

On the Performance of Maestro2 High Performance Network Equipment, Using New Improvement Techniques

Shinichi Yamagiwa¹ Kevin Ferreira¹ Luis Miguel Campos¹ Keiichi Aoki²
Masaaki Ono² Koichi Wada² Munehiro Fukuda³ Leonel Sousa⁴

¹ PDM&FC, Rua Latino Coelho, 87, 1050-134 Lisboa, Portugal.
{yama, kferreira, lcampos}@pdmfc.com

² Parallel and Distributed Computing Laboratory, University of Tsukuba, 1-1-1 Tennodai,
Tsukuba, Ibaraki, 305-8573, Japan. {k1,ono,wada}@is.tsukuba.ac.jp

³ Computing and Software Systems UW1-331, University of Washington Bothell,
18815 Campus Way NE Bothell WA, 98011-8246. mfukuda@u.washington.edu

⁴ IST/INESC-ID, Rua Alves Redol 9, 1000-029, Lisboa, Portugal. las@inesc-id.pt

Abstract

Cluster computers have become the vehicle of choice to build high performance computing environments. To fully exploit the computing power of these environments, one must utilize high performance network and protocol technologies, since the communication patterns of parallel applications running on clusters require low latency and high throughput, not achievable by using off-the-shell network technologies. We have developed a technology to build high performance network equipment, called Maestro2. This paper describes the novel techniques used by Maestro2 to extract maximum performance from the physical medium and studies the impact of software-level parameters. The results obtained clearly show that Maestro2 is a promising technology, presenting very good results both in terms of latency and throughput. The results also show the large impact of software overhead in the overall performance of the system and validate the need for optimized communication libraries for high performance computing.

1 Introduction

Whether they are used as stand alone or as part of larger environments (such as Grids), clusters have become the the facto mechanism to build high performance computing environments. The growing interest in clusters has been driven by fast performance increases in commodity computing resources such as processors, memories, networks, and its peripherals.

However the tendency among researchers building such cluster computers is to use off-the-shelf compo-

nents. As such the most popular clusters use Ethernet and TCP/IP as its communication device and protocol, respectively. Although there are obvious advantages (for instance price and simplicity) to build clusters using off-the-shelf components especially for computation, there are also serious disadvantages especially when using conventional networking technology. Those conventional WAN or LAN-based network devices and protocols are not optimal for clusters, since they include overheads to guarantee reliability in wide area communications [5]. Given that clusters are very often organized in geographically localized spaces such as laboratories or offices, hardware and software for communication can be optimized to improve the overall performance of the system.

However to fully exploit the computing power of these environments, one must utilize high performance network and protocol technologies, since the communication patterns of parallel applications running on clusters require low latency and high throughput, not achievable by using off-the-shell network technologies. For instance, distributed parallel simulations that exchange small messages among processors at each simulation step require low latency in communication. Many matrix computations such as FFT that exchange large messages among processors require high throughput.

To address this issue, we have developed a high performance network technology, optimized for cluster computing, called Maestro2. Maestro2 is the successor of the Maestro network developed by the research group led by Professor Koichi Wada at University of

Tsukuba in Japan. Maestro's architecture has been described in detail in [9, 10, 8].

Through the experience gained during the Maestro project, we have identified two issues that restricted low level performance (hardware and data link layer protocol). The first issue occurred at the link layer. In [9] we proposed the network burst transfer at the link layer. In the network burst, the sender aggregates multiple packets that compose part of a message, and sends them in burst to the receiver. We confirmed network burst was able to keep the utilization ratio of the physical medium higher than conventional link layers. However, when the sender finished transferring the number of packets decided at the beginning of the transfer, the burst transfer is inevitably terminated. This termination increases arbitration overhead in acquiring the physical medium. The second issue occurred in the switch. Maestro's switch transfers one message at a time. In addition, the transfer is stopped when the source or destination network buffer becomes full or empty. This mechanism decreases the dynamic communication performance.

In [8] we have proposed solutions to both problems and have analyzed the performance of the proposed solutions both through simulation and by extensive testing of data link layer protocols.

