Experience with the design of a communication protocol for PC clusters

Jichao Zhang, Weimin Zheng, Di Chang, Huaxia Xia

Dept. of Computer Science and Technology. Tsinghua Univ.
Beijing 100084, P.R.China
zjc99@mails.tsinghua.edu.cn
zwm-dcs@tsinghua.edu.cn

Abstract. With the advent of high-speed LANs and rapid development of PC technologies, the communication software evolution has lagged behind. Traditional communication protocol stacks fail to deliver much of the performance of physical network to user applications. So efficient communication protocols are needed to take full advantage of the high performance of underlying interconnection networks. This paper analyzes the critical issues in designing an efficient communication protocol and introduces our experience with the design and implementation of FMP (Fast Message Passing). FMP is a reduced user-level message passing system designed for Myrinet-based PC clusters. It provides high-speed, reliable and in-order message delivery, and supports multiplexing and protective accessing of the network interface. Moreover, standard parallel programming interface including PVM/MPI on top of FMP are both implemented.

Keywords: PC cluster; message passing system; communication protocol; FMP; Myrinet

1 Introduction

PC clusters based on high-speed interconnection networks offer a cost-effective and scalable alternative to MPPs. Compared with MPPs, PC clusters exploit standardized interconnection networks and communication protocols and have lower cost, more available software and more accessibility [1]. The dramatic advance of the performance of computing components and the advent of high-speed interconnection networks such as Myrinet [2] provide PC clusters with the potential performance comparable to that of MPPs. Communication software is crucial for cluster systems, which affects the efficiency of parallel computing, the adaptability to the applications and the scalability of the clusters. However, the advance in network hardware solely is not sufficient to speed up the communication performance, and efficient communication protocols are needed to deliver the high performance of network to user applications. Traditional communication protocols such as TCP/IP fail to delieve much of the network performance to the applications for their low efficiency, which has been a bottleneck of clusters systems. It's necessary to develop high performance communication software to match the high-speed of lower level physical network and

bridge the gap of the performance of user-level applications and that of lower level network hardware.

Existing communication protocols are often based on low-speed and unreliable network interconnections, and incorporate complex communication mechanism and redundant functionalities. Moreover, the communication systems reside in the operating system and so message passing processes involve system calls and context switches. As a result, the communication systems are characterized by large processing overhead, which prevents them from fully exploiting the performance of the physical network. Detailed analysis shows that the main communication overhead contributors are redundancy of messaging layers, excessive data copies, operating system involvement and context switches, and error control [6][7]. In consideration of the communication overhead contributors, the most promising technology to improve the communication efficiency is to the implement the user-level communication. User-level communication removes operating system involvement from the critical communication path and provides direct user-level access to the network. However, user-level communication protocol has to maintain multiplexing and protective accessing of the network between competing processes without operating system serving as the coordinator. This problem can be solved either by restricting network accessing to one process or by implement mechanism which ensure multi-processor multiplexing the network. In summary, an efficient communication protocol should targeted the following aims:

- Provide high-speed, reliable, and in-order message delivery.
- Share the physical network among several processes.
- Providing protection between processes accessing the network simultaneously.
- Provide well-known and standardized programming interface.

Despite the great advance in the research of communication software, it still lags behind the network interconnections in terms of performance. FMP is designed to address these communication problems. FMP(Fase Message Passing) is a reduced user-level communication protocol based on Myrinet network. It aims to fully utilize the high performance network equipments and bridge the gap between the performance of raw hardware and that of user applications. The communication layers

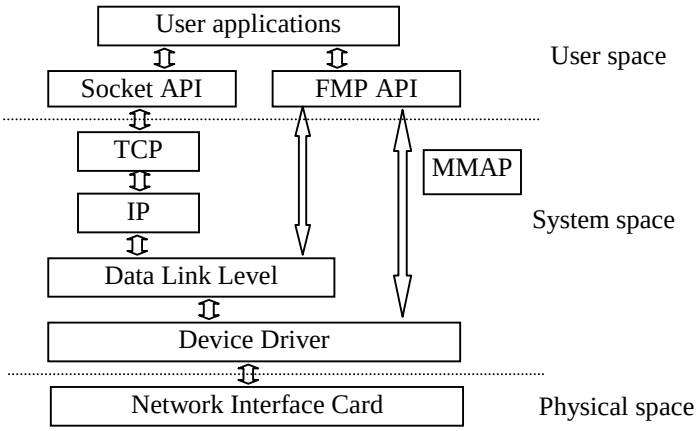

Fig. 1. Communication layers of TCP/IP and FMP

of TCP/IP stack and FMP are compared in Fig. 1. As is shown in the figure, TCP/IP incorporates excessive messaging layers and redundant functionalities in layers, which contributes to its low efficiency and hides the high performance of network hardware from user applications. Compared with TCP/IP, FMP reduces messaging layers, provides direct user-level access to the network interface, and avoids excessive data copies. Moreover, FMP removes the operating system from the critical communication path, reduces the overhead of operating system involvement in messaging and context switches.

