1998 International Computer Symposium
Workshop on Computer Networks, Internet, and Multimedia
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Intelligent Disk-to-LAN Integration Strategy & Performance Modeling for Multimedia Servers

Liou Shiang-Chun, Huang Yueh-Min, Chen Wo-Chin, and
Liou Jun-Lin +
Dept. of Engineering Science, National Cheng-Kung University, Tainan, Taiwan
Dept. of Broadband Network Systems, CCL, ITRI, Hsinchu, Taiwan +
E-mail: dahlia@system.es.ncku.edu.tw

Abstract

In this paper, we have focused on the current I2O [1] (Intelligent I/O) architecture and have proposed an intelligent integration strategy for disk and LAN adapters. To describe the performance of this new architecture, a reasonable model is also given in this paper. The goal of our experiment is to set up an accelerating channel between disk and LAN interfaces. Our simulation result gives readers an expressively answer about how the Disk-to-LAN performance can be improved by the intelligent integration. With this technology, a multimedia server which needs high I/O bandwidth can achieve a better performance.

1. Introduction

In the age of networking multimedia, there are more and more loads on network and disk subsystems in personal computers. Although CPU's processing speed has been exponentially increasing in decades, certain applications, such as VoD (Video on Demand), still can exhaust the computing powers of PCs (personal computers). Under a such kind of exhausted situation, computing systems will easily get into unpredictable response and eventually out of control. Therefore we studied the problem "why two separate high speed (but dumb) disk and LAN I/O subsystems can not achieve the performance which they should do". LAN, in this paper, means a Ethernet LAN and Disk means a SCSI disk.

1.1 Related research

In reference [2], the data from Intel's study reveals that if only a single dumb (Non-I2O) Fast Ethernet is installed in a PC, the CPU is 50% utilized; if two are installed, the CPU is almost exhausted. If add the 3rd and 4th dumb Fast Ethernets into a PC, the total throughput of four Ethernets is almost the same as if only two of them are installed. This experiment result in Figure 1 demonstrates that " CPU is one of the bottleneck of network throughput" and it becomes useless to add the 3rd and 4th dumb Fast Ethernets into the same PC". In the following section, we use an example to demonstrate that the CPU is a bottleneck.

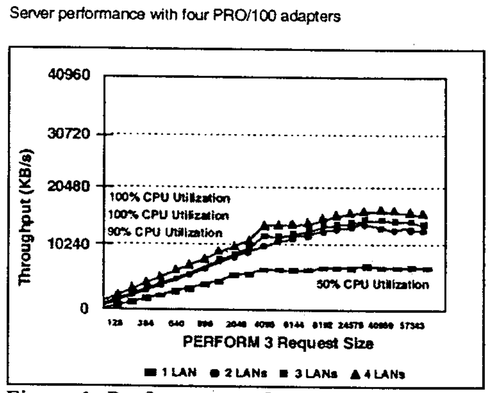

Figure 1: Performance of traditional network cards.

1.2 A simple experiment

Why the network performance of MS Windows is so poor ? For finding out key issues of the problem, we designed a simple experiment to evaluate the network throughput of Windows 98 and set up the experiment platform as following:
Pentium II 233MHz, 64MB RAM, Intel "82557" 10/100 Ethernet PCI, Intel Express 10/100 Ethernet Switch, Windows 98

We put server and client on the same local network without going through any router. When one PC transferred data to the other (including disk access), we found that TCP's throughput achieved only 23.42Mbps that was really amazing! After that, we used another more formal benchmark (HP's netperf [7]) to test. When data to be transferred were from memory but not from disk (i.e. without any disk access) and TCP/UDP size > = 4096Bytes, it achieved a better performance. Four test results are shown as following:

Table 1: Throughput of a dumb 82557 Ethernet card.

Throughput (Mbps)	48.46	48.94	48.76	48.79

During the evaluation, we observed that the CPU had the following typical behavior:

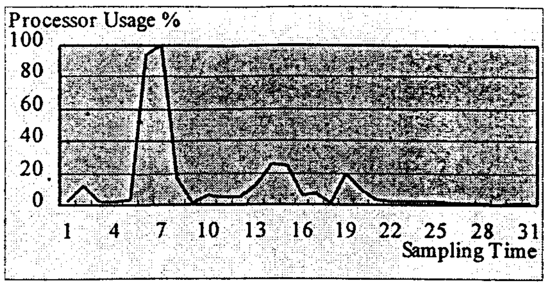

Figure 2: The usage of a host processor.

