# Lightweight Messages: True Zero-Copy Communication for Commodity Gigabit Ethernet*

Hai Jin, Minghu Zhang, Pengliu Tan

Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

**Abstract.** *Gigabit Ethernet has become the main cluster interconnection for its low price and well backward compatibility. But the communication performance through Gigabit Ethernet is quite disappointing due to its performance discrepancies between the hardware and the communication software. To achieve over two-third physical bandwidth of a gigabit network interface, zero-copy protocol architecture is absolutely necessary. Unfortunately, it is impossible to realize true zero-copy communication over non re-programmable commodity Gigabit Ethernet adapters because the DMA engines cannot separate the protocol headers from the payload data directly. This paper introduces LM (Lightweight Messages), a true zero-copy communication mechanism which combines the application level fragmentation scheme with the driver level defragmentation scheme, for existing non-programmable Gigabit Ethernet adapters. Finally, experimental results show that LM can provide better performance than other lightweight communication approaches over commodity Gigabit Ethernet.*

## 1. Introduction

Gigabit Ethernet becomes the main commodity cluster interconnects due to its low price and well backward compatibility. Using Gigabit Ethernet, a high performance cluster system can be configured in a LAN environment. However, although the processors can reach gigahertz speed and the network hardware and I/O buses can provide gigabit bandwidth, the high overhead of the existing communication software fails to achieve gigabits per seconds communication bandwidth over Gigabit Ethernet.

Two main approaches are adopted to reduce the overhead of the existing communication software including the improvement of the TCP/IP stacks and the substitutions of TCP/IP stacks. The former focuses on implementing improved TCP/IP stacks [3, 6]. Two alternatives can be considered to the latter approach including communication protocols with efficient OS support [5, 15] and the user-level network communications [7, 9, 13]. One of the most important strategies of these approaches is the zero-copy communication software architecture.

User level network communication approaches are not feasible in commodity

* This paper is supported by National Science Foundation of China under grant 90412010.

Gigabit Ethernet because most of these facilities require special programmable NIC (*Network Interface Card*) hardware. TCP/IP stack substitutes are meaningful and can be used in many applications, but most conventional applications are coded for the socket APIs. Improved TCP protocols [5, 10] have been introduced with excellent performance. While in our approach, we dedicate to an improved UDP protocol for the considerations of its simple, low cost processing and good portability to real-time communications.

On the other hand, high throughput applications often send large packets while the MTU (*Maximum Transmission Unit*) in Gigabit Ethernet remains small (1500 bytes). Thus most Ethernet drivers use at least one data copy to separate the protocol headers from the payload data at the receiver side, unless the fragmentation can be done in hardware [14]. Defragmentation scheme in hardware is an extra complex task and many zero-copy approaches remain the last defragmentation data copy [5, 15]. *Speculative Defragmentation* [6] is proposed to eliminate the last defragmentation data copy over Gigabit Ethernet, but this approach needs the on-chip DMA hardware to separate the headers from the payload data. Zero-copy TCP in Solaris [3] is proposed to hold the entire TCP packet in the network interface buffer before segmentation. Taking into account existing simple commodity Gigabit Ethernet NICs, we find most of the on-chip DMA hardware has no capability of separating the headers from the payload data. So, can true zero-copy communication protocol still be feasible over these Gigabit Ethernet products?

This paper proposes LM (*Lightweight Messages*), which combines the application level fragmentation scheme with the driver level defragmentation scheme, to realize true zero-copy communication over non re-programmable Gigabit Ethernet adapters which have no capability of separating the protocol headers from the payload data. The organization of this paper is as follows: Section 2 describes the Gigabit Ethernet and the zero-copy communication software architectures. Section 3 introduces the design and implementation of LM mechanism in detail. The performance evaluations and results are shown in Section 4. Finally, Section 5 concludes this paper.

