Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-023

Experimental Analysis of VI Architecture

Irene Jacobson, Jim Macdonald, and David Du

March 29, 2000

Experimental Analysis of the VI Architecture

Irene B. Jacobson, James A. MacDonald and David H. C. Du[1]
Computer Science and Engineering Department
University of Minnesota

Abstract

Since the TCP/IP protocol is slow, the Virtual Interface Architecture (VIA) was designed to increase the speed of the connection between machines in a cluster. VIA allows the application program to communicate directly with the Network Interface Card (NIC). It bypasses the TCP/IP protocol stack, therefore decreasing latency and increasing data throughput. VIA was developed to work in a System Area Network (SAN) consisting of a cluster of high volume file servers. This report describes the testing that was done on the Giganet cLAN, the test results, and some comparisons with TCP/IP. The testing that was done with VIA over Giganet cLAN determined the latency for small packet sizes and throughput for large packet sizes. The lowest latency was 14.2 microseconds with a packet size of 288 bytes. The highest throughput was 87.7 MB/second. The report also compares the latency and throughput results with TCP/IP over Giganet cLAN and over Gigabit Ethernet using the Intel PRO/1000 Gigabit Server Adapter. An echo program was used to find the latency and throughput of Giganet cLAN and Gigabit Ethernet on the same machines. The average test results of both mediums were close. The lowest latency for TCP/IP over Giganet cLAN was 84 microseconds and the highest throughput was 37.6 MB/sec. The lowest latency for TCP/IP over Gigabit Ethernet was 68 microseconds and the highest throughput was 39.85 MB/sec. For packet sizes over 350 Kbytes, the throughput started to decrease for both Giganet cLAN and Gigabit Ethernet over TCP/IP. Since the throughput decreased with both mediums, the probable cause is the TCP/IP implementation in Windows NT Server 4.0. The average time it takes to make a VI connection was measured at 1.152 microseconds. So in VIA applications, the number of VI connections made should be minimized.

[1] This work is supported in part by the National Science Foundation through CISE contract number ??????

1. Introduction

While the TCP/IP protocol is commonly used for computer interconnect technology today, it still has its disadvantages. One of the problems with the protocol is that it is too slow. It is slow because there are several processes that occur while transporting a packet from one computer to another. If the TCP/IP protocol stack can be bypassed, this will eliminate making an extra copy of the data and it will avoid processes that occur in the protocol stack, such as check-sum computation. Research is being done to try to solve this and other problems by creating a protocol that will allow the application software to bypass the protocol stack and copy from the user-level memory into kernel memory. In order to avoid the bottleneck produced while using the standard communication protocol, the Virtual Interface Architecture (VIA) was developed. It was developed to work in a System Area Network (SANs) or a cluster of high volume file servers. The VI Architecture allows the application programmer to DMA from the user-level memory onto the network interface card (NIC) without using the TCP/IP protocol stack. This means the TCP/IP protocol stack can be bypassed and the latency will be decreased. Since the data will not have to be copied from the user-memory into the TCP/IP protocol buffers and then to the kernel memory, the data throughput greatly increases. Three major corporations worked on the specifications for VI Architecture: Intel, Compaq and Microsoft. Now, there are over a hundred companies working with it. The hardware system that we used is from Giganet Inc.

2. Hardware Description

The cluster is made up of four quad-processor Dell Servers running Windows NT Server 4.0 and Red Hat Linux 2.2.12-20smp. The hardware system that is used in this evaluation of the VI Architecture is called a Giganet cLAN (cluster LAN). Each computer in the Giganet cLAN has a cLAN Host Adapter installed in a PCI 64-bit slot. The cLAN host adapter is a network interface card designed to work with the Giganet switch to transport the VI Architecture. The cLAN Cluster Switch is an 8-port switch used to make the physical connection between two computers of a Giganet cluster LAN. Copper cables connect each of the NIC's to the cLAN Cluster Switch. Giganet cLAN drivers are installed under both Linux and Windows NT Server.

3. Software Description

One of the largest inhibitors in producing high-speed communication between two network nodes is the time it takes to send a message through the operating systems TCP/IP protocol stack. When a packet is being sent from one machine to another using the TCP/IP protocol, such as in the user application ftp or telnet, the following steps take place. First, the user application has to make a *write* system call to send data. Then, the data moves along the TCP/IP protocol stack to the ethernet device driver. The ethernet device driver for the NIC is responsible for getting the data from the NIC of the sending computer to the NIC of the receiving computer. The device driver will copy the data from the TCP/IP buffers into kernel memory. In order to do this; it has to lock down a segment of allocated kernel memory, so it is not swapped out. Once the data is in kernel memory, it has to be transferred directly to the NIC. The data that is on the NIC can be sent directly to the NIC on the receiving computer. Now, once the receiving computer has the data written on its NIC, the ethernet device driver for the receiver has to get the data into its kernel memory. This means that the device driver will have to initiate a DMA operation to copy the data from the NIC to the kernel memory. From the kernel memory, the data is then copied to the TCP/IP buffers and then it is passed to the end-user application program. In this process, the data copies that are made are very slow as is the TCP/IP protocol stack processing. Research is being done to try to solve this and other problems by creating a protocol that will allow the application software to bypass the protocol stack and copy from the user-level memory into kernel memory. A protocol used in SANs is the Virtual Interface (VI) Architecture. The VI Architecture is designed so that the application program can communicate directly with the NIC. The VI Architecture allows the application programmer to copy directly from user memory to the NIC. This means the TCP/IP protocol stack can be avoided and latency will be decreased. Also, since the data will not have to be copied into the TCP/IP buffer and then into the kernel memory, the time that it takes to copy large amounts of data will decrease and so the data throughput greatly increases.

