The Journal of Supercomputing, 26, 59–75, 2003

Firmware-Level Latency Analysis on a Gigabit Network

HYUN-WOOK JIN hwjin@os.korea.ac.kr

Department of Computer Science and Engineering, Korea University, 1, 5Ka, Anam-Dong, Sungbuk-Ku, Seoul, 136-701 Korea

CHUCK YOO hxy@os.korea.ac.kr

Department of Computer Science and Engineering, Korea University, 1,5Ka, Anam-Dong, Sungbuk-Ku, Seoul, 136-701 Korea

JIN-YOUNG CHOI choi@formal.korea.ac.kr

Department of Computer Science and Engineering, Korea University, 1, 5Ka, Anam-Dong, Sungbuk-Ku, Seoul, 136-701 Korea

Abstract. Gigabit networks are equipped with "increasingly" intelligent network interface cards, and the firmware running in the cards does various tasks related to end-to-end communication. For an accurate performance evaluation of gigabit networks, it is very important to characterize and quantify the firmware. However, the firmware has been neglected in the latency analyzes of network protocols.

This paper presents an in-depth latency analysis of Myrinet. Our findings include that the major bottleneck is the network interface card itself. This is true especially for so-called lightweight user-level protocols (such as BPI of Myrinet) designed for high-speed communication. Although BPI is very lean and efficient in the host, its sending throughput becomes similar to UDP. This result is very unexpected and surprising. Through firmware-level measurements, we identify that the cause of bottleneck is the DMA performance.

Keywords: latency analysis, gigabit network, Myrinet, cluster, Asynchronous UDP, firmware

1. Introduction

With the advances in high-speed networks (e.g., Myrinet [1] and Gigabit Ethernet [2]), there are many efforts to enhance the performance of network protocols in order to fully utilize the bandwidth and apply those enhanced protocols to clusters. A characteristic of gigabit networks is that the network interface card (NIC) takes a more active role than in "slow" networks. NIC has been a simple device that has minimal memory and just enough intelligence to transmit packets out to the wire, but now the trend is that NIC is getting smarter with more memory and more processing power. The rationale is to: (1) off-load CPU; and (2) keep up with the fast transmission.

In addition to its improved processing capability, NIC of gigabit networks has an intelligent firmware running inside. The firmware does much more tasks including packet processing, interrupt handling, etc. than before. The device driver in the host

interacts with the firmware to send or receive packets. Therefore, for an accurate performance evaluation of gigabit networks, it is very important to understand the impact of NIC.

The firmware of NIC, however, has been neglected in the performance analyses of network protocols. Such analyses give only partial evaluation of gigabit networks. For example, Myrinet NICs of Myricom, ACEnic of Alteon Networks, and G-NIC II of Packet Engines contain an on-board co-processor and sophisticated firmware, and they perform important protocol steps related to end-to-end communication. Therefore, we believe, the NIC firmware of gigabit networks should be considered as a layer of the protocol stack.

The goal of this paper is to present a firmware-level latency analysis that covers the firmware itself of NIC as well as the interaction between the host and firmware. Our analysis is more advanced because most existing performance measurements are performed at the device driver level [3–5]. We also evaluate three factors (shown in Section 5) that influence the network performance through the firmware-level latency analysis. For the first step of our analysis, this paper focuses on the send side only.

The rest of this paper is organized as follows. Section 2 provides a background that outlines the protocols analyzed in this paper. We use UDP, an optimized protocol called Asynchronous UDP [6], BPI [7] that is a lightweight user-level protocol for Myrinet, and enhanced BPI. Section 3 briefly describes the measurement methodology. Section 4 shows the latency measurement results. From the firmware-level measurements, we identify three factors that limit the network performance in Section 5. Finally, the paper concludes with Section 6.

2. Background

The efforts to take advantage of gigabit networks on clusters can be classified into two approaches. The first approach is the optimization of a traditional kernel-level protocol [6, 8–13]. The second is the development of a new lightweight user-level protocol [14–21]. In this paper, we attempt to analyze both kernel-level and user-level protocols in the firmware-level.

We choose UDP as a kernel-level protocol because UDP is far lighter than TCP, and multimedia protocols like RTP (real time protocol) [22] are based on UDP. To compare with UDP, we analyze Asynchronous UDP, which is lighter and more optimized than UDP. As user-level protocols, BPI of Myrinet Software [7] and enhanced BPI are analyzed. This section explains Asynchronous UDP and BPI briefly. In addition, the firmware of Myrinet NIC is described.

2.1. Asynchronous UDP

Asynchronous UDP is a new optimization of UDP proposed in Yoo et al. [6]. It is designed to fully utilize the bandwidth of high-speed networks. The experiment results show that it can utilize 98% of ATM's bandwidth and improves the throughput by 133% over UDP. In addition, Asynchronous UDP saves processor

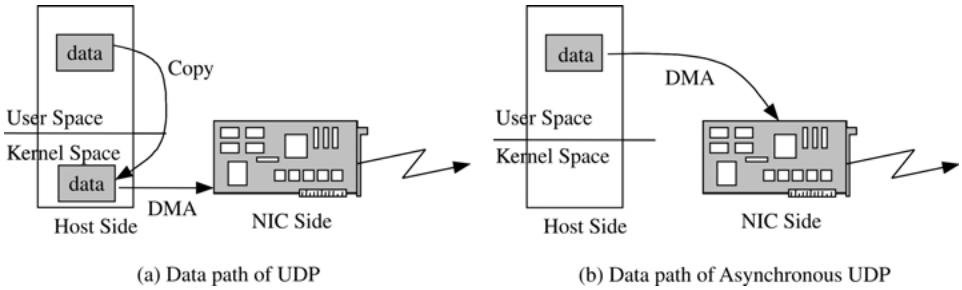

Figure 1. Comparison of data path.

cycles, and applications can send more UDP data and/or do something else while data are being sent because communication is done in parallel with computation.