Figure 2 show that "under Windows98, the operating system and protocol stacks had already exhausted one server-level PC equipped with Pentium II 233, so the CPU had no capability to keep full the frame buffer on the Fast Ethernet card. This is also the key reason why the performance of the Fast Ethernet can not reach 100Mbps.

2. Disk-to-LAN acceleration

Reference [1] is the specification of the I2O architecture. The key idea of it is "using an I/O platform to deal with the tasks of I/O subsystems". In fact, the same idea was proposed in the age of mainframe computer. The main features proposed by the I2O groups include a "common communication layer" which provides a common message passing channel among every driver modules. The advantages of common communication layer include the high performance, platform independence, and ease of maintenance.

Our co-op research project [6] with ITRI CCL K400 was to develop a Disk-to-LAN acceleration card. In other words, one of its goals was to provide a high speed Disk-to-LAN channel. We believe that the I/O architecture must be reformed, because letting a general purpose CPU do all kinds of computing tasks is very inefficient. So, we needed a dedicated I/O processor and an I/O O.S. that form the I/O platform to deal with I/O tasks. We follow the I2O architecture to design our Disk-to-LAN acceleration card and use the i960rx (an Intel's RISC processor) and IxWorks (a WindRiver's RTOS, lack of file and network subsystems) as our I/O processor and I/O O.S. In reference [1], it is still an open issue "how to set up the acceleration channel between two I2O interfaces", so Disk-to-LAN acceleration is a worth topic of research.

3. Operational analysis

Under the traditional dumb architecture, as indicated in Figure 3a, the data flow clearly needs at least 6 memory copies to transfer a bulk of data from the sector buffer of a disk controller to the frame buffer of a LAN controller. Let's consider the I2O architecture, as solid line in Figure 3b. If we treat disk and LAN as two independent devices, then the later integration needs the same times of memory copies as the former dose. In addition, an I2O device uses message passing instead of function call to communicate among driver modules. Obviously, a message passing is much slower than a function call. Due to these two reasons, it seems no reason that I2O performs better than dumb I/O does.

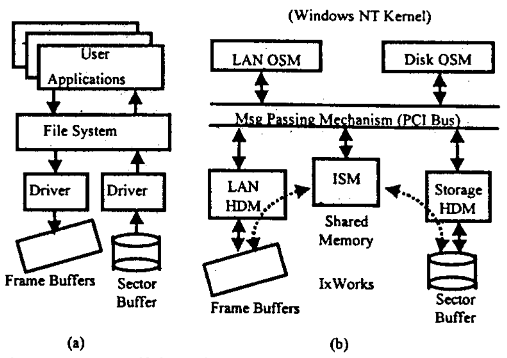

Figure 3: Traditional v.s. Intelligent Data flows

To resolve the performance uncertainty of the I2O architecture, we modified the data flow with dash line in Figure 3b and added an ISM module between two HDMs. Then, our intelligent integration strategy needs only two memory copies and the ISM module helps us improve greatly the Disk-to-LAN performance. Besides that, adopting a dedicated I/O processor, we could furthermore reduce the main CPU's overhead and resolve the I/O bottleneck caused by a CPU. The main tasks of the ISM are to allocate a shared memory and to set up a data channel between two HDMs.

Our next question was " How can the intelligent integration architecture improve the performance of Disk-to-LAN? " To answer this question, we had to look into the details of the disk and LAN operations and to investigate the relationship among host processor, I/O processor, LAN controller (Intel i82557), and disk controller.

First of all, we analyzed the operations of an I2O LAN adapter. To understand the performance issues of a NIC (network interface card), we must consider how a NIC communicate with a host computer. As we know, there are at least three kinds of computer I/O mechanisms, such as DMA, I/O processor and shared memory. These three mechanisms are not mutual exclusive, but they co-operate with each other under the I2O architecture. A typical I2O LAN interface operates as following:

Frames receiving

1. A LAN controller detects and receives the incoming frames from the connected network's physical layer then puts these frames into local memory.
2. After 1st step, a LAN controller triggers an interrupt to notice the I/O processor that there are some frames stored in local memory ready to be processed. The I/O processor then checks these frames and discards some of them whose header does not match the local MAC address.