4. Test Results

We did some low level testing of the VIA over the Giganet cLAN system using the Windows NT Server 4.0 operating system. An echo program was written to get the timings for data transfer. The following is the pseudo code:

```
Client Process:
Begin
   VI initialization
   Prepare memory
   Do N iterations
   Begin
      Start timer;
       Send M bytes packet
      Receive M bytes packet
      Stop timer
   End
   Disconnect VI
   Write Data to file
End
```

→ → →
← ← ←

```
Server Process:
Begin
   VI initialization
   Prepare Memory
   Do N iterations
   Begin

      Receive M bytes packet
      Send M bytes packet

   End
   Disconnect VI
End
```

First, the VI initialization is done on both the client and the server side. Then, memory is allocated and aligned on each machine. The timer is started on the client, and then it sends an M byte packet to the server process. The server process receives that M byte packet and sends it back to the client process. When the client process receives the original M byte packet, the timer stops. The round trip timing is used so that only one machine does the timing, otherwise the client and the server machines would have to synchronize clocks. Since we only want to look at the time it takes to get from one machine to another, the round trip times are calculated, but then divided by two to get the latency times. The minimum, maximum and average times are written to a file. The size of

N was always fifty, but size of M varied from each of the graphs pictured. The error bars in the graphs below represent the standard error. See appendix A for a detailed explanation. A similar client/server application was written to get the TCP/IP results and is described in section 5.

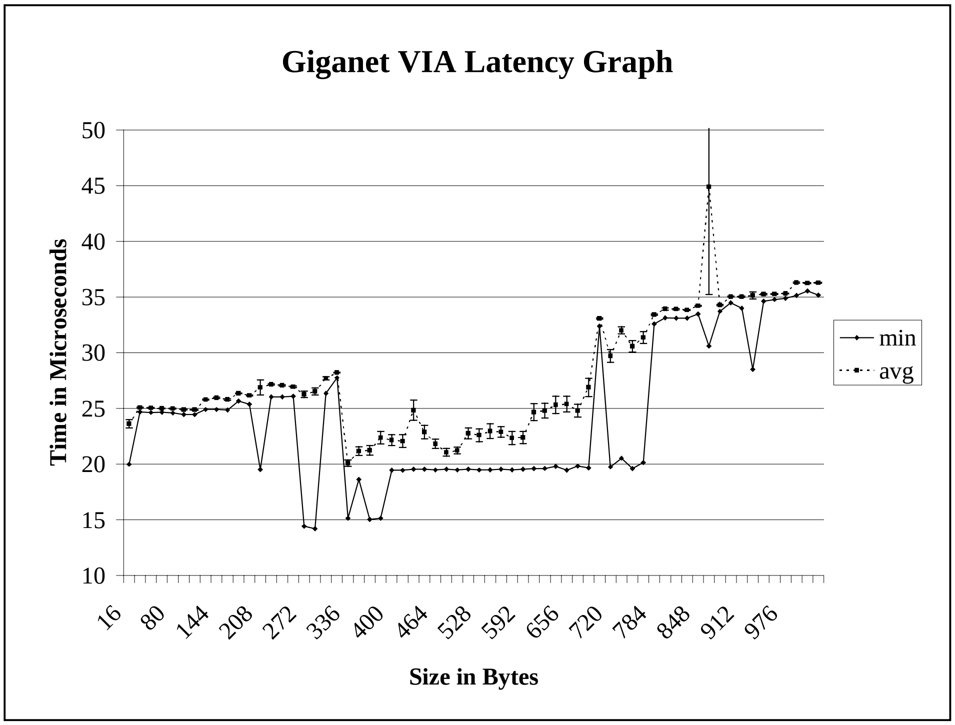

Figure 1. Latency of VIA over Giganet sending small packet sizes with a sample size of 64

4.1 Latency

Latency is the time it takes for a packet to get from a sender to a receiver. Figure 1 shows the results of the VIA over the Giganet switch, starting at 16 bytes and increasing by 16 bytes until a total of 1024 bytes of data were sent. The lines represent the minimum and average times. The error bars show the standard error for the average results. It is interesting to see that the minimum latency of 14.2 microseconds is reached when the packet size is 288 bytes. The large error bar when the packet size is 864 bytes is probably an indication of a contention for CPU and other resources. Figure 2 shows the latency results of the TCP/IP protocol with the same data sizes used in figure 1. The lines represent average times, along with the error bars, and the minimum times.