Section 2 describes the novel techniques used by Maestro2 to extract maximum performance from the physical medium. Section 3 describes in detail the environment, both in terms of hardware and software, used to conduct all the experiments. Based on this description, all results should be easily reproduced and validated by any researcher in the field. In section 4 we present experimental results that show the performance of Maestro2 hardware at various levels (data link layer, application layer), using well known benchmarks and network performance measurement tools. As an example we also present results for some currently available gigabit-based network technologies (Gigabit Ethernet and Myrinet). Section 5, concludes this paper by indicating future areas of research in this area.

2 Principles of Maestro Technology

Maestro's architecture has been extensively described in the literature [9, 10, 8], and we ask you to refer to those sources for a detailed understanding of the technical issues. In here we will only describe, briefly, the current implementation of communication hardware (network interface and switch) using Maestro2 technology and the novel techniques used by Maestro2 to extract maximum performance from the physical medium, namely, *continuous burst, out of*

order switching and high performance communication software.

2.1 Basic hardware implementation

A simple Maestro2 network is composed of network interfaces (NI) and a switch box (SB) as shown in Figure 1. Each network interface is connected via two LVDS (Low Voltage Differential Signaling) [2] cables for transmission and reception, and is connected to a commodity processing element such as a PC or a Workstation via a 64bit@66MHz PCI bus. The connection between NI and SB is full-duplex, and the peak bandwidth is 6.4Gbps. Currently, the SB has eight ports for connections to NIs. One or more ports can increase the fan out of the switch by cascading SBs. The NI includes a NI manager, a PCI interface, network FIFO buffers, a link controller (MLC-X) and LVDS transmitter/receiver. The NI manager works as a processor element in charge of handling communications, and is composed of a PowerPC603e@300MHz and 64Mbyte SDRAM. The PCI interface maps the address space of the SDRAM and host processor's memory into the PowerPC's address space. The 8Kbyte network FIFO buffers store incoming and outgoing messages. The MLC-X is a full duplex link layer controller on which the continuous network burst transfer is implemented. MLC-X supports two communication channels between the network FIFO buffers. And finally, LVDS transmitter/receiver drives its physical medium under control of MLC-X. It transmits and receives data via its 6.4Gbps full duplex link. The PCI interface, network buffers, and MLC-X are implemented into the Virtex-II FPGA chip [7]. The SB consists of eight SB interfaces, an SB manager and a switch controller. Each SB interface manages two ports and includes a message analyzer. The message analyzer extracts each message header of an incoming message, and passes it to the SB manager. The SB manager consists of a PowerPC603e, 32Mbyte SDRAM, and a routing circuit that generates and writes requests to the switch controller. The switch controller transfers messages from source to destination(s) using the out-of-order mechanism mentioned next.

2.2 Performance improvements implementation

Continuous network burst transfer The first technique is the continuous network burst transfer. This technique allows for the continuation of the network burst as long as there are packets in the network buffer. This can be done by inserting the *continue command* at each boundary of the burst. We use a frame configuration similar

to the one used for the original network burst transfer [9]. That is, the frame consists of one or more fine-grained *packets* and a header. A packet consists of user data. At the beginning of the transfer, the header of the frame is dispatched to the physical layer, and arbitration is done to acquire the physical medium. When there are multiple packets to be sent in the network buffer, the continuous network burst is invoked for all or part of those packets. As the burst transmission progresses, it is possible to append subsequent packets to the network buffer. The conventional network burst never continues the transmission beyond the number of packets predetermined at the beginning of the transmission because the frame length has to be determined at the beginning of the header transfer. However, the continuous network burst is able to continue the transmission by inserting the continue command into the first burst transmission without having to reacquire the physical medium. Figure 2 compares the conventional network burst transfer with the continuous network burst transfer. It assumes that messages are written into the sender's network buffer one after another. The conventional network burst transfer terminates soon after it finishes transferring the amount of packets determined at the beginning of the transfer. Therefore, the arbitration time for acquiring physical medium accumulates. On the other hand, the continuous burst transfer can reduce the number of arbitrations. This technique increases the utilization ratio of the physical medium, and achieves higher throughput than the conventional network burst.