## 2. Gigabit Ethernet and Zero-Copy Software Architectures

### 2.1. Gigabit Ethernet

Gigabit Ethernet offers an excellent opportunity to build gigabit networks [4]. Although many Gigabit Ethernet NICs have the re-programmable capability and some researchers have engaged in re-programming the firmware to implement expected prototype systems [1, 14], but the current trend seems to indicate that the Gigabit Ethernet adapters will hardly provide any degree of programmability in the future, mainly due to price constrained design choices. *Jumbo Frames* is an alternative technique in Gigabit Ethernet adapter which uses larger MTU (up to 9000 bytes) to reduce the number of interrupts and the overhead of the communication protocol processing. But *Jumbo Frames* can not solve the problem of defragmentation data copy and it also requires *Jumbo Frames* supports of both end nodes and Ethernet switches. On the other hand, *Jumbo Frames* also introduces higher latencies in

Ethernet switches because most commodity switches use store-and-forward mechanism. *Coalescing Interrupts* technique is another feature in Gigabit Ethernet which lowers the CPU utilization by decreasing the interrupt frequency. Although this technique can achieve higher throughput to large messages, but it also brings higher end-to-end latencies for small messages due to a lower interrupt response.

### 2.2. Zero-Copy Software Architectures

Several zero-copy schemes have been proposed as the classification from [3, 6]:

*User-Level Communications [7, 9].* In these approaches, the interface memory is accessible or pre-mapped into user or kernel address space. The low level hardware abstractions for the network interfaces and special semantic restrictions of the APIs are also provided. Unfortunately, these approaches require complicated hardware and can hardly be implemented in simple Gigabit Ethernet NICs.

*User/kernel Shared Memory [8, 10].* This scheme provides shared semantics between the user and the kernel address space and the data is moved between the shared memory and the NIC by DMA. The user-kernel data copy is eliminated by the per-process buffer pool that is pre-mapped in both the user and kernel address spaces. One major disadvantage of this approach is the new semantics of APIs.

*User/kernel Page Remapping with Copy on Write [3].* This implementation uses DMA to transfer data between the NIC and the kernel buffers. On the other hand, the user buffers are remapped to the kernel buffers by editing the MMU table and perform data copies only needed. Page remapping is adopted in LM approach.

*Blast Transmitting [11, 12].* Blast protocols can improve the throughput of large transfers by delaying and coalescing acknowledges messages. The driver's buffer chain is modified to permit the separation of the protocol headers from the payload data of an incoming packet by the NIC. Finally, the data parts are directly written to the user space or remap the kernel space to user space.

## 3. LM: True Zero-Copy Communication Mechanism

The key procedures for zero-copy communications are the fragmentation and defragmentation schemes. We first introduce the original fragmentation and defragmentation schemes of a standard UDP packet with 7168 bytes size: On the sender side, the user data are copied to the user send buffer, then the data are split into five regular IP fragments and each fragment is copied to the kernel buffer, and the five IP fragments are DMAed to the NIC finally. On the receiver side, the NIC DMAs each incoming fragment to a socket buffer and all the fragments are re-assembled and copied to the application user receive buffer.

In LM, the data copies between the user space and the kernel space are replaced by page remapping. When using user/kernel page remapping and DMA technique to realize zero-copy communication, the remapped user/kernel space must be page aligned and the DMA regions must be continuous physical address spaces. On the transmission side, if the user/kernel space is page remapped and the data is consecutively placed, how to place the protocol headers during the kernel processing

is to be solved. Obviously, if the protocol headers and the payload data, which compose of an Ethernet frame, are placed to different kernel spaces, they cannot be DMAed to the NIC using only one DMA operation. Two DMA regions are used [6] (one for the protocol headers and one for the payload data) to transmit an Ethernet frame to the NIC. But in the implementation of LM, we can hardly control the DMA operations like that. Before an Ethernet frame is DMAed to the NIC, its protocol header and payload data must be placed in consecutive kernel space. On the other hand, an incoming Ethernet frame is simply DMAed to the pre-allocated socket buffer without the separation of protocol headers and the payload data. How to DMA several incoming fragment frames to one consecutive kernel buffer is the main challenge to be solved in LM, which leads to the true zero-copy communication.