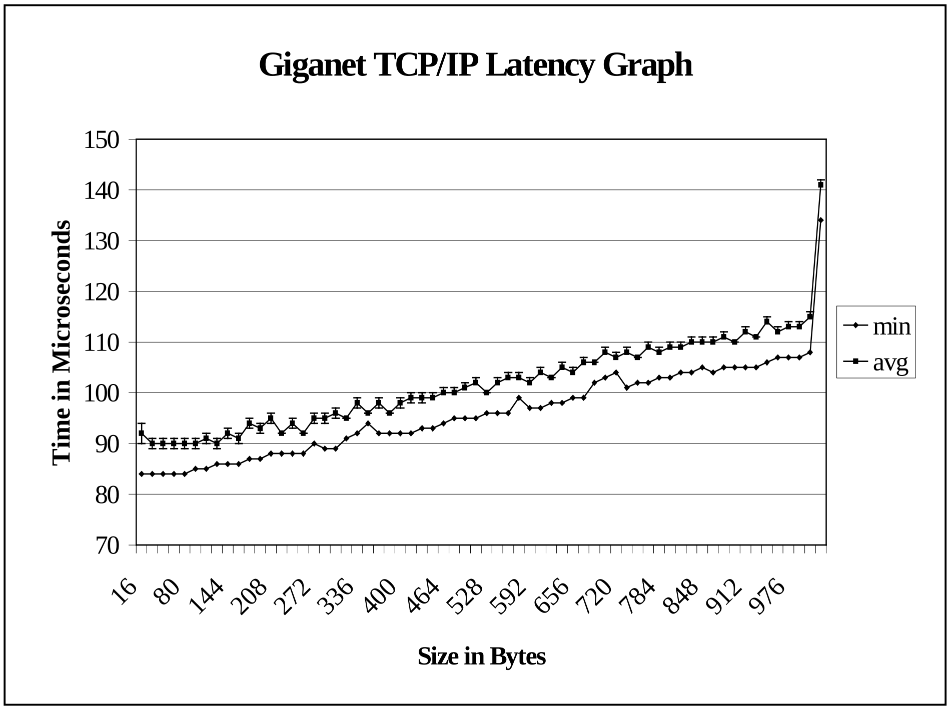

Figure 2. Latency of TCP/IP over Giganet sending small packet sizes with a sample size of 64

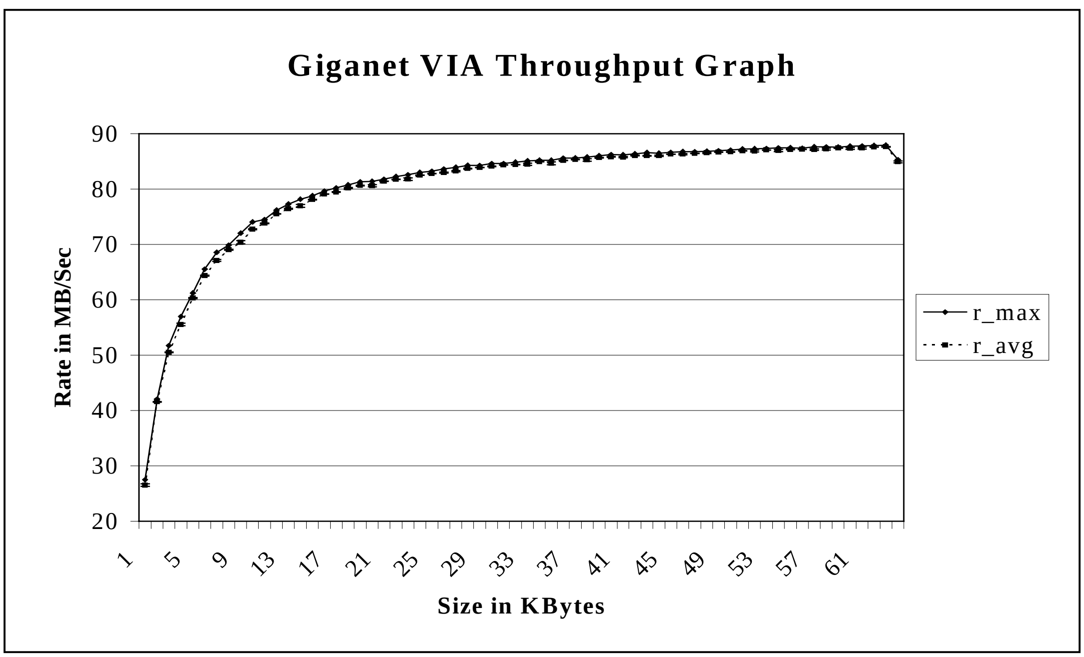

Figure 3: Throughput of VIA over Giganet sending large packet sizes with a sample size of 64

4.2 Throughput

The throughput is the amount of data that can be transferred from one machine to another and is usually measured in bytes per second. Figure 3 shows the results of the VIA over Giganet throughput starting at 1 Kbyte, incrementing by 1 Kbyte until a total of 64 Kbytes of data were sent. The lines represent the average and the maximum rates measured in Mbytes/second. From a size of 1 Kbyte to a size of 17 Kbytes, the throughput rate increases from 25 Mbytes/second to 80 Mbytes/second. From a transfer size of 19 Kbytes to 64 Kbytes the rate goes from 80 Mbytes/second to a maximum rate of 87.6 Mbytes/second.