Out of order switching While routing messages at a switch, individual messages might be blocked when the destination buffer becomes full. Such occurrences force subsequent messages to be blocked if the switch only routes messages in-order. This in-order switching decreases performance when the next message to route is for a different destination and when the destination buffer is not full. To improve efficiency in switching, we have implemented an *out-of-order switching* mechanism. In the out-of-order switching, multiple transfer requests are stored in a transfer reservation buffer. For each request it is examined whether or not its destination buffer is full. If the buffer is not full, the corresponding request is marked as active. The switch works by picking up active requests and routing the messages to

their destinations. The out-of-order switching is applicable to various kind of switch architectures, such as crossbar, single or multiple buses-based, and so on. Gains in performance by using out-of-order switching are only significant when the data traffic coming from the several network interfaces attached to the switch is intense.

High performance communication software In addition to the improvements at the network hardware level, we have also designed and implemented a high performance communication software.

Conventional protocols such as TCP, etc, were devised to guarantee reliability in wide area communications. However, in cluster environments, those same mechanisms become the limiting factor of the potential performance of the physical network. Conventional protocols do not allow the parallel application software to bypass the communication layer(s) and send/receive messages directly. As such, the current communication software environment for parallel applications can not eliminate the overhead imposed by the the conventional communication protocol layer, as pointed out in [5]. A solution was suggested in [6], which basically consists of a zero-copy mechanism that allows the parallel application to bypass the communication layer(s) and to request communication services directly to the physical hardware.

We have developed a message passing library for Maestro2, named *MMP*, which can be used to obtain a much higher level of performance than conventional communication protocols, over Maestro2. The two key observations in developing *MMP* were: First, in order for the communication not to impact the overall performance of the parallel application, it needs to avoid unnecessary overhead. In particular, small messages require low latency and large messages require high throughput. Second, computation must have higher priority in accessing the CPU than communication. Usually, conventional protocols are responsible for creating the chunks of data that are to be passed along the communication stack, to manage the ordering of messages received and provide for independence of port among other tasks. While the host processor is occupied by these tasks, computation can not occur which leads to a substantial degradation in overall performance.

To address the first issue, we implemented a zero

copy mechanism. This reduces the communication overhead, by avoiding unnecessary copying of data. Moreover, we have also implemented a dynamic allocation strategy for pin-downed area. This mechanism allows the application program to pin-down any memory area dynamically, allows the paging mechanism by the operating system and makes the memory area to be cacheable as soon as *MMP* finishes using it. To deal with the second issue, *MMP* communication functions were made to be non-blocking. This allows for actual time at which the communication takes place to be flexible. In addition, complex communication operations such as the creation of data chunks, migrates to the physical network. Therefore, the communication code will be overlapped with the computation code on the host processor.

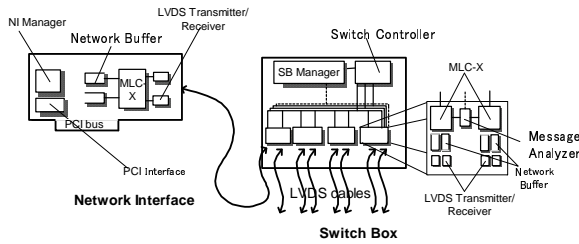


Figure 1. Maestro2 Cluster Network

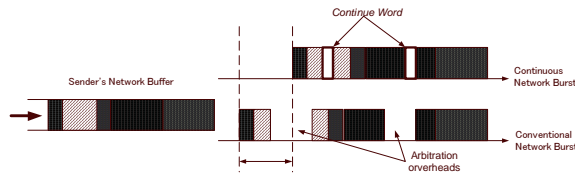


Figure 2. Continuous network burst vs. conventional network burst

3 Experimental Environment

The experimental results described in section 4 were obtained using the following set of hardware:

Hardware

Host machine Dual 1.0GHz Intel Pentium-3 with Serverworks HE-SL chipset, with 512Mbyte SDRAM PC133 and PCI bus 64bits/66Mhz

Maestro Network Interface Maestro2 Y-PCI64-P

Gigabit ethernet Network Interface Netgear GA622T

Myrinet Network Interface M3F-PCI64B-2

Software The Operating System used was Linux 2.4.7-10smp. To obtain results we used both *Netperf 2.2p13* and our implementation of *ping-pong*.