3. The I/O processor programs the DMA controller after receiving enough amounts of frames. The programmed DMA controller (not I/O processor) is responsible for pushing those frames stored in the local memory into the host memory.
4. When the push is finished, the LAN controller interrupts the host processor for notification of new data to be processed.

Frames transmitting

1. The host processor prepares the transmitting information in host memory and executes the device driver to notify the I/O platform of new data to be transferred.
2. The I/O Processor then programs the DMA controller to pull the data from host memory to I/O processor's local memory.
3. Once complete, the DMA controller notifies the I/O processor of the new data to be transmitted.
4. The I/O Processor organizes these data into frames, and adds the header, the tail, and CRC information.
5. The I/O processor programs the LAN controller to begin the data transmission on physical layer. The LAN controller interrupts the I/O processor after every frame transmission.

Advantages of I2O LAN adapters

There are many advantages of an I2O LAN adapter, we mainly list three of them:

Reduced frequency of host processor's interrupt. When a traditional dumb I/O is used, a host processor has to context switch for every frame receiving interrupt. Context switch wastes lots of time and worsens the processor and bandwidth utilization, so the total system gets poor performance. If an I2O is used instead of a dumb I/O, a host processor will not be interrupted until I/O processor gathers certain amounts of incoming frames. In other words, each time a frame arrives or is transmitted, a LAN controller generates an interrupt to I/O processor but not directly to host processor. Therefore host processor save much time in context switch.

Less frame loss in LAN controller. An I2O platform is special designed for high speed I/O, so it achieves a lower interrupt latency comparing to a dumb I/O and has enough time to process huge amounts of arriving frames. This is the reason that I2O has less frame loss in LAN controller.

Light overhead on host processor. Not the host processor but the I/O processor is responsible to do framing tasks, so there is a much lighter overhead on host processor. On the other hand, the data transferred between host memory and local memory does not include the framing overhead such as header, tail, and CRC, so the bandwidth utilization of data bus between host memory and local memory is more efficient.

4. Read and write of a SCSI disk

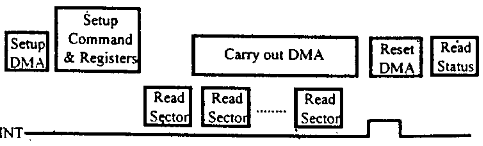

Figure 4: SCSI's DMA read command.

What we want to do is to improve the Disk-to-LAN's performance. Not only I2O Fast Ethernets' operations, but also the SCSI disks' operations are necessary to be investigated.

For increasing the read and write throughput of a SCSI disk [4], we should make use of the bulk transfer mechanism, i.e. not to interrupt a processor for every sector transferring completion. For example, if a reading sector buffer is 64 KB (about a track size), we should read 64KB for each time instead of 512 Bytes (a sector size). Thus a DMA controller does not generate an interrupt for every sector until bulk transfer complete, as depicted in Figure 4. The bulk transfer results in less disk interrupts and is good for promoting the disk throughput. Due to less disk interrupts, other tasks can get more CPU time and whole system becomes more efficient. Most of the contemporary design of disks has the bulk transfer mechanism.

Advantages of the SCSI disk architecture

The read and write operations of I2O disks also base on the cooperation among DMA controller, host processor, I/O processor and shared memory. Therefore, they have the same advantages as I2O LANs do, as the follows.

- Reduced frequency of host processor's interrupt.
- Light overhead on host processor.
- Much higher disk throughput.

5. The integration strategy of LAN-to-Disk (shared memory)

Shared memory is very useful for a large amount and real time data transfer. It is a suitable high performance data channel between two I/O devices due to the following reasons:

- Only two memory copies are needed comparing to the original six copies (Figure 3b).
- Data copies do not interfere with the host processor but an I/O processor. So other tasks can be performed by a host processor concurrently as some data are transferred via shared memory channel. Finally, concurrent executions improve the whole system's efficiency.