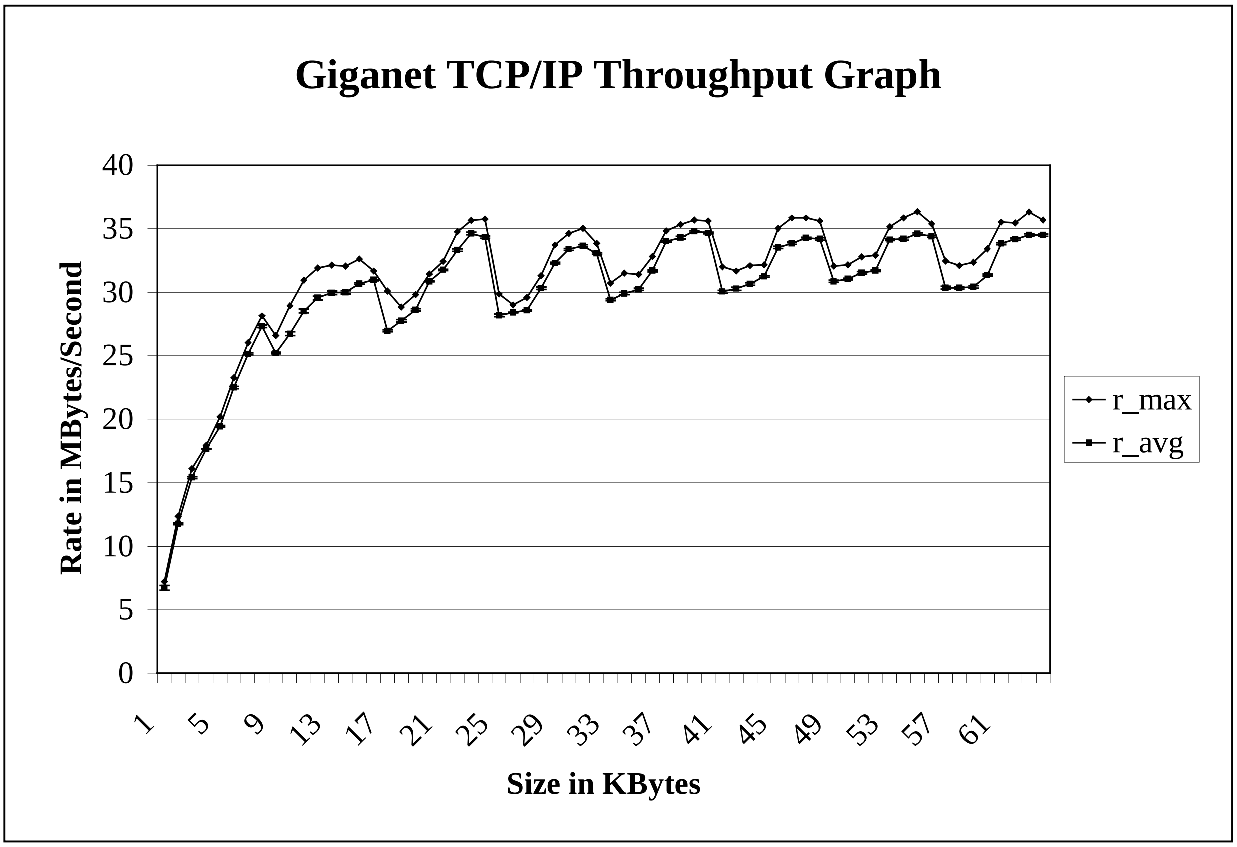

Figure 4. Throughput of TCP/IP over Giganet sending large packet sizes with a sample size of 64

Figure 4 shows the throughput of using the TCP/IP protocol over the same Giganet hardware. The packet size starts at 1 Kbyte, is incremented by 1Kbyte until a packet size of 64 Kbytes. The lines represent the maximum and average rates measured in Mbytes/second. The maximum rates fluctuate between 30 and 35 Mbytes/second with packet sizes ranging from 20 Kbytes to 64 Kbytes. The saw-tooth affect of the graph is typical for the TCP/IP protocol.

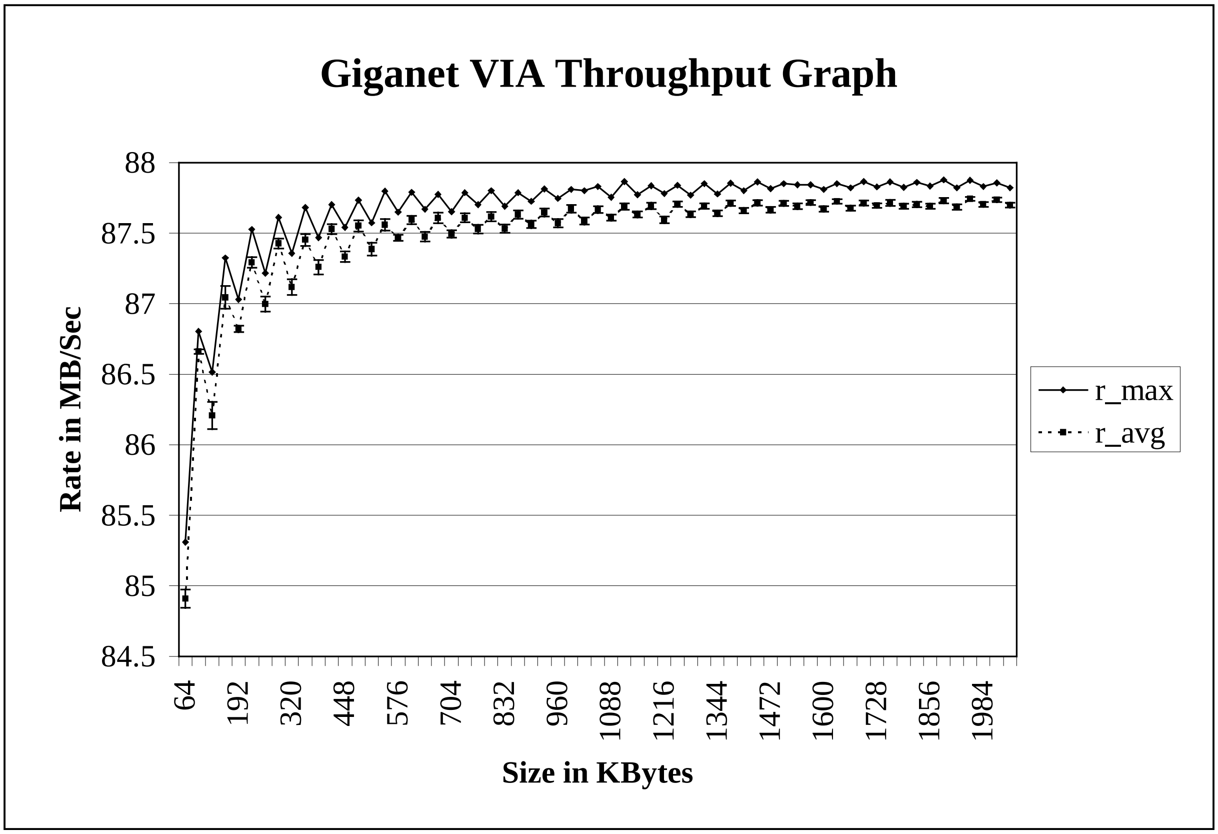

Figure 5. Throughput of VIA over Giganet sending larger packet sizes with a sample size of 64