In all experiments, the only user level process(es) running on the environment were the software performance tools, the application being evaluated, or both depending on the experiment being conducted.

The results obtained by us should be easily reproduced by any researcher in the field, using the same conditions described above.

4 Experimental Results

In this section, we present experimental results that show the performance of Maestro2. As an example we also present results for other network technologies, namely, Gigabit Ethernet and Myrinet.

A well known benchmark, *ping-pong*, was used extensively during the result gathering phase. The ping-pong benchmark, measures the time a packet of *S* size takes to travel from Host A to Host B and back to Host A (round-trip time). For a given packet size, the round-trip time is heavily influenced by the overhead added by the software. To validate our implementation of *ping-pong*, we compared its results with the results obtained using the TCP_RR option of *Netperf* software (at the TCP/layer), and concluded that they almost identical.

In all cases, the metrics measured were *latency* and *throughput*. Latency is calculated by dividing the round-trip time in half for small messages (smaller than 64 bytes). Throughput is calculated by dividing the packet size by one half of the round-trip time.

Low Level Transfer

At this level we aim to measure the raw performance of the network hardware, with the only overhead being added by the data link layer protocol.

4.1 Data Link Layer

4.1.1 Experiment 1

Description Using ping-pong, we allow for the message size to be transferred to vary between a minimum of 32 bytes to a maximum of 262144 bytes. We measure performance for Maestro2 (with and without continuous network burst). Transfers are

from host’s memory to host’s memory, without participation of a switch.

Objectives First, to study the impact of message size in terms of throughput for the Maestro2 under consideration, secondly, to validate continuous network burst technique.

Analysis of Results Latency results are given in table 1 and throughput results are given in figure 3. From figure 3 we observe that the throughput, for message sizes up to 256 KBytes, increases, reaching a maximum value of 2695 Mbit/sec. Contrary to our expectations, at the data link layer, the implementation of continuous network burst does not yield significant improvements in terms of throughput. The most significant result of this experiment is however the extremely short latency measured at this level. For comparison purposes, we show latency results for Myrinet and Gigabit ethernet.

Table 1. Latency at data link layer, with and without Continuous Network Burst

Measurement at Data Link Layer	Latency
Maestro2 with Cont. Net. Burst	1.1 μ s
Maestro2 without Cont. Net. Burst	1.2 μ s
Gigabit ethernet	8.8 μ s
Myrinet	2.7 μ s

High Level Transfer

In here, we measure performance for different communication mechanisms: TCP, specialized communication libraries (MMP in Maestro2 and GM [1] in Myrinet).

4.2 TCP-Transport Layer

To measure performance using basic TCP as transport layer, we used the well known software tool Netperf [4]. The main advantage of using such measurement tools is wide acceptance of the performance results obtained by other researchers. For sake of comparison with the results obtained at the data link layer experiments, we also measured the performance using ping-pong

4.2.1 Experiment 2

Description For a fixed-size MTU (1500 bytes), fixed-size sender/receiver buffers (default socket

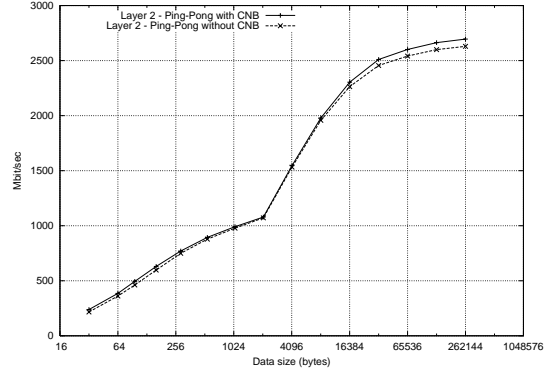


Figure 3. Maestro2 - Data link layer throughput, using ping-pong, with and without Continuous Network Burst (CNB)

values: send buffer 16384 bytes and receive buffer 87380 bytes). We allow for the message size to be transferred to vary between a minimum of 4 bytes to a maximum of 1048576 bytes. We measure ping-pong performance of Maestro2 (with continuous network burst), we also show results for Gigabit Ethernet and 100Mbit Ethernet.