The remainder of this paper is organized as follows. Section 2 gives an overview of FMP, including the LAN platform and software architecture of FMP. The details of the implementation of FMP are discussed in Section 3. Section 4 illustrated the performance results of FMP and MPI [3]/PVM [4] over FMP (MPI-FMP/PVM-MPI), followed by the analysis of the results, and some other related research projects are covered in this section. Finally, Section 5 presents our conclusions and the future work.

2 Overview of FMP

2.1 The LAN platform – Myrinet based PC Cluster

FMP LAN platform is a PC cluster based on Myrinet network. Each PC is a SMP with two Pentium III 550 MHz microprocessors running Windows NT, 128MB main memory, and a Myrinet network interface card(NIC) on PCI Bus.

The PCs in the cluster are connected by Myrinet from Myrient Inc. Myrinet is a high performance system area network (SAN), based on the technology used for communication within concurrent and parallel supercomputers. It has a maximum transfer rate of 1.28Gbits/s and an error rate of 10^{-15}. Myrinet consists of three basic components: a switch, an NIC per PC, and the cables connecting the cards to the switches. The switch transmits variable-length packets using wormhole routing and provides hardware flow control via back-pressure, in-order delivery. Each Myrinet NIC contains a programmable network controller called LANai, three DMA engines, and up to 1 MB of SRAM. The LANai processor runs the Myrinet Control Program(MCP) storied in SRAM and controls the actions of three DMA engines. One of the three DMA engines takes care of extracting incoming messages from the network to the SRAM, another moves messages in the SRAM to the network, and the third one moves data from NIC SRAM to host memory and vice-versa.

2.2 Software architecture

The description of the protocol stack is illustrated in Fig. 2.

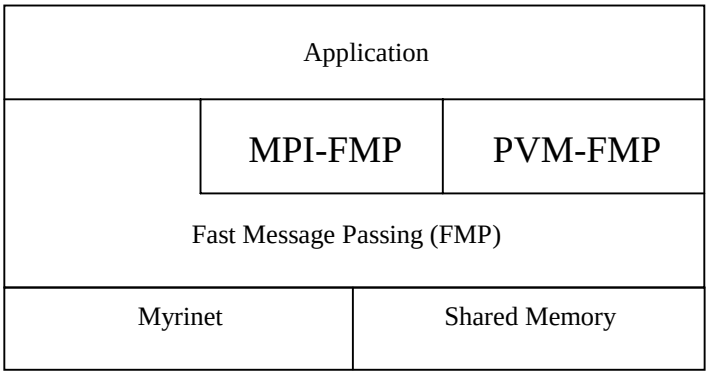

Fig. 2. Discription of the protocol stack

FMP is a highly optimized, high performance, and reliable messaging layers for Myrinet, which provides a virtual interface for the network. It presents a low-level communication interface, which can be directly used by user applications and servers as an intermediate layer for higher level communication protocols. FMP interface consists of several basic send and receive communication functions, providing basic communication functionalities and services. FMP is an asynchronous protocol which doesn't have to establish a connection between the communication processes before communication. Messages are directly transmitted between processes with the communication functions and each message has a process ID and a message ID as message identifies. This standard communication model matches the low-level communication protocol with high-level communication system, and makes it convenient for the portability of high-level communication environment such as MPI/PVM on top of FMP. We have implemented MPI-FMP/PVM-FMP to provide more functionalities and ease of use for user applications.

3 Design and implementation issues of FMP

While reducing the messaging layers and avoiding system calls improve the communication performance considerably, there are some other important issues related to the functionality and performance of communication protocols: the communication model, data movement by DMA vs Programmed I/O (PIO), message pipelining, flow control and multiplexing and protection [5][6]. FMP addresses these issues as follows.

3.1 Data movement by DMA vs PIO

During communication, messages must be transferred from the host memory to the NI and vice-versa. On Myrinet, there are two strategies for effecting these transfers: DMA or Programmed I/O (PIO). Data movements have a significant influence on the

performance of communication protocols, so efficient data movements are essential for obtaining high performance.