### 3.1 Application Level Fragmentation Scheme

In order to assemble an Ethernet frame within the consecutive kernel space, we introduce the *application level fragmentation* (ALF) scheme as shown in Fig.1, where the data to be sent are split into several regular parts with pre-defined size and copied to the application user send buffer with regular holes to place the protocol headers.

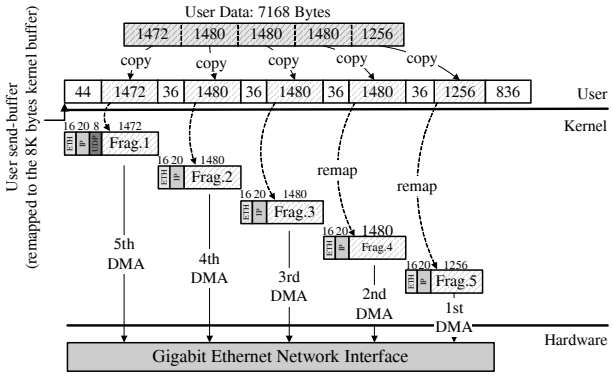


**Fig. 1.** Application level fragmentation mechanism

7168 bytes user data to be sent are split into five blocks with 1472, 1480, 1480, 1480, and 1256 bytes, respectively. Each block presents the payload data of an IP fragment. When copied to the user/kernel remapped user send buffer, these five blocks are separated with holes of predefined size (44 bytes is reserved at the head of the user send buffer to place the protocol headers of the first fragment packet and 36 bytes to place the protocol header of other fragments). Finally, the headers and the payload data of each of the five Ethernet frames are placed in consecutively kernel space and each frame can be DMAed to the NIC using only one DMA operation. To cater for the driver level defragmentation scheme, we reverse the transmission order of the IP fragments that the last IP fragment is transmitted firstly and finally the first fragment.

### 3.2 Driver Level Defragmentation Scheme

To realize true zero-copy receiving communication (we denote this as the *LM_RECV traffic*), the *driver level defragmentation* (DLD) scheme is introduced as shown in Fig. 2. All fragmenting packets are DMAed to one single kernel buffer (has been remapped to user receive buffer) while the payload data are placed consecutively. Due to the reversing transmission order mentioned in the *ALF* mechanism, the last fragment is placed at the tail of the kernel buffer firstly. The second last fragment is placed consecutively to the last one and the protocol headers of the last fragment are overwritten by the payload data of the second fragment. Then the third last fragment rewrites the protocol headers of the second last fragment, and so on. Finally, when all the fragmenting packets arrive, the total payload data are placed in continuous kernel buffer space and the application can use these data directly without data copies.