The key idea of Asynchronous UDP is to remove the copy operation in the protocol processing. The data in user space is directly moved to NIC as shown in Figure 1. We refer the details of Asynchronous UDP to Yoo et al. [6] and Jin et al. [13].

In comparison with previous approaches to remove the copy overhead [8–12], Asynchronous UDP does not require any hardware support and does not cause copy-on-write. For example, Copy Emulation [11] is similar to Asynchronous UDP, especially in the point that it keeps the pointer of buffers to avoid data copy and that it does fragmentation and header building based on this pointer. However, Copy Emulation can cause copy-on-write as a penalty of removing the copy overhead. Furthermore, Copy Emulation is highly dependent on virtual memory system, but Asynchronous UDP does not depend on virtual memory system at all.

2.2. BPI

BPI allows an application to communicate over the network directly from user space. BPI offers flexible interface to an application. For instance, BPI translates the virtual address of the data buffer into the physical address for DMA between the host and NIC memories. For the purpose of the address translation, BPI allocates a copy block in the kernel memory and mmaps (memory maps) the area into the user space when BPI is initialized. The copy block is a physically linear area of the host memory, and BPI knows the physical base address of the copy block. Before sending, BPI copies the data into the copy block and calculates the physical address by the offset from the base address. Figure 2 describes the data path of BPI send. BPI uses DMA to/from the copy block without kernel intervention. U-Net [14] also has a similar user interface.

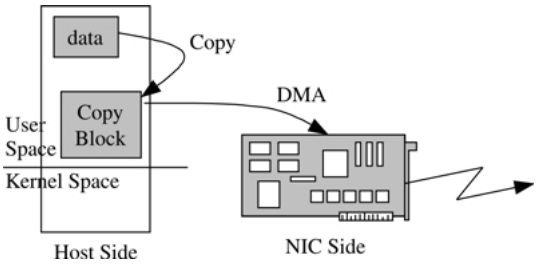

Figure 2. Data path of BPI.

2.3. *MCP (Myrinet Control Program)*

MCP is the firmware of Myrinet NIC, which is divided into two layers for sending: hostTx and netTx. The hostTx layer covers the data movement between the host and NIC sides using DMA (EBUS-LBUS DMA). Figure 3 depicts how EBUS-LBUS DMA is performed on a system based on an Intel Pentium Pro processor. First, MCP initiates a DMA request to the host/PCI bridge (Figure 3(1)). Then, the bridge checks whether the contents of the memory area for DMA exist in the processor's internal cache (Figure 3(2)). If the contents are cached, they should be flushed for the cache consistency (Figure 3(3)). After the cache consistency is guaranteed, the EBUS-LBUS DMA is carried out (Figure 3(4)). The netTx layer is responsible for the send-DMA (Figure 3(5)) that moves the data to the Myrinet LAN.

Myrinet NIC has a separate DMA engine for each of EBUS-LBUS DMA and send-DMA. MCP is a well-designed firmware to utilize which feature for performing EBUS-LBUS DMA and send-DMA in parallel [23].

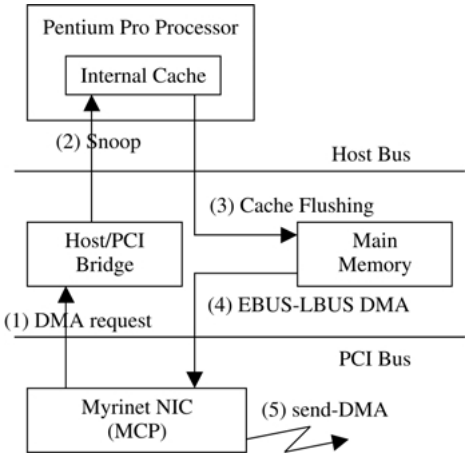

Figure 3. DMA path on an Intel Pentium Pro processor system.

3. Latency analysis methodology

A typical methodology to measure the latency is the kernel instrumentation (e.g., `do_gettimeofday()` kernel function or `rdtsc` assembly instruction). There have been many latency analyses based on the kernel instrumentation. One example, to name a few, is the detailed latency analysis of UDP on top of FDDI by Kay and Pasquale [4]. Another latency analysis is done with TCP on ATM [5]. It uses a high-resolution clock on TurboChannel card for the measurement. It analyzes the latency only at the device driver level as in Kay and Pasquale [4]. Elbert et al. [24] showed the latency of TCP over Gigabit Ethernet.

A weakness of the kernel instrumentation is that the measurements are restricted to the host side latency, and it cannot measure the latency that the NIC firmware induces. That is, the measurement method based on the kernel instrumentation alone cannot analyze the full spectrum of the latency in gigabit networks accurately.

As an ad-hoc solution, a logic analyzer was used to measure the latency of NIC firmware, but it requires a special hardware board to connect the logic analyzer to the I/O bus. Using a logic analyzer is very cumbersome and needs hardware expertise. It is very clear that a systematic approach is needed, and the need is indispensable for gigabit networks.