Objectives To show Maestro2 TCP/IP performance, using MTU = 1500.

Analysis of Results Latency results are given in table 2 and throughput results are given in figure 4. From figure 4 we see that Maestro2’s throughput is, not surprisingly, far superior than the obtained using other technologies. Maestro2’s throughput is 230% superior to Gigabit Ethernet and 450% than fast ethernet for MTU=1500bytes. However we can clearly observe the impact of software overhead in the results. Results obtained at this layer, show Maestro’s throughput to be five time smaller than at the data link layer, and latency to be almost fifty times higher! We must be careful looking at the results obtained for Gigabit Ethernet. In fact we were able to obtain almost 550Mbit/sec throughput when we increased the socket size by a significant amount (to 256Kbytes). Moreover, we are well aware of throughput results close to 1Gbit/sec under special conditions (see [11, 3]). Unfortunately those results are not achievable using the conditions described in this experiment, which we feel are representative of normal use.

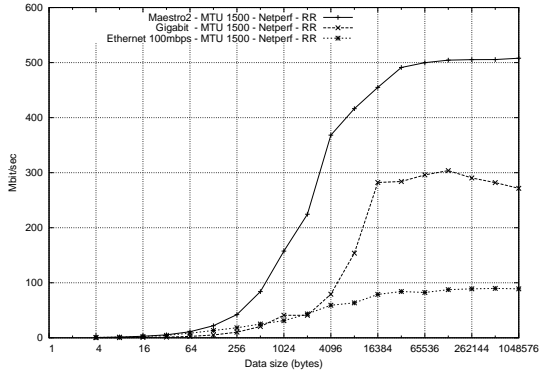


Figure 4. TCP Throughput, MTU=1500, using ping-pong test (Netperf with TCP_RR option)

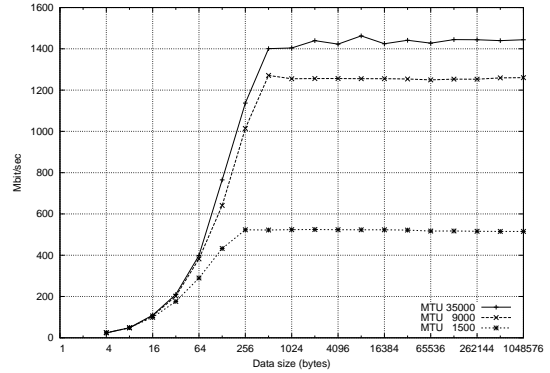


Figure 5. TCP Throughput over Maestro2 using a range of MTU, measured with Netperf

Table 2. Latency at TCP layer, for MTU 1500

Measurement for TCP layer	Latency
Maestro2	49 μ S
Netgear Gigabit	200 μ S
Fast ethernet (100mbps)	52 μ S

4.2.2 Experiment 3

Description Using Netperf, and a selected set of MTU sizes, for each MTU size chosen we allow for the message size to be transferred to vary between a minimum of 4 bytes to a maximum of 1048576 bytes. We measure performance for Maestro2 (with continuous network burst).

Objectives First, to study the relationship between MTU size and message size and if possible to determine an *optimal* size of MTU.

Analysis of Results In figure 5 we plot the throughput for Maestro2, for a set of MTU sizes. From the figure we can conclude that there is a significant improvement in performance when the MTU size increases from 1500bytes to 9000bytes (145% increase in performance) and also from 9000bytes to 35000bytes (15% increase in performance). However there was no additional improvement for larger MTU sizes. These results were expected since one of the software overheads, namely breaking the IP packets in smaller frames suitable for small MTU sizes is eliminated. We

are still studying the fact that there is no improvement for MTU=65000bytes (as compared with MTU=35000bytes) since the maximum size of IP packets is 64Kbytes.