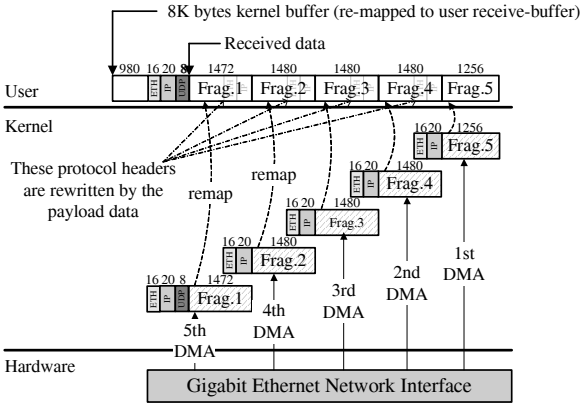


**Fig. 2.** Driver level defragmentation mechanism

### 3.3 Admission Control

A successful *DLD* processing duration means that only and exactly all the fragments from the same packet are to be received. This can be operated by the *admission control* (AC) module to prevent any other interfering packets arriving at the network device during the *DLD* processing duration. Two components concerned to the *AC* module at each node are considered including the *LM_RECV_ADDR* list and the *DLD* controller. The *DLD* controller controls whether the *DLD* scheme is active or not. A member of the *LM_RECV_ADDR* list is the following data structure:

```
structure LM_RECV_ADDR
{ long long address;/*address of the node with active DLD
mechanism*/
  int flag;/*00,01,10*/
  LM_RECV_ADDR *next;/*pointer to next structure*/
}
```

where *address* means the address of the node with active *DLD* mechanism; *flag* presents the relation between the listed address and the local address: “00” presents no

*LM_RECV traffic* between the listed address and the local address, "01" means that the local node is transmitting *LM_RECV traffic* to some other remote nodes, and "10" presents the listed address is the local address which means that some other remote nodes are transmitting *LM_RECV traffic* to the local node. Thus, the following control messages are defined and processed by the *AC* module, including *LM_REQ, LM_ENDME, LM_ACQ_OK, LM_ACQ_REJ, LM_APPEND_ADDR,* and *LM_REMOVE_ADDR*. Table 1 shows the meanings of these messages in detail.

**Table 1.**Control messages for the admission control module

| **Control Messages** | **Descriptions** |
|---|---|
| *LM_REQ* | Require *LM_RECV* traffic with specified data size |
| *LM_ENDME* | *LM_RECV* traffic is finished |
| *LM_ACQ_OK* | The *LM_REQ* request is accepted |
| *LM_ACQ_REJ* | The *LM_REQ* request is rejected |
| *LM_APPEND_ADDR* | Append specified address to the *LM_RECV_ADDR* list |
| *LM_REMOVE_ADDR* | Remove specified address from the *LM_RECV_ADDR* list |

Only one node is set to work under active *DLD* mechanism at one instant. Fig. 3 describes the processing of different type of admission control messages.

**Admission control processing**

```
1   Switch (control message) {
2       case LM_REQ:
3           if (LM_RECV_ADDR list is null) {
4               broadcast LM_APPEND_ADDR message to all nodes;
5               activate the DLD scheme;
6               send LM_ACQ_OK message to the applicant; }
7           else    send LM_ACQ_REJ message to the applicant;
8           break;
9       case LM_ENDME:
10          broadcast LM_REMOVE_ADDR message to all nodes; break;
11      case LM_ACK_OK:
12          begin to transmit specified packet; break;
13      case LM_ACK_REJ:
14          if(destination address is not listed in the LM_RECV_ADDR list)
15              begin to transmit specified packet; break;
16      case LM_APPEND_ADDR:
17          append specified address to the LM_RECV_ADDR list; break;
18      case LM_REMOVE_ADDR:
19          removing specified destination address from the LM_RECV_ADDR list; break;
20      default:
21          break;
22  }
```

**Fig. 3.** Admission control processing in LM

Thus, if node *A* wants to send *LM_RECV traffic* to node *B*, it must ensure that no *DLD* mechanism is active by checking its *LM_RECV_ADDR* list. If the *LM_RECV_ADDR* list is null, node *A* sends a *LM_REQ* message to node *B* and requires *LM_RECV traffic* transmitting to node *B*. If the *LM_RECV_ADDR* list is not

null and the address of node *B* is not listed in the list, node *A* also transmits the specified traffic to node B while node B receives the traffic in normal one-copy method. If the *LM_RECV_ADDR* list is not null and the address of node *B* is listed in the list, the traffic transmission from node *A* to node *B* is canceled. In most cases, the sender node may transmit the IP fragments to the receiver node and whether these fragments are zero-copy or one-copy received is determined by the receiver node. Although the round trip delay from a *LM_REQ* message of the sender and the *LM_ACQ_OK* or *LM_ACQ_REJ* message of the receiver is about 30 ms, but the data transfer is not delayed at the sender side to the point of the sender user applications.