We proposed a new methodology in Jin and Yoo [25]. The key idea is to use the NIC clock for measurements. A desirable resolution for latency analysis is sub-microsecond, and we found that most NIC clocks, if not all, have enough resolution. Our methodology is outlined below, and see Jin and Yoo [25] for the details.

1. The NIC clock is mmapped into the host address space as Figure 4 shows. Now, the reading of the mmapped clock from the host side simply becomes a matter of referencing a pointer. One thing to be careful is that the clock should be mmapped into the uncacheable area. If the clock is mmapped into the cacheable area, the latest clock value is not read.
2. The measurement code is inserted into the kernel and NIC firmware. By reading the NIC clock through the measurement code, time stamps are generated from both host and NIC sides (see Figure 4). In this manner, the latency in host and NIC is measured with the same clock.

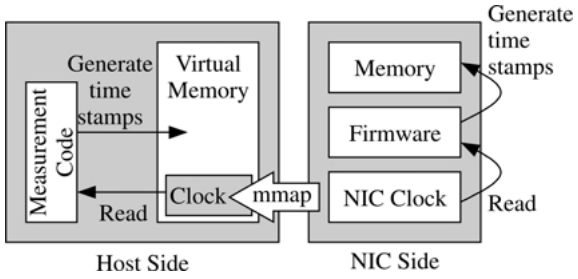

Figure 4. Measurement methodology based on NIC clock.

Since our methodology uses mmap, the measurement overhead is in the order of memory access and thus suitable for gigabit networks without additional hardware equipment like a logic analyzer.

4. Measurement results

All of our experiments run on a pair of the workstations that use an Intel 180 MHz Pentium-Pro processor, Intel 440FX PCI chipset, and M2F-PCI32C Myrinet NIC (LANai4.3) connected to a 32bit 33 MHz PCI slot. The Myrinet NICs are connected directly with M2F-CB-25 Myrinet LAN cable. The operating system used is Linux (kernel version 2.0.27).

4.1. UDP

Figure 5 shows the protocol stack of UDP, which is divided into the host and NIC sides. An application requests a send operation via the kernel through a system call. UDP/IP copies the data from the user space into the kernel space with the checksum computation and constructs its header (U/I of Figure 5). After that, the data are passed to the device driver, which attaches an Ethernet header (E of Figure 5) and inserts the send request into the send queue that is shared with MCP. When the hostTx of MCP detects the request, the data of kernel buffer are EBUS-LBUS DMA'ed into the NIC memory. Then, the hostTx constructs a Myrinet header (M of Figure 5). The netTx detects the data in the NIC memory and performs the send-DMA.

Figure 6 graphs the latencies of host and NIC sides. The increase of host side latency is due to the per-byte overheads such as copy and checksum overheads, and the EBUS-LBUS DMA overhead is the main contribution to hostTx latency as

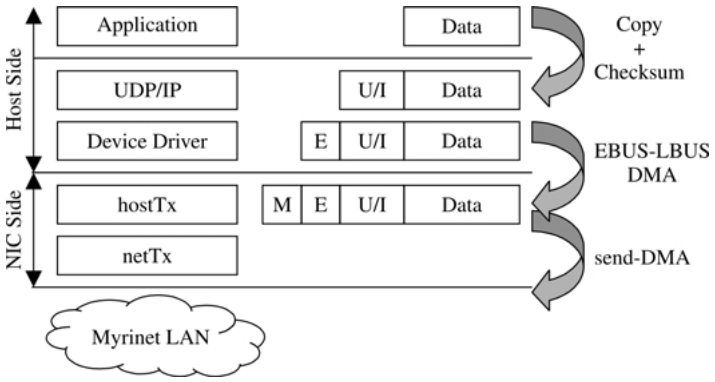

Figure 5. Protocol stack of UDP.

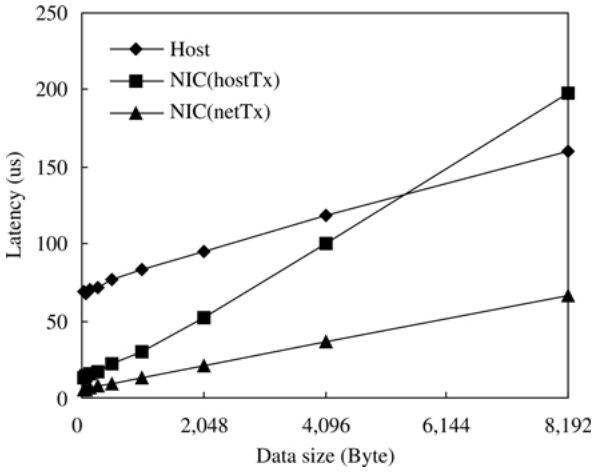

Figure 6. Latency of each side sending an UDP packet.

shown in Table 1. The send-DMA overhead dominating the netTx latency is totally decided by the network physical media.

In Figure 6, we can observe that the latency of host side and the hostTx latency cross at a data size near 5 KB, and the latency of hostTx becomes larger than the others. This means that, for data sizes larger than 5 KB, the NIC side dominantly decides the network performance. The processor on the host is getting so much faster that the crossover point of host and hostTx overheads moves to a significantly smaller data size. For example, the crossover point on an Intel Pentium III 1 GHz processor platform is about 1.5 KB. As a result, the NIC overhead is a critical factor of network performance for most data sizes, especially considering the packet size tends to become large in order to achieve a better network utilization. The details are discussed in Section 5.