4.2.3 Experiment 4

Description Using an *optimal* MTU size, 35000 bytes for Maestro2 as determined in experiment 4.2.2, 9000 bytes for Myrinet and 1500 byte for Gigabit Ethernet, we compare the throughput performance when the message size to be transferred varies from a minimum of 4 bytes to a maximum of 1048576 bytes. We use fixed-size sender/receiver buffers (default socket values).

Objectives To compare the performance at TCP layer using *optimal* MTU sizes for each technology.

Analysis of Results Latency results are given in table 3 and throughput results are given in figure 6. Maestro2 shows a performance that is roughly 19% better than Myrinet and 390% better than Gigabit ethernet.

Latency results show that Myrinet is about 20% better than Maestro2.

4.3 Specialized Communication Software Library

In this section we measure the performance using specialized communication software library, for Maestro2 we use MMP, and, for comparison purposes for Myrinet we use GM.

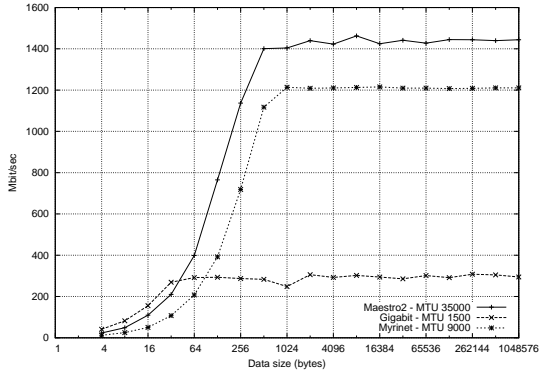


Figure 6. TCP Throughput, best MTU, using Netperf

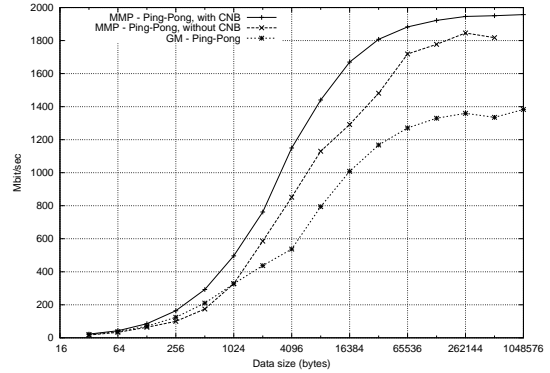


Figure 7. MMP Throughput over Maestro2 and GM over Myrinet, using ping-pong

Table 3. Latency measured at TCP layer

Measurement for TCP layer	Latency
Maestro2 (MTU=35000)	49 μ s
Netgear Gigabit (MTU=1500)	200 μ s
Myrinet (MTU=9000)	40 μ s

Table 4. Latency using Specialized Library

Specialized Library	Latency
Maestro2-MMP	11.5 μ s
Myrinet-GM	11.0 μ s

4.3.1 Experiment 5

Description Using *ping-pong*, we allow for the message size to be transferred to vary between a minimum of 32 bytes to a maximum of 1048576 bytes. We measure performance for Maestro2 (with and without continuous network burst) using MMP, and for Myrinet using GM.

Objectives To compare the performance of MMP versus GM.

Analysis of Results Latency results are given in table 4 and throughput results are given in figure 7. Two main aspects can be observed. The first relates to the fact that continuous network burst becomes an effective technique when the physical capacity of the medium is under-utilized due to software overhead. The second is the fact that Maestro2 throughput is roughly 40% higher than Myrinet and stands at almost 2000Mbit/sec. Latency results are virtually the same for both technologies.

4.3.2 Experiment 6

Description None

Objectives To compare the performance of Maestro2 at different levels of software overhead.