6. Theoretical analysis of network throughput

It is easy to show the key factors that affect the network throughput very much. The model depicted in Figure 5 is too simple and does not take the memory copy time and interrupt processing time into consideration. However, we still can clearly understand the relationship between inter-frame space and maximum throughput from this simplified model, as shown in Figure 6.

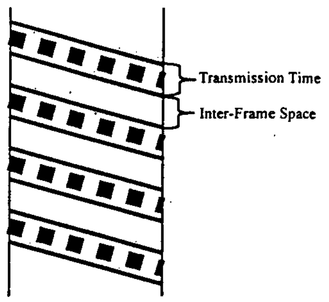

Figure 5: Performance of frame transmission.

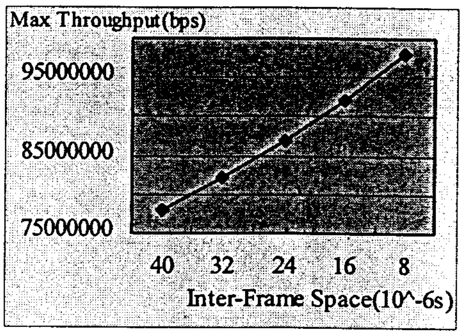

Figure 6: The relationship between inter-frame space and maximum throughput. (As R_{TX}=100Mbps)

In our one-year project, we consider only the Disk-to-LAN ability. We assume that the data placement in hard disks is in certain pre-arranged order, so the disk is a data producer and Ethernet
adapter is a consumer and also assume the data produced by disk always keep the network frame buffer full. Therefore, the inter-frame space can keep as minimum as possible.

Now what we concern is how and how much the Disk-to-LAN performance can be improved based on the design of shared memory and I2O architecture. In other words, we want to know how much this intelligent integration strategy can raise the Disk-to-LAN throughput.

Under our integration strategy, we have known that the times of memory copy are reduced from 6 to 2, and there is a dedicated RISC I/O processor to deal with all the driver level tasks. So, in the aspects of memory copies and interrupt processing, our strategy can save lots of time.

For describing and simulating a Disk-to-LAN integration card, we have to consider the effect caused by disk and LAN's co-operation. In Figure 7, we want to remind readers that a disk controller, a LAN controller, a host processor and an I/O processor can run concurrently, so their work time overlap together. According to most real cases, we make two assumptions:

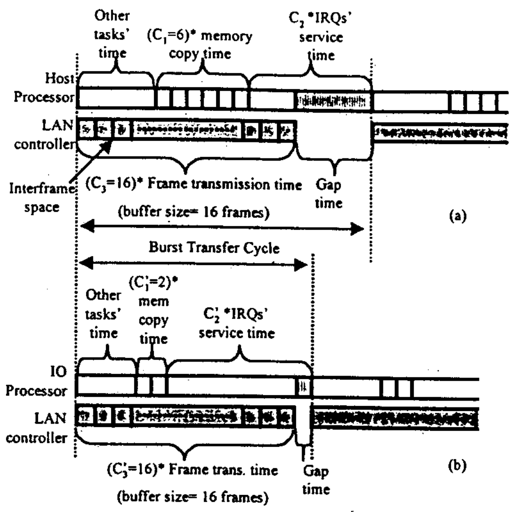

Figure 7: Performance model of
(a) two independent-dumb disk and LAN adapters.
(b) an intelligent Disk-to-LAN integration card.

- A SCSI's sustained reading throughput is larger than LAN's transmission rate.
- Due to the time overlapping, we can ignore the sector reading time as discussing performance model in Figure 7.

Figure 7a and 7b show the relationship between dumb and intelligent integration models. Our host processor is Intel's Pentium 133 running on Windows NT, and I/O processor is i960RX running on IxWorks. The memory copy time in Figure 7 depends on the DMA transfer rate not on processor' copy speed.

From model we found out four entry points to increase the Disk-to –LAN's throughput:

- Raise the Ethernet transmission rate.
- Low down the gap time (refer Figure 7b).
- Reduce the times of IRQ and IRQ's processing time.
- Decrease the times of memory copies.