Table 1. Per-byte overhead of UDP and Asynchronous UDP

	UDP		Asynchronous UDP
Data size	Copy and checksum	EBUS-LBUS DMA	EBUS-LBUS DMA
32	2.2	13.5	21.1
64	2.2	13.8	21.7
128	2.7	15.8	22.2
256	3.6	17.3	23.7
512	5.0	21.5	27.2
1,024	7.9	29.8	38.2
2,048	14.6	52.5	61.1
4,096	27.8	100.4	110.5
8,192	55.6	198.3	212.1

4.2. *Asynchronous UDP*

The EBUS-LBUS DMA overhead of Asynchronous UDP is also shown in Table 1. The EBUS-LBUS DMA overhead of Asynchronous UDP is slightly larger than that of UDP. The reason is that Asynchronous UDP needs more gathers in order to perform DMA for not only the data in user space but also the header in the kernel space.

Asynchronous UDP does not have the copy overhead, and the DMA operations of hostTx and netTx are the only per-byte overheads. Consequently, in Figure 7, the latency of host side is almost invariant unlike UDP. On the other hand, the NIC side latencies increase as the data size gets larger. Note that the intersection of host and hostTx latencies occurs at the data point adjacent to 3 KB, which is smaller than the crossover point of UDP. In the case of an Intel Pentium III 1 GHz processor platform, those latencies cross at 512 B.

4.3. *BPI*

Figure 8 depicts the protocol stack of BPI. In the host side, BPI copies the data to the copy block and inserts the send request into the send queue. Then, hostTx and netTx perform EBUS-LBUS DMA and send-DMA respectively.

The measurement results of each side are shown in Figure 9. The latencies of host and hostTx increase in proportion to the data size, which are caused by the copy and EBUS-LBUS DMA overheads as Table 2 shows. As mentioned in Section 4.1, the latency of netTx depends on the physical network bandwidth.

An interesting result in Figure 9 is that the latency of host side is larger than the NIC side for data sizes larger than 128 B. That is, in the case of BPI, the host side is the bottleneck rather than the NIC side. We can find the reason from the great

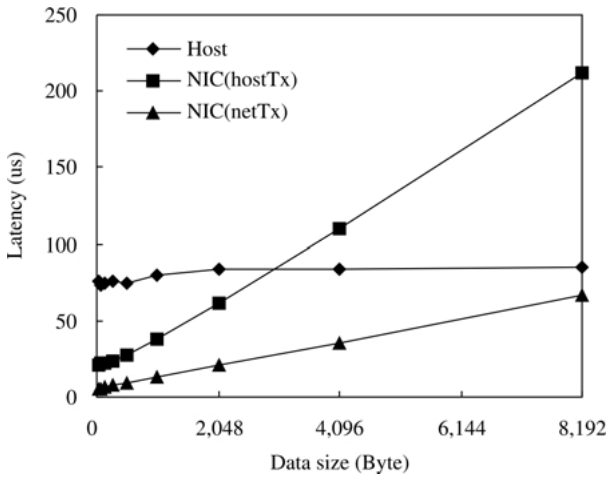

Figure 7. Latency of each side sending an Asynchronous UDP packet.

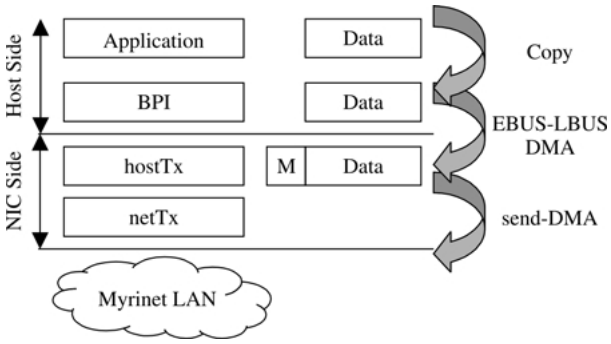

Figure 8. Protocol stack of BPI.

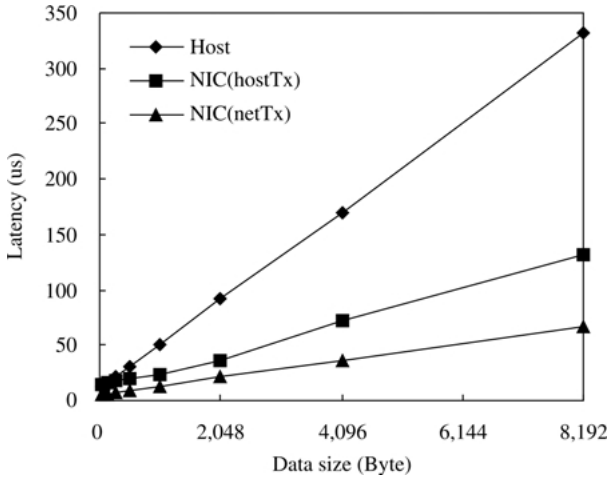

Figure 9. Latency of each side sending a BPI packet.

disparity between the copy overhead of UDP in Table 1 and that of BPI in Table 2. This surprising difference is due to the cacheability of the BPI's copy block. Because the copy block is set as an uncacheable area in the initialization of BPI, the copy from the user buffer to the copy block is done uncached. This turns out to have a large copy overhead. The details are discussed in Section 5.