## 4. Performance Evaluations

In this section, we first give the test environments which are consisted of 16 nodes single-hop cluster system. The cluster nodes may be interconnected with one Gigabit Ethernet switch or not due to different measurement requirements. Table 2 shows the configurations of each cluster node and the Gigabit Ethernet switch.

**Table 2.** Testing environments configuration

| | |
|---|---|
| Hardware | Intel Celeron 2.0 GHz CPU; 256M 266MHz DDRAM memory; 32 bit 33MHz PCI bus |
| Software | RedHat 9.0; Linux kernel 2.4.20; RTAI 3.0r4; RTNET 0.8.0 |
| Adapter | Intel PRO/1000 Gigabit Ethernet adapter |
| Ethernet Switch | TP-Link TL-SG1024 Gigabit Ethernet Switch |

The basic performance measurements, including the bandwidth, latency and CPU utilization using LM mechanism are measured compared with that of standard UDP communications. In most cases, the *Jumbo Frames* and *Coalescing Interrupts* features are turned off for both the LM and standard UDP communications without special statements. The influences of some flexible features provided by Gigabit Ethernet adapters, such as *Jumbo Frames* and *Coalescing Interrupts*, are also provided. We also compare LM with other lightweight communication approaches such as the CLIC, GAMMA and the *Speculative Defragmentation* approaches. A back-to-back connection for latency and bandwidth measurements is adopted only involving two cluster nodes and a Gigabit Ethernet switch is used if involving more than 2 cluster nodes or at special occasions. The bandwidth and end-to-end latency are measured through “ping pong” micro-benchmarks modified from NetPIPE [2] benchmark tools, where the packet size increases exponentially and each size repeats 5000 times. The CPU utilization measurements are based on “looper” processes which sit in tight loops counting as fast as they possibly can for each known CPU cycles on the system.

### 4.1 Performance Comparison of LM with Standard UDP

Fig. 4 shows the basic performance of LM compared with standard UDP protocol over Gigabit Ethernet, as well as the impacts of the *Jumbo Frames* and *Coalescing*

*Interrupts* techniques in Gigabit Ethernet to LM.