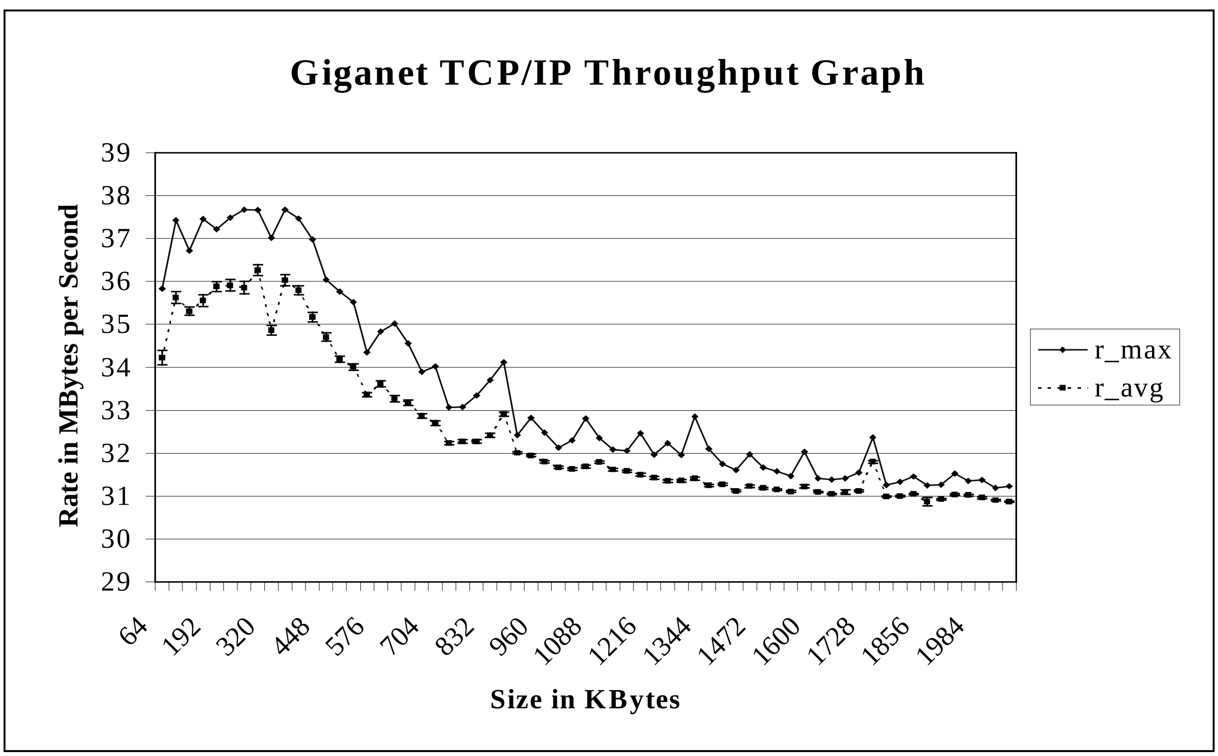

Figure 6. Throughput of TCP/IP over Giganet sending larger packet sizes with a sample size of 64

Figure 5 is a graph of the throughput of the VIA over Giganet continuing to use larger size data packets. It starts with size 64 Kbytes, increases by 32 Kbytes up to a size of 2 Mbytes. The lines on the graph represent the maximum and average rates in Mbytes/second. The throughput rates range between 85 Mbytes/second and 87.86 Mbytes/second with the large packet sizes. The zigzag affect of the graph appears to start in the beginning and then taper off as the packet sizes get larger or as time goes on. Figure 6 is a graph that represents the throughput of TCP/IP over Giganet using the same data points as in Figure 5. The packet size starts at 64 Kbytes, increases by 32 Kbytes until a size of 2 Mbytes is reached. The throughput rates range between 31 Mbytes/second and 38 Mbytes/second, as the packet size gets larger, the rate of transfer decreases. As will be shown in section 5.1, the Gigabit Ethernet card from Intel also shows this decrease in throughput with larger packet sizes. This indicates that the problem is in the Windows NT TCP/IP implementation.

4.3 VIA Connection/Disconnection Times

Here is the algorithm used for getting the connect and disconnect times:

```
VipOpenNic;
Do N iterations
Begin
        Get start time;
        VipCreateVi;
        VipConnectRequest;
        Get end time;
        connect_time[i] = end time -
start time;
        Get start time;
        VipDisconnect;
        VipDestroyVi;
        Get end time;
        disconnect_time[i] = end
time - start time;
End;
```

The time it takes to set up and take down a VIA connection from one computer to another was measured. The formula for the standard error that was used can be found in Appendix A. Out of ten thousand VIA connections, the average connection time was $1.152 \pm .002$ milliseconds. Out of ten thousand VIA disconnection times, the average time was $.148 \pm .0001$ milliseconds.