4.4. Enhanced BPI

We improve BPI by changing the cacheability of the copy block. We call it enhanced BPI. Table 2 and Figure 10 show the latency of the enhanced BPI. As shown in Table 2, the copy overhead of the enhanced BPI takes much less than BPI. This is because the enhanced BPI copies the data only to the processor's internal cache.

Table 2. Per-byte overhead of BPI and enhanced BPI

	BPI		Enhanced BPI	
Data Size	Copy	EBUS-LBUS DMA	Copy	EBUS-LBUS DMA
32	2.3	14.6	1.8	14.8
64	3.9	15.0	2.0	15.1
128	6.7	16.8	2.1	17.0
256	12.6	17.3	3.2	17.6
512	24.2	19.4	4.4	22.3
1,024	48.1	24.1	6.7	30.5
2,048	96.0	36.6	12.7	57.0
4,096	191.7	71.6	23.0	100.8
8,192	384.7	131.5	48.1	192.8

In contrast, the EBUS-LBUS DMA overhead of the enhanced BPI is slightly larger than that of BPI as shown in Table 2. The difference of the EBUS-LBUS DMA overhead is caused by the write-back cache policy for the cache consistency. There is no implicit write-back overhead when BPI is used because the copy block is uncached. However, in the case of the enhanced BPI, implicit write-back occurs by the snoop protocol when hostTx tries to EBUS-LBUS DMA the data into the NIC memory because the data have been cached during the copy operation. Therefore, the DMA overhead of enhanced BPI is a little larger than that of BPI.

5. Three factors: DMA overhead, copy operation, and buffer cacheability

This section evaluates three important factors of network performance, which are elucidated through our firmware-level measurements.

5.1. DMA overhead of NIC

Since NIC has its own processor and memory, it runs in parallel with the host. It means that, if the latency of the host side is ideally overlapped with NIC's latency, there are three possible cases depending on the length of the host side and NIC side latency, shown as follows and in Figure 11.

Case (a): the latencies of the host and hostTx are the exactly same (Figure 11(a)).

Case (b): the latency of host side is larger than that of hostTx (Figure 11(b)).

Case (c): the latency of host side is smaller than that of hostTx (Figure 11(c)).

A horizontal bar of Figure 11 represents the total latency for a request (until the request is sent). Note that the hostTx and netTx latencies can overlap because the Myrinet NIC provides multiple DMA engines, which is indispensable to NIC for achieving the full physical network bandwidth [23]. In addition, the latency of netTx is smaller than that of hostTx, so that the latency of netTx is hidden by that of hostTx. Therefore, the latencies of the host and hostTx are important.

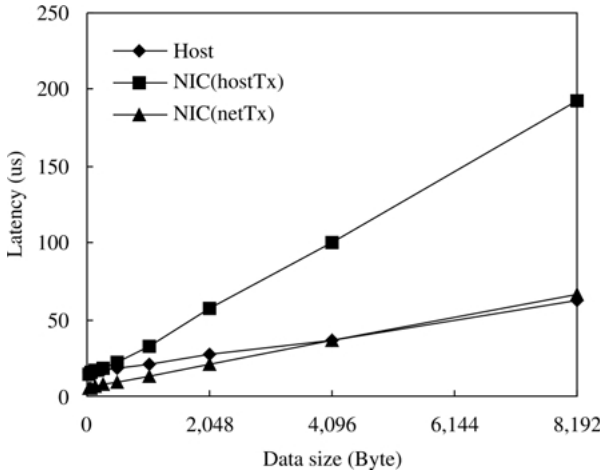

Figure 10. Latency of each side sending an enhanced BPI packet.

Therefore, the throughput is determined by the dominant latency (i.e., latency_3) as follows:

$$\begin{aligned}\text{Throughput} &= \lim_{n\to\infty} \frac{\text{data\_size} \cdot n}{\text{latency}_3 \cdot n + \sum_{i=1}^{2} \text{latency}_i} \\ &\quad \text{for } \text{latency}_i,\ i = 1,\ 2,\ 3,\ \text{s.t. } \text{latency}_j > \text{latency}_k, \quad j > k \\ &= \frac{\text{data\_size}}{\text{latency}_3}\end{aligned}$$

where n is the number of packets sent. In other words, in case (c), the throughput is determined by the hostTx latency, which implies the NIC side is a bottleneck. The opposite is true for case (b).

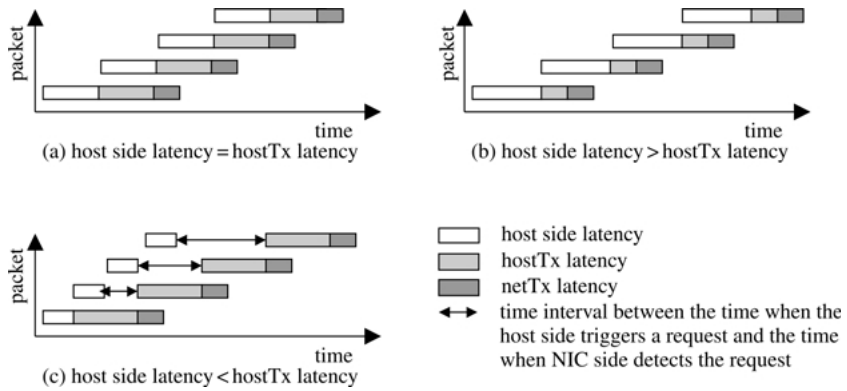

Figure 11. Latency overlapping.