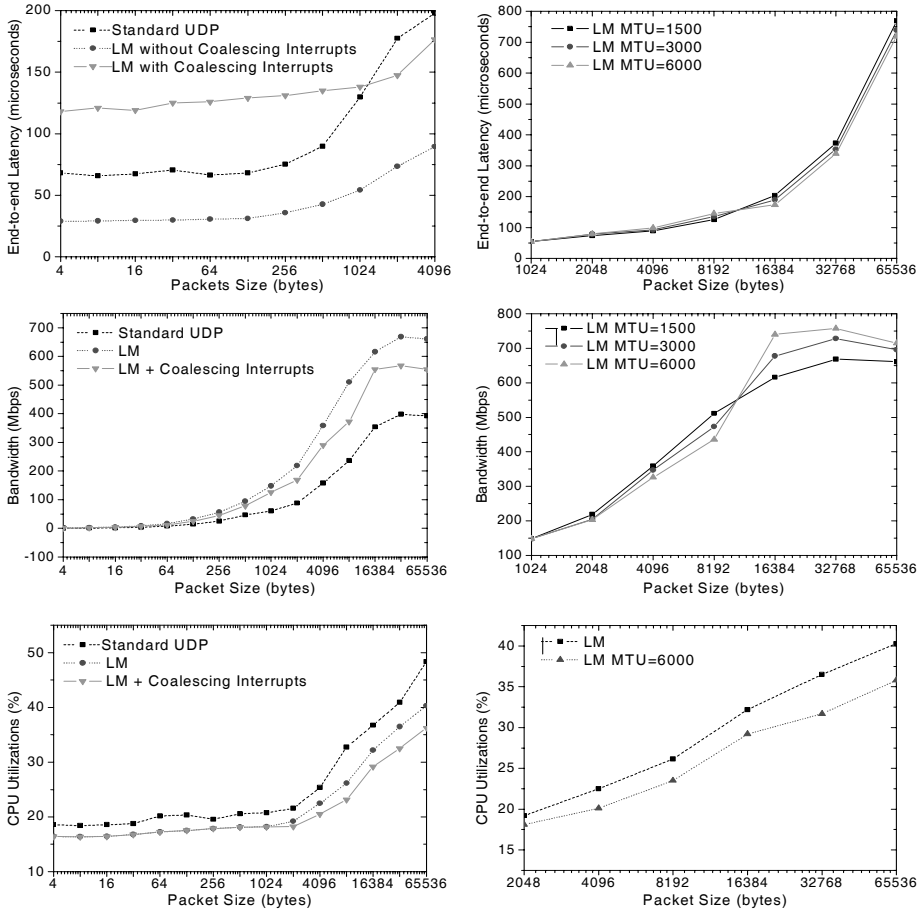


**Fig. 4.** Performance comparison of LM to standard UDP communication and the impacts of *Jumbo Frames* and *Coalescing Interrupts* to LM

The results in Fig. 4 show that LM can achieve 28.9 μs latency and 668.6 Mbps bandwidth compared with that of 68.1 μs latency and 398.6 Mbps bandwidth for standard UDP. The CPU utilization using LM is much lower than that of the standard UDP. On the other hand, the *Jumbo Frames* and *Coalescing Interrupts* techniques can reduce the CPU utilizations and *Jumbo Frames* can provide higher bandwidth for large message transmission, but they also result in worse end-to-end delay for small and medium sized messages.

### 4.2 Comparison of LM with Other Approaches

We also compare LM with other communication mechanisms for Gigabit Ethernet,

such as GAMMA, CLIC and *Speculative Defragmentation* (Spec. Defrag.) over standard MTU size and 33 MHz and 32 bits PCI buses. Table 3 shows that LM provides a slightly better performance than the other three. LM, CLIC and GAMMA depend nothing on the Gigabit Ethernet adaptors but CLIC and GAMMA can not directly support those traditional applications coded for the socket APIs. *Speculative Defragmentation* mechanism is similar to LM but it requires that the Gigabit Ethernet adapters have the capability to separate the protocol header from the payload data and cannot be implemented on non re-programmable Gigabit Ethernet adapters. LM can provide a feasible true zero-copy communication for non programmable Gigabit Ethernet produces, and it also supports socket API coded applications.

**Table 3.** Performance of LM Compared with other communication mechanisms

| Approaches | Latency | Bandwidth | Zero-copy receiving | Adapters requirements |
|---|---|---|---|---|
| LM | 28.9μs | 668.6Mbps | Yes | No |
| CLIC | 36μs | 450Mbps | No | No |
| GAMMA | 32μs | 568Mbps | No | No |
| Spec. Defrag. | Not mentioned | 600Mbps | Yes | Yes |

## 5. Conclusions and Future Works

In this paper, we introduce *Lightweight Messages* (LM), a feasible true zero-copy communication mechanism over existing non re-programmable Gigabit Ethernet adapters. The main contribution of LM is to realize true zero-copy communication over the non re-programmable Gigabit Ethernet adapters by reserving "holes" at the user sender buffer, sending IP fragments of large packets in reverse order and overwriting the protocol headers of the previous fragment at the receiver side. LM can efficiently eliminate the last data copy of incoming fragmenting packets under the conditions that the DMA engine has no capability to separate protocol headers from the payload data. The performance results show that LM can achieve slightly better performance than other relative lightweight communication mechanisms for Gigabit Ethernet under similar measurement environment. In sum, LM provides a feasible method to realize true zero-copy communication using any simple low price Gigabit Ethernet products, and this is valuable because the current trend seems to indicate that the Gigabit Ethernet adapters will hardly provide any degree of programmability in the future mainly due to price-constrained design choices.

