.

For more information email us at info@veritest.com or call us at 919-380-2800.

Disclaimer of Warranties; Limitation of Liability:

VERITEST HAS MADE REASONABLE EFFORTS TO ENSURE THE ACCURACY AND VALIDITY OF ITS TESTING, HOWEVER, VERITEST SPECIFICALLY DISCLAIMS ANY WARRANTY, EXPRESSED OR IMPLIED, RELATING TO THE TEST RESULTS AND ANALYSIS, THEIR ACCURACY, COMPLETENESS OR QUALITY, INCLUDING ANY IMPLIED WARRANTY OF FITNESS FOR ANY PARTICULAR PURPOSE. ALL PERSONS OR ENTITIES RELYING ON THE RESULTS OF ANY TESTING DO SO AT THEIR OWN RISK, AND AGREE THAT VERITEST, ITS EMPLOYEES AND ITS SUBCONTRACTORS SHALL HAVE NO LIABILITY WHATSOEVER FROM ANY CLAIM OF LOSS OR DAMAGE ON ACCOUNT OF ANY ALLEGED ERROR OR DEFECT IN ANY TESTING PROCEDURE OR RESULT.

IN NO EVENT SHALL VERITEST BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH ITS TESTING, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL VERITEST'S LIABILITY, INCLUDING FOR DIRECT DAMAGES, EXCEED THE AMOUNTS PAID IN CONNECTION WITH VERITEST'S TESTING. CUSTOMER'S SOLE AND EXCLUSIVE REMEDIES ARE AS SET FORTH HEREIN.