5. Comparing VIA with TCP/IP

To compare the results of VIA over Giganet with other protocols, an echo program was used to run with TCP/IP over the Giganet system and TCP/IP over a Gigabit Ethernet system using the Intel PRO/1000 Gigabit Server Adapter. Here is the algorithm of the echo program that was used:

```
Client Process:
Begin
  Setup
  Prepare buffers
  Do N iterations
  Begin
    Start timer;
    Send M bytes packet
    Receive M bytes packet
    Stop timer
  End

  Write Data to file
End
```

→ → →
← ← ←

```
Server Process:
Begin
  Setup
  Prepare Buffers
  Do N iterations
  Begin

    Receive M bytes packet
    Send M bytes packet

  End

End
```

First, the setup is done on both the client and the server side. Then, the buffers are initialized on each machine. On the client process, the timer is started then it sends an M byte packet to the server process. The server process receives that M byte packet and sends it back to the client process. When the client process receives the original M byte packet, the timer stops. Again, the round trip timing is used so that only one machine does the timing, otherwise the client and the server machines would have to synchronize clocks. To get the latency time, the round trip time was divided by two. The minimum, maximum and average times are written to a file. The standard error is calculated according to Appendix A and written to a file. The size of N was always fifty, but size of M varied from each of the graphs pictured.

5.1 Comparing Latency of TCP/IP with VIA

The latency of VIA over Giganet cLAN (GnetVI) is being compared with the latency of TCP/IP over Giganet cLAN (GNetIP) and the latency of TCP/IP over Gigabit Ethernet (Intel). The echo program was run starting with a packet size of 16 bytes, increasing by 16 bytes and ending with a packet size of 1024 bytes. The sample size is 64 and only the average latency times are being compared in the graph in Figure 7. The average latency for VIA over Giganet cLAN is between 20 and 40 microseconds while the average latency for TCP/IP protocol is between 70 and 140 microseconds. The average latency times for TCP/IP over Giganet are fairly close to the latency times of TCP/IP over the Gigabit Ethernet system. This would be expected since the delay would be due to the software protocol and not the hardware medium. According to Giganet, they claim the lowest latency for VIA over the Giganet cLAN is less than 8 microseconds.

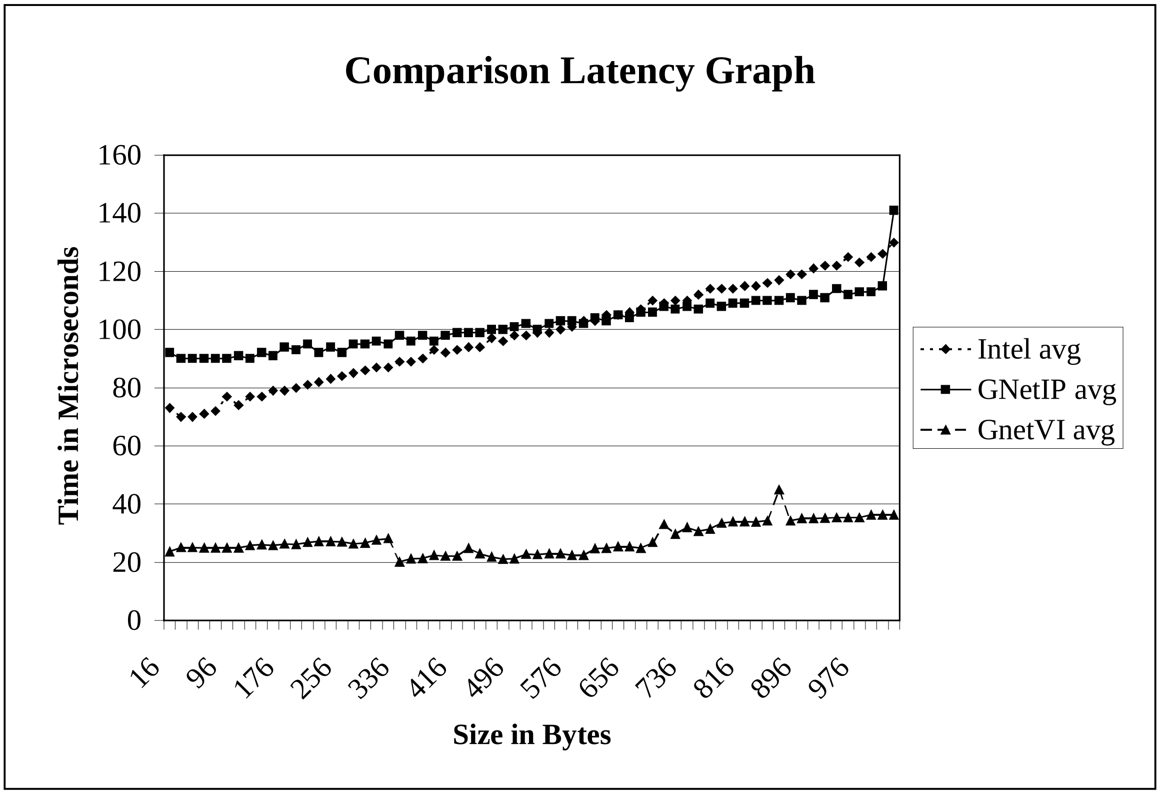

Figure 7. Latency graph of the average times for VIA and TCP/IP over Giganet cLAN and TCP/IP over Intel Gigabit Ethernet