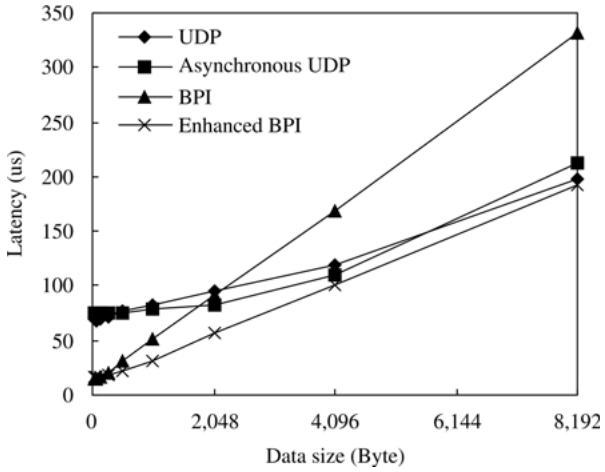

Figure 12. Latency of dominant side.

Figure 12 is the graph that shows the dominant latency of four protocols. Table 3 details which case each protocol belongs to. For example, the latency of Asynchronous UDP is dominated by the NIC side when the data size is bigger than 3 KB. For data smaller than 3 KB, it is dominated by the host side.

There are two interesting findings in Figure 12. One is that, except BPI, the latencies converge as the data size gets bigger. Surprisingly, this means that UDP has a similar throughput to the enhanced BPI by the equation described although the enhanced BPI is much lighter than UDP. The reason is that the latency of NIC dominates. In other words, regardless of how fast the host pumps the data out, NIC becomes a bottleneck, which is due to the per-byte overhead of hostTx—that is, EBUS-LBUS DMA overhead.

The second finding is that the latency of the enhanced BPI is dominated by the NIC side for most data sizes. Note that the user-level protocols such as BPI have very low overhead in the host side in comparison with UDP and Asynchronous UDP, but the throughput is limited by NIC.

Another issue is the time interval to detect the requests from the host, which is shown in Figure 11(c) as a bi-directional arrow. We observe that it takes at least 4 μs to detect a request from the host side. This interval continuously increases because the NIC side latency takes longer than the host. This can cause the send queue to overflow because the host sends the data more frequently than the capability of NIC.

From Figure 12 and Table 3, it is quite clear that the DMA performance is critical

Table 3. State of each protocol with data size

Case	UDP	Asynchronous UDP	BPI	Enhanced BPI
(b)	$<=5$ KB	$<=3$ KB	>128 B	>64 B
(c)	>5 KB	>3 KB	$<=128$ B	$<=64$ B

to reduce the overall latency and achieve a high throughput. A lesson is that the DMA engine of NIC and chipset of I/O bus have to be carefully designed for gigabit networks.

5.2. *Copy operation of user-level protocols*

Many user-level protocols are designed for high-speed networks [7, 14–21]. They remove the kernel from the communication path. However, by this reason, some of them should copy the network data to a DMA'able area [7, 14]. As described in Section 2, BPI copies data to the copy block. On the other hand, although Asynchronous UDP is a kernel-level protocol, it does not copy the network data at all.

Then, the question is which one is more efficient. Figure 13 shows the overall latency of UDP, Asynchronous UDP, and enhanced BPI. As expected, the enhanced BPI shows a smaller latency than the others. However, we noticed that the slope of enhanced BPI is steeper than that of Asynchronous UDP. Therefore, we experimented with a different processor to see the impact of the copy overhead since Asynchronous UDP has no copy overhead.

For example, Figure 14 shows the measurement results that are performed on a workstation equipped with an Intel 120 MHz Pentium processor. On this system, the copy operation is 3 times more expensive than on an Intel 180 MHz Pentium Pro processor. We can see that, in Figure 14, the latency of the enhanced BPI and that of Asynchronous UDP are crossed at the large data size (about 6 KB), and Asynchronous UDP shows better latency after the crossover.

It is generally believed that the user-level protocols have far smaller latency, but our evaluation shows that a user-level protocol can be heavier than an optimized kernel-level protocol. This unbelievable result says to us that the per-byte overhead,

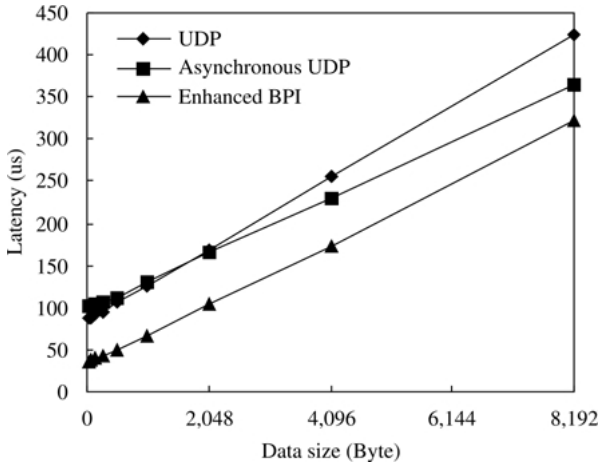

Figure 13. Overall latency on an Intel 180 MHz Pentium Pro processor.