Many works will be improved to LM: such as more efficient admission control processing, adaptive interrupts controls or interrupt substitutes like polling. In the next step, we will implement the LM mechanism under real-time micro kernel and plant it to distributed real-time and embedded control systems.

## References

[1] I. Pratt and K. Fraser, "Arsenic: a user-accessible gigabit Ethernet interface", *Proceedings of the 12th Annual Joint Conference of the IEEE Computer and Communications Societies*

*(INFOCOM 2001)*, Vol.1, 22-26 April 2001, pp.67-76.

[2] NetPIPE: A Network Protocol Independent Performance Evaluator, Available: http://www.scl.ameslab.gov/netpipe/.

[3] H. K. J. Chu, "Zero-Copy TCP in Solaris", *Proceedings of the USENIX 1996 Annual Technical Conference*, San Diego, CA, USA, January 1996, pp.253-264.

[4] P. Balaji, P. Shivam, P. Wyckoff, and D. Panda, "High Performance User Level Sockets over Gigabit Ethernet", *Proceeding of IEEE International Conference on Cluster Computing*, Sept. 2002, pp.179-186.

[5] G. Ciaccio, M. Ehlert, and B. Schnor, "Exploiting Gigabit Ethernet capacity for cluster applications", *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN 2002)*, 6-8 Nov. 2002, pp.669-678.

[6] C. Kurmann, M. Muller, F. Rauch, and T. Stricker, "Speculative Defragmentation - a Technique to Improve the Communication Software Efficiency for Gigabit Ethernet", *Proceedings of the 9th International Symposium on High-Performance Distributed Computing,* Pittsburgh, PA, USA. 2000, pp.131-138

[7] V. Eiken, A. Basu, V. Buch, and W. Vogels, "U-net: A userlevel network interface for parallel and distributed computing", *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP-15)*, Cooper Mountain, CO, USA, Dec. 1995.

[8] P. Druschel and L. L. Peterson, "FBufs: A high-bandwidth crossdomain transfer facility", *Proceedings of the 4th ACM Symposium on Operating System Principles*, Asheville, NC, December 1993, pp.189-202.

[9] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A. Merritt, E. Gronke, and C. Dodd, "The virtual interface architecture", *IEEE Micro,* Vol.18, No.2, March–April 1998, pp.66-76.

[10] V. S. Pai, P. Druschel and W. Zwaenepoel, "I/O-Lite: A unified I/O buffering and caching system", *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI '99)*, 1999, pp.15-28.

[11] J. B. Carter and W. Zwaenepoel, "Optimistic Implementation of Bulk Data Transfer Protocols", *Proceedings of the 1989 Sigmetrics Conference*, May 1989, pp.61-69.

[12] S. W. O'Malley, M. B. Abbot, N. C, Hutchinson and L. L. Peterson. "A transparent blast facility", *Internetworking: Research and Experience*, Vol.1, No.2, Dec. 1990.

[13] H. Tezuka, A. Hori, Y. Ishikawa, and M. Sato, "PM: An Operating System Coordinated High Performance Communication Library", *Proceedings of High-Performance Computing and Networking*, 1997.

[14] P. Shivam, P. Wyckoff, and D. Panda, "EMP: Zero-copy OSbypass NIC-driven Gigabit Ethernet Message Passing", *Proceedings of 2001 International Conference on supercomputing (SC2001)*, Denver, Colorado, USA, Nov. 2001.

[15] A. F. Diaz, J. Ortega, A. Canas, F. J. Fernandez, M. Anguita and A. Prieto, "The lightweight protocol CLIC on Gigabit Ethernet", *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS 2003)*, April 2003.