In fact, some readers might want to reduce the time of other task's time (including context switch). But in our point of view, we have targeted our host platform on Windows NT series' operating system. Under this situation, what we can do is "No running too many applications simultaneously". We do not want to involve in the NT's black box, so we make no change in Windows NT, except that OSM module got from Microsoft.

Only considering different kinds of 100Mbps Fast Ethernet adapters, the gap time have determined an adapter's network utilization and throughput. Our intelligent integration strategy adopts I2O architecture and uses shared memory mechanism to reduce the times of IRQ and IRQ's processing time, to decrease the times of memory copies, and finally to low down the gap time (Figure 7b). Consequently, we are on the way to improve the Disk-to-LAN's network utilization and throughput.

In Table 2, we arranged and measured many reference values of inter-frame space, and defined our variables in Table 3. From the model of Figure 7, we can derive the relationship among variables into equations. From Equations 1 to 4, readers can easily calculate the values of gap time, utilization, deltaGap and prove the performance increase of a Disk-to-LAN integration strategy.

Table 2: Inter-frame space.

Ethernet Type	10M	10M	100M	100M	Giga
T_{IFS} (10^{-6}sec)	(min) 9.6	(dumb) 365	(min) 0.96	(I2O) 8	(min) 0.096

Table 3: Variable definitions.

T_{OT}	Average Time of Other Tasks
T_{IRQ}	Average Time of IRQ service
T_{IFS}	Min. Inter-Frame Space Time
B_{FS}	Bytes of max Frame Size
$B_{DMA\ Unit}$	Bytes per "DMA memory copy Unit"
R_{TX}	Rate of Transmission
R_{DMA}	Rate of DMA memory copy
C_1	Number of memory copies in a burst cycle
C_2	Number of IRQ services in a burst cycle
C_3	Number of "frames buffer units"

$$Gap = \left[T_{ot} + C_1 * \frac{B_{DMA\_Unit}}{R_{DMA}} + C_2 * T_{IRQ}\right] - \left[C_3 * \frac{B_{FS}}{R_{TX}} + (C_3 - 1) * T_{IFS}\right] \quad ....(eq.1)$$

$$Utilization = \frac{C_3 * \frac{B_{FS}}{R_{TX}}}{Gap + (C_3 - 1) * T_{IFS} + C_3 * \frac{B_{FS}}{R_{TX}}} \quad (eq.2)$$

$$Gap = \frac{C_3 * \frac{B_{FS}}{R_{TX}}}{Utilization} - \left[(C_3 - 1) * T_{IFS} + C_3 * \frac{B_{FS}}{R_{TX}}\right] (eq.2a)$$

$$deltaGap = delta\ mem\_copy\_time + delta\ IRQ\_service\_time \quad ....(eq.3)$$

or

$$deltaGap \approx (C_1 - C_1')(\frac{B_{DMA\_Unit}}{R_{DMA}}) + (C_2 - C_2')T_{IRQ} \quad ....(eq.3a)$$

$$Ideal\ Utilization = \frac{\frac{B_{FS}}{R_{TX}}}{T_{IFS} + \frac{B_{FS}}{R_{TX}}} \quad (as\ Gap = T_{IFS})(eq.4)$$

7. Simulation results

From a long-term point of view, if assuming that the frame buffer is always filled with the data from disk and ignoring the probability of frame collision on an Ethernet switch, the gap time will decrease and approach to the inter-frame space theoretically. In such a situation, the utilization of network bandwidth will have nothing to do with variable C_3, as shown in Equation 4.

But in practical situation, the gap time got from real case is much larger than an inter-frame space. For example, according to the data from reference [2], we know that the average inter-frame time of an Intel's I2O Ethernet card is 8 ms (microsecond). Based on the fact, we derived that the gap time of this card in our model is $8*C_3$-$0.96*(C_3$-1) ms. Also, we traced the driver code of this card and found out that the frame buffer at most can be filled with 16 frames. So, by definition C_3=16 and the gap time will be $8*C_3$-$0.96*(C_3$-1)=113.6 ms.

Tables 4: Reference values of variables.