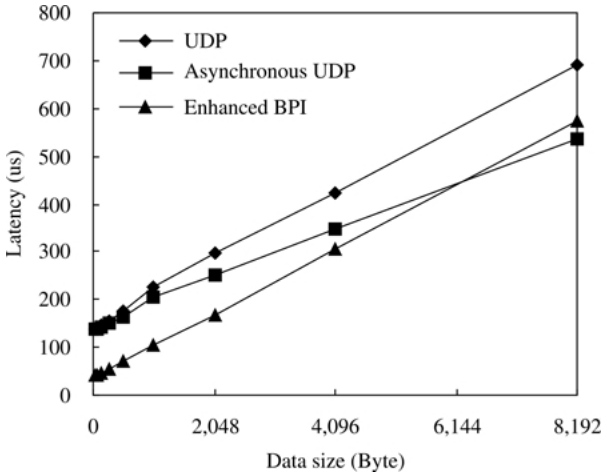

Figure 14. Overall latency on an Intel 120 MHz Pentium processor.

especially the copy overhead, is a very important factor to user-level protocols. Therefore, the user-level protocols should consider not only kernel bypassing but also copy avoidance like VIA [18].

5.3. *Cacheability of network buffer*

Finally, we verify the effect of the buffer cacheability. Figure 15 shows the overall latencies of BPI and enhanced BPI. Compared to BPI, the enhanced BPI is faster by 40%. The heavy latency of BPI, which is lager than that of UDP for larger data than

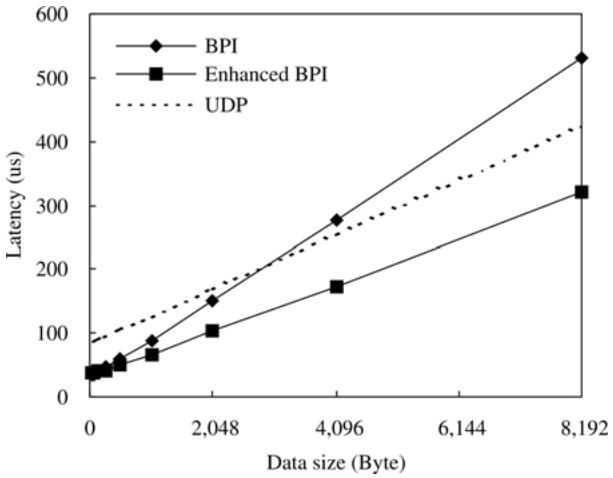

Figure 15. Overall latency of BPI and enhanced BPI.

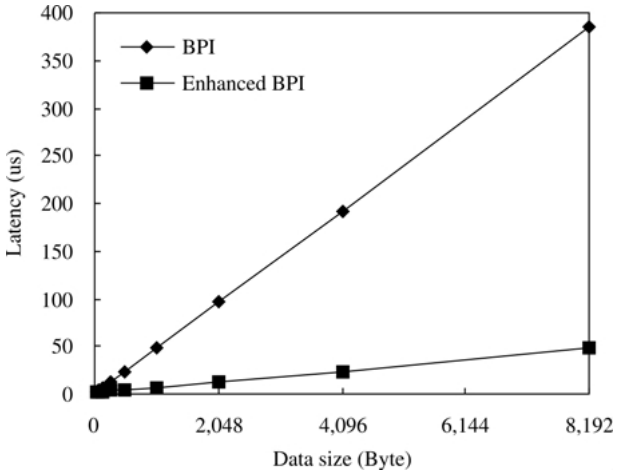

Figure 16. Comparison of copy overhead.

3 KB, is due to the uncached copy block. This is the main reason why the dominant latency of BPI is in the host side in Figure 12 and Table 3.

In the initialization of BPI, it allocates a memory area of the kernel space for the copy block and mmaps this area into the user space so that an application can directly access the copy block. When the copy block is mapped, BPI sets the PCD (Page-level Cache Disable) flag [26] of the page-table that maps the copy block. It means that the copy block of BPI is uncached because the PCD flag of the page-table entry controls the cacheability of individual pages in Intel Pentium Pro processor. Therefore, the copy from user buffer to the copy block is memory (or cache) to memory copy.

The enhanced BPI, however, uses the cacheable copy block. In this case, the copy from user buffer to the copy block is from memory (or cache) to cache. This leads to a 1/8 cheaper copy overhead than using the uncached copy block as in Figure 16. This means that the network buffer concerned with copy operation should be a cacheable memory area because an uncached memory area causes a large copy overhead. The current Myrinet Software version reflects the idea we found.

6. Conclusions

This paper presents a firmware-level latency analyses of a gigabit network. A contribution of this paper is that our latency analyses covers the NIC side as well as the host side, which is more thorough than the existing analysis based on the device driver level. We believe that an analysis without firmware taken into account gives only partial evaluation of gigabit networks.

Through the firmware-level measurements, we identify that the performance of DMA between the host and NIC memories is a limiting factor of reducing the

latency. A surprising result is that UDP and the enhanced BPI have a similar throughput on sender when the latencies are fully overlapping because NIC is the bottleneck. In theory, the enhanced BPI should be much lighter than UDP. This result implies something that has to be considered in designing I/O bus systems for gigabit networks.

Another finding is that the copy overhead is critical to the effectiveness of user-level protocols. The per-byte overhead such as copy overhead increases in proportion to the data size. Therefore, for large data sizes, even the user-level protocol can be heavier than an optimized kernel-level protocol, such as Asynchronous UDP. As a result, the user-level protocols have to avoid copying the data.