5.2 Comparing Throughput of TCP/IP with VIA

The throughput of VIA over Giganet cLAN (GnetVI) is being compared with the throughput of TCP/IP over Giganet cLAN (GNetIP) and the throughput of TCP/IP over Gigabit Ethernet (Intel). The echo program was run starting with a packet size of 1 Kbyte, increasing by 1 Kbyte and ending with a packet size of 64 Kbytes. Figure 8 shows the average rates measured in Mbytes/second. The average throughput for VIA over Giganet cLAN when the packet size is above 18 Kbytes is between 80 and 90 MB/Sec. while the average throughput for TCP/IP is between 20 and 40 MB/Sec. The average throughput times for TCP/IP over Giganet cLAN is so close to the throughput times of TCP/IP over the Gigabit Ethernet system, it is difficult to see the difference in the 2 graphs. This would be expected since the delay in the throughput would be due to the software protocol and not the hardware medium. According to Giganet, the highest throughput for VIA over the Giganet cLAN is 160 MB/Sec full duplex. This corresponds well to the half duplex results obtained by the echo program. Figure 9 also shows the average throughput rates measured in MB/Sec. The starting packet size is 64 Kbytes it increases by 32 Kbytes until it reaches the size 2Mbytes. The average throughput for VIA over Giganet ranges between 84.9 and 87.7 MB/sec. The average throughput for TCP/IP over Giganet and Gigabit Ethernet range from 35 – 37 MB/sec for the packet sizes between 64 Kbytes and 350 Kbytes. When the data size reaches 350 Kbytes, the throughput starts to decrease. The throughput decreases to 30 – 32 MB/second. This is probably due to Windows NT implementation of TCP/IP since the decrease happens over the Giganet and over the Gigabit Ethernet mediums.

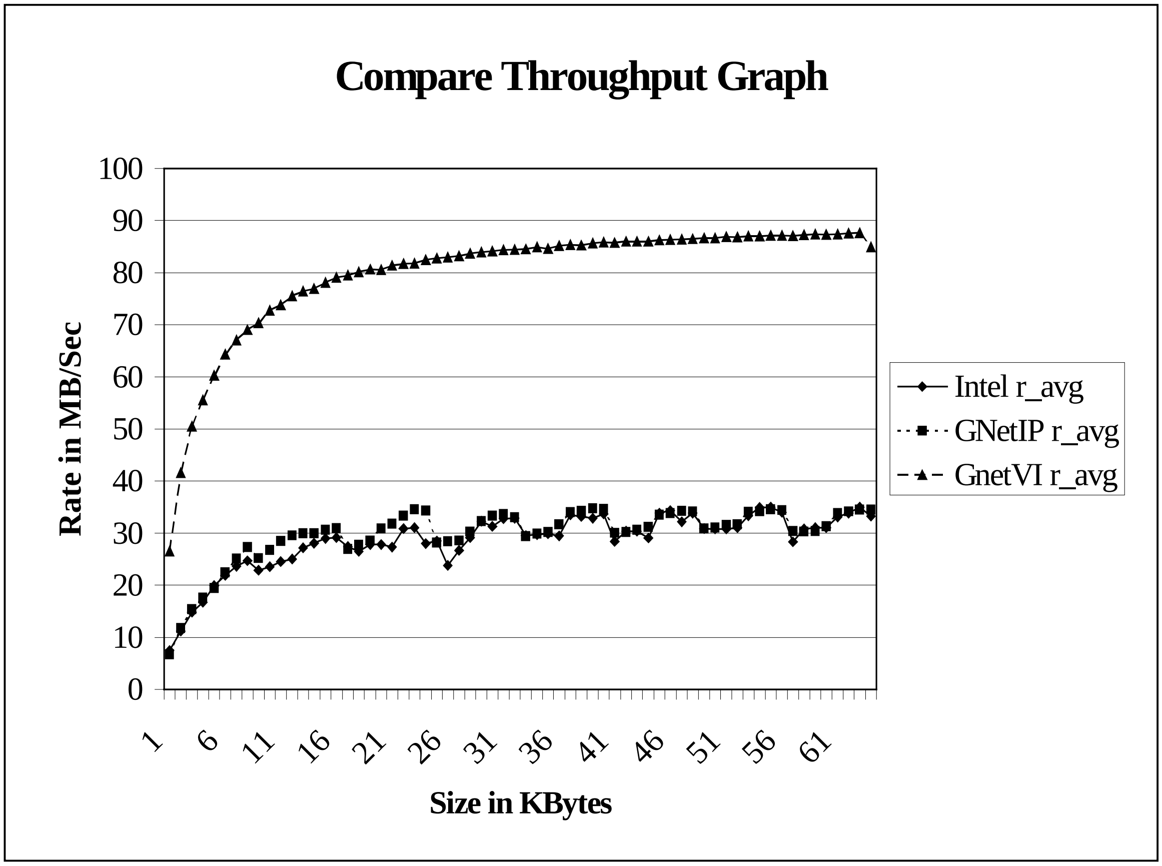

Figure 8. Throughput graph of the average times for VIA and TCP/IP over Giganet and TCP/IP over Intel