B_{FS}	1518Bytes
R_{TX}	100Mbps
T_{IFS}	0.96*10^-6 sec.
Gap	113.6*10^-6 sec.
C_3	Number of "frames buffer units"

Table 4 records some useful values of defined variables with that we can show readers the relationship between frame buffer size and network bandwidth utilization. Input those values into Equation 2 and the result is Figure 8a.

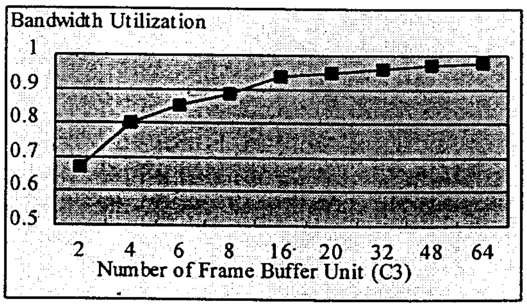

Figure 8a: Relationship between buffer size and utilization. (As gap time =113.6 ms)

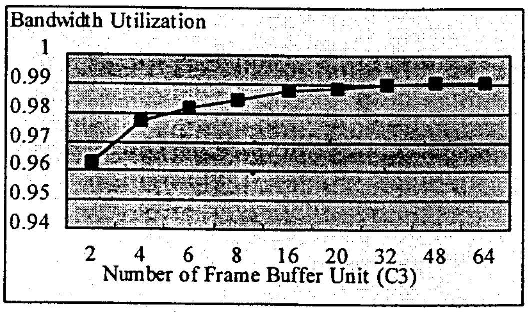

Figure 8b: Relationship between buffer size and utilization. (As gap time =8 ms)

After doing many trials (Figure 8b is a result of one trial) to find out the relationship between buffer size and utilization, we made a conclusion—when C_3 is larger or equal to 16, the bandwidth utilization approaches the saturate state (or optimal state).

Due to the shared memory channel in the Disk-to-LAN integration card, we save four memory-copy time and C_1:C_1'=6:2. On the other hand, we assume the pre-fetch (DMA) buffer of a SCSI disk can store at most one track data which is a multiple of 16 frames' size, such that using the DMA read command, the disk controller interrupts the I/O processor after every DMA complete.

From previous discussion of "**Frames transmitting**", we know that the LAN controller interrupts the I/O processor after every frame transmission. Therefore, the interrupt times' ratio of disk controller to LAN controller is the inverse of (64KB/(1518B-protocol overhead)) and is about 0.021. We only take the interrupts from disk and LAN into consideration, so C_2 is the total interrupts caused by the disk and LAN in a burst transfer cycle. C_3 is the max number of frames stored in a frame buffer during a burst transfer cycle", also C_3 is the number of network interrupts during a burst transfer cycle. We traced the driver's source code of Intel's 82557 network card, and found that the frame buffer of 82557 can store 16 frames at most. So, C_3=16 and C_3'=16(1+0.021)=16.336.

7.1 Performance improvement

As described in section "**A simple experiment**", two dumb and independent disk and LAN cards co-operate together, their network bandwidth utilization only achieve 23.42%. Using this utilization, we can derive the gap time=0.00604473 second from Equation 2a, where the transmission rate is 100Mbps, max frame size is 1518 bytes, and other parameters are listed in Table 5.

From reference [2], we know that the average inter-frame time of an Intel's I2O Ethernet card is 8 ms. Based on it, we have derived that the gap time of I2O 82557 card in our model is 8*C_3-0.96*(C_3-1) =113.6 ms.. Furthermore, according to the Equation 3a, we can estimate the gap time of our integration card from the gap time of the I2O 82557 card, and we estimated it by Equations 5b. The gap time's relationship can be shown as Figure 9.

$$Gap_{Disk\text{-}to\text{-}LAN} = GapI2O-82557 + deltaGap \quad (eq.5)$$

$$Gap_{Disk\text{-}to\text{-}LAN} = C_3 * 8us - (C_3 - 1)T_{IFS} + deltaGap \quad \ldots\ldots\ldots\ldots\ldots\ldots(eq.5a)$$

$$Gap_{Disk\text{-}to\text{-}LAN} = C_3 * 8us - (C_3 - 1)T_{IFS} \quad \ldots\ldots(eq.5b)$$
$$+ (C_1 - C_1')(B_{DMA\_Unit} / R_{DMA}) + (C_2 - C_2') * T_{IRQ}$$