Finally, the overhead of copying uncached buffers offsets the advantage of user-level protocols. If the user-level protocol has to perform the copy operation, the buffer concerned with the copy operation should be a cacheable memory area. Otherwise, the uncached buffer induces the memory (or cache) to memory copy, which is a greatly expensive operation.

Acknowledgment

This research was supported by University Software Research Center Supporting Project from Korea Ministry Information and Communication.

References

1. N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet—A gigabit-per-second local-area network. *IEEE-Micro*, 15(1):29–36, 1995.
2. Gigabit Ethernet Alliance. Media Access Control (MAC) Parameters, Physical Layer, Repeater and Management Parameters for 1000 Mbps Operation. IEEE Std 802.3z, 1998.
3. D. D. Clark, V. Jacobson, J. Romkey, and H. Salwen. An analysis of TCP processing overhead. *IEEE Communications Magazine*, 27(6):23–29, 1989.
4. J. Kay and J. Pasquale. Measurement, analysis, and improvement of UDP/IP throughput for the DECstation 5000. *Proceedings of USENIX Winter Conference*, pp. 249–258, 1993.
5. A. Wolman, G. Voelker, and C. A. Thekkath. Latency analysis of TCP on an ATM network. *Proceedings of USENIX Winter Conference*, pp. 167–179, 1994.
6. C. Yoo, H.-W. Jin, and S.-C. Kwon. Asynchronous UDP. *IEICE Transactions on Communications*, E84-B(12):3243–3251, 2001.
7. Myricom Inc. Myrinet User's Guide. http://www.myri.com, 1996.
8. C. Dalton, G. Watson, D. Banks, C. Calamvokis, A. Edwards, and J. Lumley. Afterburner. *IEEE Network*, 7(4):36–43, 1993.
9. P. Druschel and L. L. Peterson. Fbufs: A high-bandwidth cross-domain transfer facility. *Proceedings of 14th ACM SOSP*, pp. 189–202, 1993.
10. H. J. Chu. Zero-copy TCP in Solaris. *Proceedings of 1996 Winter USENIX*, 1996.
11. J. C. Brustoloni and P. Steenkiste. Copy emulation in checksummed, multiple-packet communication. *Proceedings of IEEE Infocom '97*, pp. 1124–1132, 1997.
12. A. Gallatin, J. Chase, and K. Yocum. Trapeze/IP: TCP/IP at near-gigabit speeds. *Proceedings of 1999 USENIX Technical Conference*, 1999.

13. H.-W. Jin, C. Yoo, and S.-K. Park. Stepwise optimizations of UDP/IP on a gigabit network. *Proceedings of the 8th International Euro-Par Conference (Euro-Par 2002)*, LNCS, 2400:745–748, 2002.
14. T. V. Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A user-level network interface for parallel and distributed computing. *Proceedings of 15th ACM SOSP*, pp. 40–53, 1995.
15. S. Pakin, M. Lauria, and A. Chien. High performance messaging on workstations: Illinois fast messages (FM) for Myrinet. *Proceedings of Supercomputing '95 (SC95)*, 1995.
16. C. Dubnicki, L. Iftode, E. Felten, and K. Li. Software support for virtual memory-mapped communication. *Proceedings of the 10th International Parallel Processing Symposium*, 1996.
17. L. Prylli and B. Tourancheau. BIP: A new protocol designed for high performance networking on Myrinet. *Proceedings of IPPS/SPDP98*, 1998.
18. D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, A. M. Berry, E. Gronke, and C. Dodd. The virtual interface architecture. *IEEE Micro*, 18(2):66–76, 1998.
19. T. Takahashi, S. Sumimoto, A. Hori, H. Harada, and Y. Ishikawa. PM2: A high performance communication middleware for heterogeneous network environments. *Proceedings of Supercomputing 2000 (SC2000)*, 2000.
20. I. Pratt and K. Fraser. Arsenic: A User-accessible gigabit ethernet interface. *Proceedings of IEEE Infocom 2001*, pp. 67–76, 2001.
21. P. Shivam, P. Wyckoff, and D. Panda. EMP: Zero-copy OS-bypass NIC-driven gigabit ethernet message passing. *Proceedings of Supercomputing 2001 (SC2001)*, 2001.
22. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A treansport protocol for real-time applications. RFC 1889, 1996.
23. H.-W. Jin, K.-S. Bang, C. Yoo, J.-Y. Choi, and H.-J. Cha. Bottleneck analysis of a gigabit network interface card: Formal verification approach. *Proceedings of the 9th International SPIN Workshop on Model Checking of Software (SPIN 2002)*, LNCS, 2318:170–186, 2002.
24. S. Elbert, Q. Snell, A. Mikler, G. Helmer, C. Csanady, K. Stearns, B. MacLeod, M. Johnson, B. Osborn, and I. Verigin. Gigabit ethernet and Low-cost supercomputing. Technical report IS-5126. Ames Laboratory, 1998.
25. H.-W. Jin and C. Yoo. Latency analysis of UDP and BPI on Myrinet. *Proceedings of 18th IEEE International Performance, Computing, and Communication Conference (IPCCC'99)*, pp. 185–191, 1999.
26. *Intel Co. Pentium Pro Family Developer's Manual*, Volume 3: Operating System Writer's Guide. Order Number: 242692, 1995.