Analysis of Results Figure 8 shows the performance results, using *ping-pong* (using TCP), MMP and at the data link layer. The most striking aspect of these curves is the impact of software overhead in the performance results. Using TCP we obtain a 65% loss in performance as compared with the performance obtained at the data link layer, whereas when using MMP the loss in performance is only 38%. In terms of raw performance we observe that at the data link layer we obtain a transfer rate of almost 2700 Mbit/sec (84% of the physical layer) whereas using MMP and TCP we achieve only 2000Mbit/sec and 1000Mbit/sec respectively.

5 Conclusions

In this paper we have shown that Maestro2 technology is able to provide high performance, in terms of low latency and high throughput, both at the data link layer, TCP and MMP. Maestro2 is able to utilize up

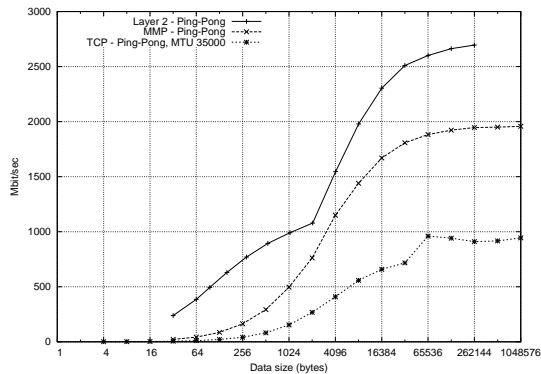


Figure 8. Maestro2 - TCP versus MMP versus Data link layer throughput, using ping-pong

to 84% of the physical layer bandwidth (one way) and the high performance communication software (MMP) yields an improvement of over 100% (throughput) in comparison with standard linux TCP implementation.

The next steps in the development of the technology are to obtain results using MPI, to study scalability issues and provide shared memory support.

Acknowledgments

We would like to thank Agência de Inovação (Portugal) for supporting Maestro2's research via a grant under *Programa POCTI - Medida 1.2*. This research was also supported by Japan Society for the Promotion of Science, a Grant-in-Aid for Scientific Research(C),14580361, 2002.

References

- [1] Myricom Corporation. <http://www.myrinet.com/>.
- [2] IEEE Standard Department. IEEE standard for low-voltage differential signals (LVDS) for scalable coherent interface (SCI). Technical report, 1996.
- [3] Richard Hughes-Jones, Ralph Spencer, and Steve Parsley. High data rate transmission for vlbigrd using the academic network. In *2nd eVLBI Workshop*, 2003. http://www.jive.nl/jive/evlbi_ws/presentations/hughes-jones.pdf.
- [4] Rick Jones. *Netperf*. <http://www.netperf.org/netperf/NetperfPage.html>.

- [5] Vijay Kararncheti and Andrew A. Chien. Software overhead in messaging layers: Where does the time go? In *International Conference on Architectural Support of Programming Languages and Operating Systems*, pages 526–531, 1994.
- [6] Hiroshi Tezuka, Atsushi Hori, Yutaka Ishikawa, and Mitsuhsa Sato. Pm: An operating system coordinated high performance communication library. In *High-Performance Computing and Networking*, volume 1225 of *Lecture Notes in Computer Science*, pages 708–717. Springer-Verlag, 1997.
- [7] Xilinx Inc. *Virtex-II Platform FPGA Data Sheet*, 2002. <http://www.xilinx.com>.
- [8] Shinichi Yamagiwa, Luis Miguel Campos, Keiichi Aoki, Masaaki Ono, Tetsuya Sakurai, and Koichi Wada. Maestro2: A new challenge for high performance cluster network. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, July 2002.
- [9] Shinichi Yamagiwa, Munehiro Fukuda, and Koichi Wada. Design and performance of maestro cluster network. In *IEEE International Conference on Cluster Computing (CLUSTER2000)*, November 2000.
- [10] Shinichi Yamagiwa and Koichi Wada. Design and implementation of message passing library on maestro network. In *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 87–90, August 2001.
- [11] Evgeniy Zaitsev. *Gigabit Network Adapters on 64bit PCI Bus, AMD760MPX Platform*. <http://www.digit-life.com/articles2/gig-eth-64bit-amd760/index.html>.