

Maestro2: High Speed Network Technology for High Performance Computing

Keiichi Aoki*, Shinichi Yamagiwa†, Kevin Ferreira†‡, Luis Miguel Campos†, Masaaki Ono*, Koichi Wada*, Leonel Sousa‡

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

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

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

Abstract— Cluster computers have become the de facto mechanism 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 overhead in communication software. 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.

I. INTRODUCTION

The tendency among researchers building cluster computers is to use off-the-shelf components, like Ethernet devices and TCP/IP communication protocol. 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 [7]. 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.

To achieve low latency and high throughput required by high performance parallel computing, we have developed a high performance network technology, 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 [11], [12], [10], [3].

Through the experience gained during the Maestro project, we have identified two issues that restricted performance in data link layer and routing. The first issue occurred at the link layer and is focused on the utilization ratio of 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 [10] we have proposed solutions to both problems and have analyzed the performance of the proposed solutions both through simulation and by extensive experiments of data link layer protocols.

In this paper, we describe the hardware implementation for performance improvement, and present the performance evaluation of the communication softwares developed on Maestro2.

Section II describes the novel techniques used by Maestro2 to extract maximum performance from the physical medium. In section III we present experimental results that show the performance of Maestro2 hardware at various levels, using well known benchmarks and network performance measurement tools. For comparison, we also present results for some currently available gigabit-based network technologies (Gigabit Ethernet and Myrinet). Section IV, concludes this paper with indicating future plans of this research.

II. MAESTRO2 TECHNOLOGY

Maestro's architecture has been extensively described in the literature [11], [12], [10], [3]. Here we will 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 network burst*, *out-of-order switching* and high performance communication software *MMP*.

A. 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. The fan-out of the switch can be increased by cascading SBs. The NI includes a NI manager (composed by a PowerPC603e@300Mhz and 64Mbyte SDRAM), a PCI interface, network 8-kbyte FIFO buffers, a link controller (MLC-X) and LVDS transmitter/receiver. The PCI interface,

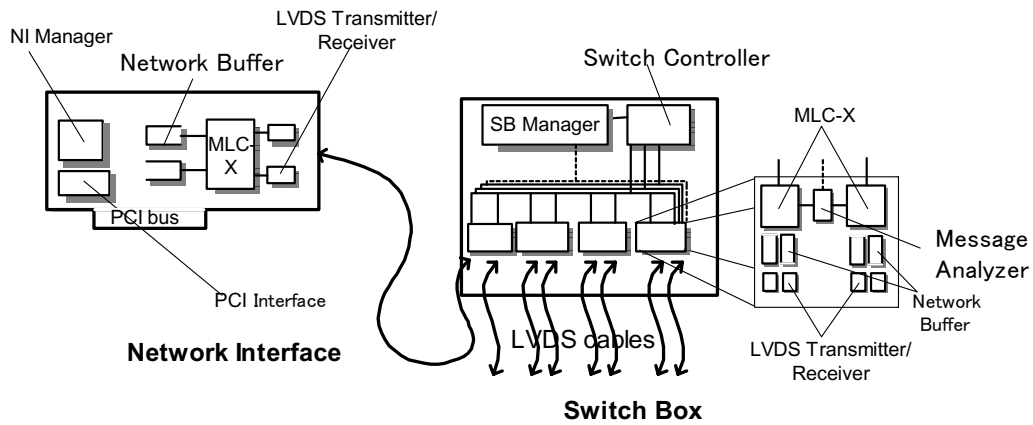


Fig. 1. Maestro2 Cluster Network

network buffers, and MLC-X are implemented into the Virtex-II FPGA chip [9]. 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.

B. Performance improvements

Continuous network burst

This technique allows for the continuation of the network burst as long as there are packets in the network buffer. This is 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 [11]. 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 a 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. 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. 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. 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

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 [7]. A solution was suggested in [8], 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. 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.

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 can be performed by NI. Therefore, communication will be overlapped with computation.