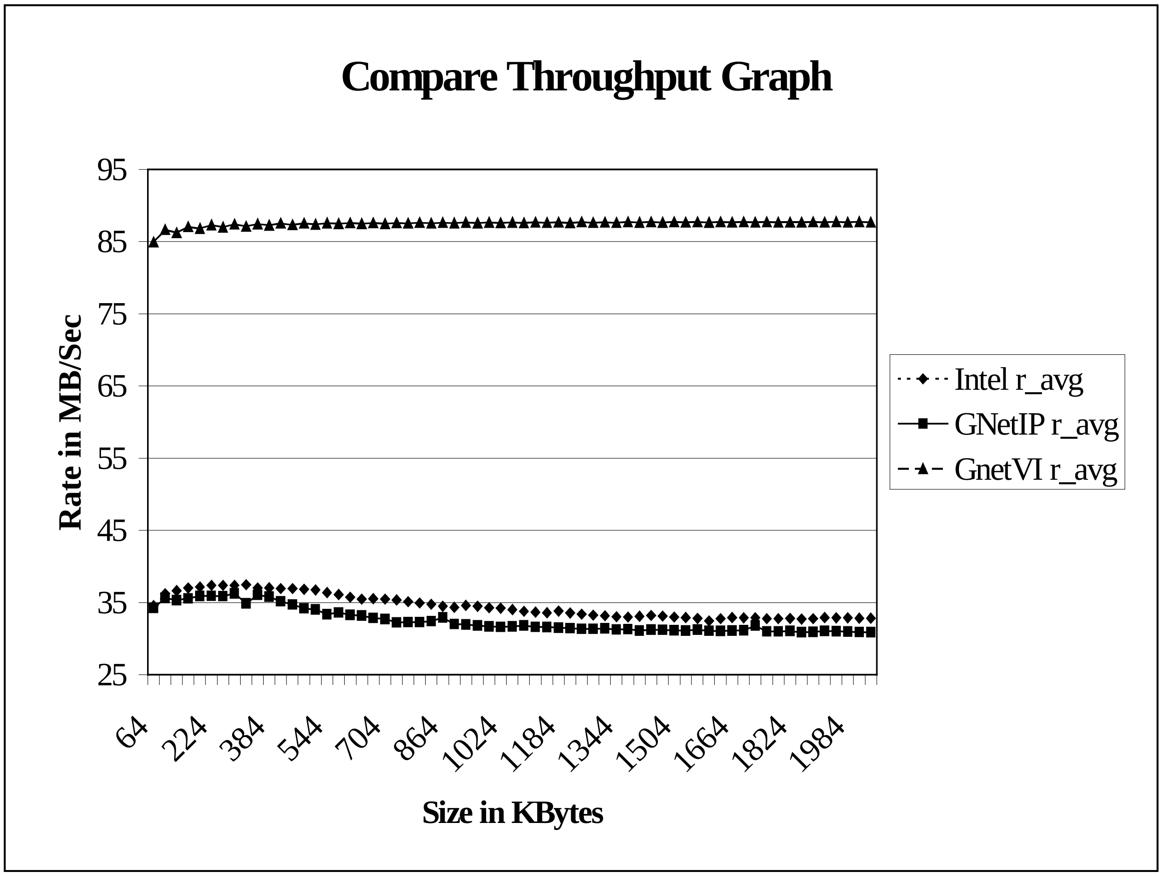

Figure 9. Throughput graph of the average times for VIA and TCP/IP over Giganet and TCP/IP over Intel

6. Conclusions and Future Work

Virtual Interface Architecture is impressive over Giganet Cluster LAN. Since the time to create and destroy a VI connection is very large, application programmers should minimize the number of times this is done. Applications used for VIA are MPI, Oracle databases and web services. Future work will include running the echo program on the same cluster of machines under Linux instead of Windows NT Server.

Appendix A

A program sends a packet of size M (in MBytes) to another computer N times. In each iteration the time t_i for i = 1 to N is recorded. The average time t_{avg} and the standard error of the average δ_t, are calculated using the following two equations:

$$t_{avg} = \frac{\sum_{i=1}^{N} t_i}{N}$$

$$\delta_t = \sqrt{\frac{\sum_{i=1}^{N} \left(t_i - t_{avg}\right)^2}{(N-1)N}}$$

In other words the error of t_{avg} is plus or minus δ_t.

The equation for the rate R in MB per second is $R = \frac{M}{t_{avg}}$, and the derivative of R with respect to t is $\frac{dR}{dt} = \frac{-M}{\left(t_{avg}\right)^2}$. Using the laws of error propagation, the equation for the error in R is:

$$\delta_r = \sqrt{\left(\frac{dR}{dt}\delta_t\right)^2}$$

Example: It takes 0.041374 $\pm$ 0.001866 seconds to send 1 MB. This gives:

R = 1MB / 0.041374 Sec. = 24.17 MB/Sec.

The error in R is

$$\delta_R = \sqrt{\left(\left(\frac{-1MB}{0.041374^2} 0.001866\right)\right)^2} = 1.044$$

R = 24.17 $\pm$ 1.044 MB/Sec.

References

[1] “The Virtual Interface Architecture” by Dave Dunning, Greg Regnier, Gary McAlpine, Don Cameron, Bill Shubert, Frank Berry, Anne Marie Merritt, Ed Gronke, Chris Dodd .Intel Corporation. IEEE Micro, Vol 18, No. 2, March/April 1998.

[2] “An Implementation and Analysis of the Virtual Interface Architecture”. Philip Buonadonna, Andrew Geweke, and David E Culler. Department of EE and Computer Science, University of California, Berkeley. www.cs.berkeley.edu/~philipb/via.

[3] “Demonstrating the benefits of Virtual Interface Architecture”. www.intel.com. jointly written by Clariion Corporation; Dell Computer Corporation;Giganet; IBM Corporation; Intel Corporation; qLogic; Visual Insights. September 1998.

[4] “Virtual Interface (VI) Architecture: Defining the path to low-cost High-Performance Scalable Clusters”. www.intel.com. 1997.

[5] “The Virtual Interface Architecture Proof-of-Concept Performance Results**” Frank Berry, Elen Deleganes, Anne Marie Merritt. www.intel.com.

[6] “Distributed Network Computing Over Local ATM Networks”. Mengjou Lin, Jenwei Hsieh, David H.C. Du, Joseph P. Thomas, and James A. MacDonald. UMSI 99/51.

[7] “HIPPI over ATM Networks; Extending Connections for Distributed Computing”. Jenwei Hsieh, David Hung-Chang Du, James A. MacDonald, Joseph P. Thomas, Jack Pugaczewski. IEEE Concurrency. October-December 1997.