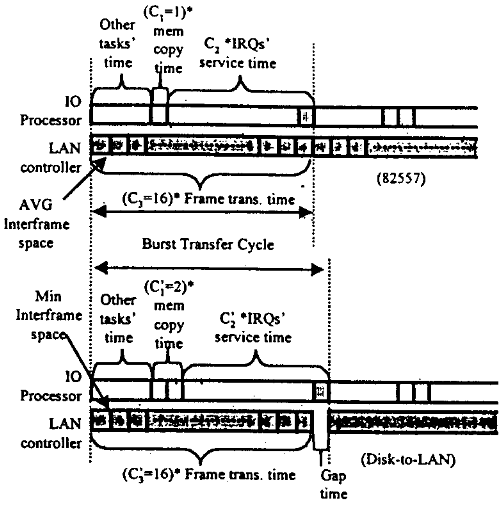

Figure 9: The relationship between 82557 and Disk-to-LAN cards.

One DMA disk read can support at most three burst transfer cycles, so one memory copy time in a burst transfer cycle is about 1/3 of a DMA disk read time. Therefore, we assume that C_1 is 1/3 of a DMA disk read time and that C_1' is 2/3 of a DMA disk read time. Due to (C_2-C_2')*(T_{IRQ}) << (C_1-C_1')*(B_{DMA_Unit}/R_{DMA}), we ignore (C_2-C_2')*(T_{IRQ}).

Table 5: Reference values of variables.

$B_{DMA\ Unit}$	64Kbytes
R_{DMA}	35MBps
T_{IFS}	0.96*10^-6 sec.
Average T_{IFS}	8*10^-6 sec. (8us)
C_3	16
C_3'	16.336
C_1	1/3
C_1'	2/3

From Equation 5b and parameters in Table 5, we found that the gap time of Disk-to-LAN integration card is about 0.000709 sec. Again, input the gap time into Equation 2, we derive that the network bandwidth utilization of the integration card is 0.719. Comparing the result to the result obtained from the previous simple experiment, we conclude that the network utilization can be improved from 0.2342(TCP layer's utilization, when disk and LAN cards are dumb and independent) to 0.719 (MAC layer's utilization, when our Disk-to-LAN integration strategy is used).

8. Conclusions

In this paper, we have proposed a network throughput model for Disk-to-LAN acceleration. Using this model, we can analyze the performance of any kind of disk and LAN integration and find out the key factors that reduce the performance.

We used the reference data to put into those equations we derived, and successfully showed the relationship between frame buffer size and bandwidth utilization. We found that when frame buffer size = 20* 1518 Bytes, the bandwidth utilization will approach its maximum amount.

Finally, we found that the gap time of the Disk-to-LAN acceleration is 0.000709sec and then the bandwidth utilization of the Disk-to-LAN acceleration can be derived as 0.719. So, we conclude that our Disk-to-LAN acceleration strategy has raised the network bandwidth utilization from 0.2342 (as that in the simple experiment) to 0.719.

In this paper, we did not consider the case of LAN-to-Disk acceleration yet, owing to that there are too many issues of what the best data placement on disks is. Theoretically, different data placement policies will result in different throughput levels. LAN-to-Disk acceleration is a good future work, one possible assumption is that if the data placement is continuous so the disk controller can write data to disks as soon as possible and achieve the maximum LAN-to-Disk throughput.

9. References

[1]. Intelligent I/O Specification version 1.5 。
[2]. Byron Gillespie, "PCI Intelligent I/O Design for High Performance Servers", Intel Corporation
[3]. M. Morris Mano, "Computer System Architecture, 3rd Edition", p402-429
[4]. Friedhelm Schmidt, "The SCSI Bus and IDE Interface", Addsion-Wesley, p55-58, p161-163
[5]. Gilbert Held, "LAN Performance Issues and Answers 2nd Edition ", p228-p230
[6]. Liou Shiang-Chun, Huang Yueh-Min, "The Mid-term Report of I2O Study Project" , The technical report of CCL, Feb. 1998
[7]. http://www.cup.hp.com/netperf/NetperfPage.html.