Moreover, we have developed MPI [4] library on MMP(MPI

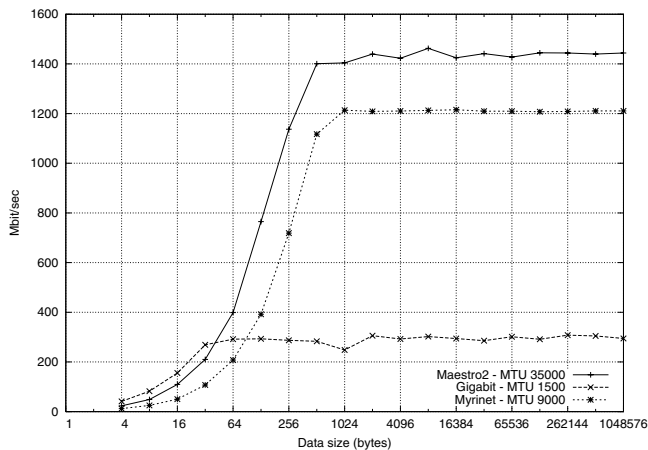


Fig. 2. TCP Throughput, best MTU, using Netperf

over MMP). We have adapted the MPICH to build MPI over MMP. MPICH is composed of several communication interface layers: user interface, abstract device interface(ADI), protocol, channel interface. We have implemented the channel interface with MMP to build MPICH for Maestro2. This way user programs written for MPI can communicate over Maestro2 and obtain higher communication performance without changes.

III. EXPERIMENTAL RESULTS

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

The experiments were done with the experimental environment shown in table I.

TABLE I
EXPERIMENTAL ENVIRONMENT

Component	Description
Host Machine	Dual 1Ghz Pentium 3 with Serverworks HE-SL chipset, 512 Mbyte PC133 SDRAM
O.S.	Linux 2.4.7-10 SMP
Myrinet	PCI64B,M3F-SW16-8F
Gbit Ethernet	Netgear GA622T,GS508T

To obtain results we used both Netperf 2.2p13, Netpipe and an implementation of *ping-pong*.

In all cases, the metrics measured were *latency* and *throughput*. Latency is calculated by dividing the round-trip time in half. Throughput is calculated by dividing the data size by latency.

The experimental evaluation of Maestro2 were done at high level data transfer. we measure performance for different communication mechanisms: TCP, specialized communication libraries (MMP in Maestro2 and GM [1] in Myrinet) and MPI library.

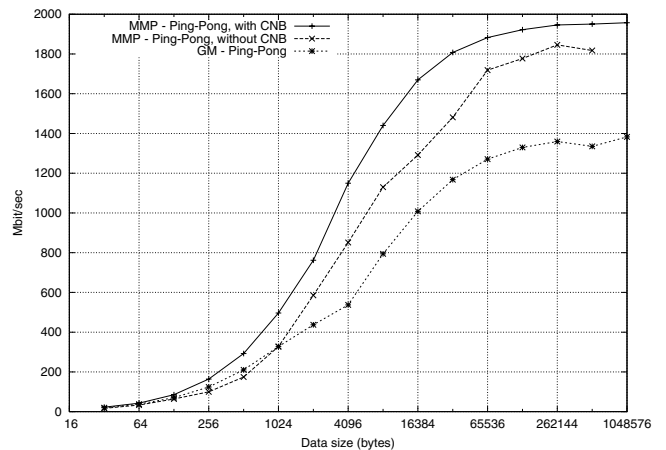


Fig. 3. MMP Throughput over Maestro2 and GM over Myrinet, using ping-pong

A. TCP-Transport Layer

To measure performance using basic TCP as transport layer, we used the software tool Netperf [6].

To compare the performance at TCP layer using *optimal* MTU sizes for each technology, we used MTU size, 35Kbytes for Maestro2 as showed in [3], 9Kbytes for Myrinet and 1500 byte for Gigabit Ethernet (as NETGEAR does not support jumbo frames), we compare the throughput performance when the message size to be transferred varies from a minimum of 4 bytes to a maximum of 1Mbytes. We use fixed-size sender/receiver buffers (default socket values). Latency results are given in table II and throughput results are given in figure 2. 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. However we can clearly observe the impact of software overhead in the results. Results obtained at this layer, show Maestro's throughput to be about half than at the data link layer (Figure 4), and latency to be almost fifty times higher (see [3])! 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 1Gbps under special conditions (see [5]). Unfortunately those results are not achievable using the conditions described in this experiment, which we feel are representative of normal use.

TABLE II
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

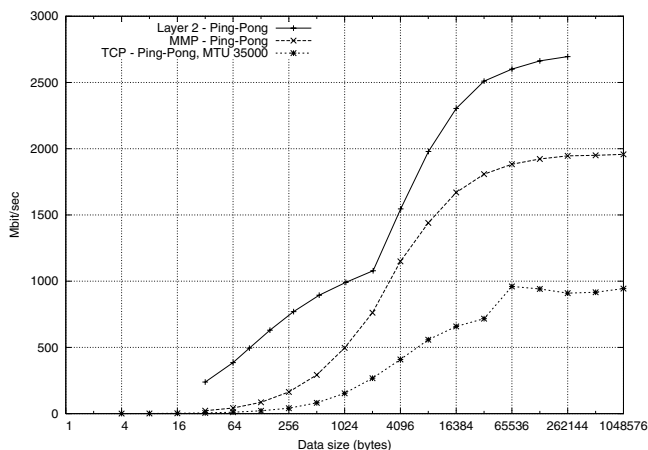


Fig. 4. Maestro2 - TCP versus MMP versus Data link layer throughput, using ping-pong

B. 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.

To compare the performance of MMP with GM, we used *ping-pong*, and we allow for the message size to be transferred to vary between a minimum of 32 bytes to a maximum of 1Mbytes. We measure performance for Maestro2 (with and without continuous network burst) using MMP, and for Myrinet using GM. Latency results are given in table III and throughput results are given in figure 3. 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 (see [3]). The second is the fact that Maestro2 throughput is roughly 40% higher than Myrinet and stands at almost 2Gbps. Latency results are about the same for both technologies.

Figure 4 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, where as 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 2.7Gbps (84% of the physical layer) where as using MMP and TCP we achieve only 2Gbps and 1Gbps respectively.

TABLE III
LATENCY USING SPECIALIZED LIBRARY

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

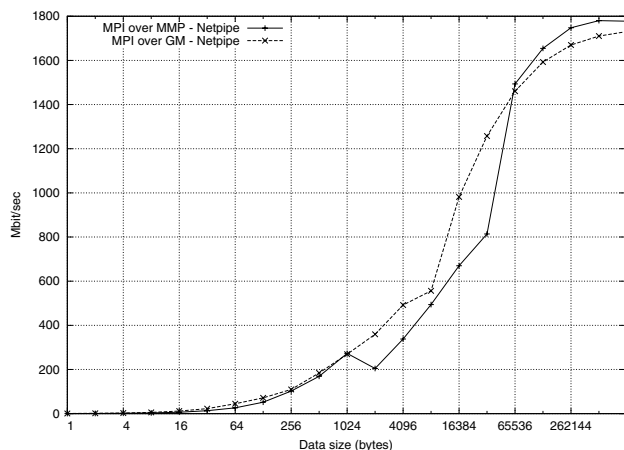


Fig. 5. MPI over MMP versus MPI over GM throughput, using netpipe

C. MPI Library

We measure the performance using MPI library and compare the performance between Maestro2 and Myrinet. We used MPI over MMP for Maestro2, and MPI over GM for Myrinet. To measure the performance, we used netpipe compiled for MPI.

Figure 5 shows the performance of MPI over MMP and MPI over GM. The maximum throughput of MPI over MMP is 1.78Gbps. This archives about 96% of raw MMP performance. On the other hand, the maximum throughput of MPI over GM is 1.71Gbps. Therefore we confirmed that the maximum throughput of MPI over MMP is 70Mbps higher than MPI over GM.

IV. CONCLUSIONS

In this paper we have shown that Maestro2 technology is able to provide high performance, in terms of low latency and high throughput, using TCP, MMP and MPI. Maestro2 is able to utilize up 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. We have also shown that MPI over MMP archives 96% of MMP performance and it outperform MPI over GM.

The next steps in the development of the technology are to study scalability issues and provide shared memory support.

REFERENCES

- [1] Myricom Corp. <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] Shinichi Yamagiwa et al. On the performance of maestro2 high performance network equipment, using new improvement techniques. In *IEEE International Performance, Computing, and Communications Conference (IPCCC2004)*, April 2004.
- [4] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*, 1995. <http://www-unix.mcs.anl.gov/mpich/>.
- [5] Richard Hughes-Jones, Ralph Spencer, and Steve Parsley. High data rate transmission for vlbgrid using the academic network. In *2nd eVLI Workshop*, 2003. http://www.jive.nl/jive/evlbi_ws/presentations/hughes-jones.pdf.

- [6] Rick Jones. *Netperf*.
<http://www.netperf.org/netperf/NetperfPage.html>.
- [7] Vijay Karamcheti 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.
- [8] 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.
- [9] Xilinx Inc. *Virtex-II Platform FPGA Data Sheet*, 2002.
<http://www.xilinx.com>.
- [10] 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.
- [11] Shinichi Yamagiwa, Munehiro Fukuda, and Koichi Wada. Design and performance of maestro cluster network. In *IEEE International Conference on Cluster Computing (CLUSTER2000)*, November 2000.
- [12] 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.