DMA operations free the host processor from communication, and so data movement can proceed in parallel with host computation. It seems that DMA always outperforms PIO. However, the setup cost of DMA operations maybe as much as that of data movement itself, which makes it not suitable for short messages. In addition, DMA operations deal with physical addresses while user applications deal with virtual address, and so address mappings are needed for DMA operations. Data can be copied to a buffer inside the operating system kernel or to a pinned-down buffer before starting the DMA operation. Dynamically pinning user pages can eliminate data copies, which implements the ze

Compared with DMA, PIO directly moves data between host memory and NI memory, and has much lower initialization overhead. But PIO requires expensive operating system involvement which should be eliminated from the critical communication path. For long messages movement, operating system involvement incurs great overhead. So PIO is not suitable for long messages. A comparison of DMA vs PIO is given in Fig. 3.

In consideration of these different properties, we choose different methods according to the size of messages: PIO for short messages (<512bytes) and DMA for large ones (⩾512bytes).

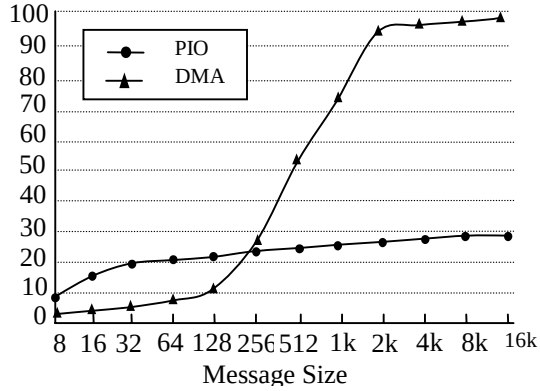

Fig. 3. Comparison of DMA and PIO modes

3.2 Credit-based Flow control mechanism

It's critical for message passing systems to guarantee reliable data delivery. However, data transfer may be unreliable due to unreliable network hardware or buffer overflow(flow control) problems. Because all networks have finite buffers, flow control is necessary to achieve reliable delivery, ensuring a receiver has enough buffer space to store incoming messages. When the network hardware is extremely reliable, as in the case of Myrinet which has an error rate of $10^{-15,}$ the hardware proves to be almost error-free. Based on Myrinet, FMP assumes the network to be reliable and focuses solely on preventing buffer overflow problems.

FMP implements credit-based flow control mechanism, which ensures reliable data transfer and eliminates the overhead of retransmissions. Each process allocates some credits for each other process which will send messages to it. Every time a process sends a message to the other process, it must have a credit. And if it runs out of the credits, the sender has to wait until the corresponding receiver returns some credits. When the receiver receives a certain number of messages, it returns the credits to the sender which can be reused by the sender. For the receiver returns the credits to the

sender by attaching the number of the credits in the messages, no much additional retransmission overhead is incurred.

The incurred overhead of flow control in FMP is tested on our PC cluster. In terms of communication latency, there is only about 2us overhead.

3.3 Message pipeling

For a complete message transfer, there can be four DMA operations: from host memory to NI memory(host-to-NI) and from NI memory to network(NI-to-network) at the sender side, from network to NI memory(network-to-NI) and from NI memory to host memory(NI-to-host) at the receiver side. Generally, these DMA operations are sequential and the receive operations have to wait for the finish of the send operations. However, the NI can be programmed to start transmitting data to the network while the host-to-NI DMA transfer is still in progress. In the similar way, the NI-to-host data transfer can be initiated while the network-to-NI data transfer is in progress at the receiver side. By overlapping the network-to-NI(NI-to-network) and NI-to-host(host-to-NI) data transfers, the message transfer is fully pipelined like in the wormhole routing. FMP employs pipelining mode to overlap message transfers, which improves the communication performance impressively.

In the implementation of FMP, a whole message is segmented into several message pieces. When a piece of message is ready to be sent on the sending side, the host or NI sends the message piece by DMA, and on the receiving side, the NI or host can receive the arrived message pieces by DMA before the finish of the whole message send. Whole message transfers are also pipelined, which enables starting a new message transfer before the finish of the previous message transfer.

3.4 Multiplexing and Protection

Since low-level messaging systems often give user applications direct access to the network interface, one process may corrupt the other process's memory in NI. Even with a single process, it may modify the NI control program and crush the whole system. So multiplexing of the NI and protective access to NI memory are necessary for multi-user and multi-process environment.

FMP provides user applications with multiplexing of the NI and protective access to NI memory. For inter-host process communication, several processes in the same host may access the network interface simultaneously. FMP provides protective access to the NI memory. Each process accessing the network is allocated a send buffer and a receive buffer in NI memory for sending and receiving messages respectively. Both buffers are queues, as is shown in Fig. 4. The send and receive buffers of each process are established on initialization and the addresses of both buffers are mapped to the user space. User applications can access the buffers directly, which bypasses the operating system involvement. Since different processes in the same host access separate NI memory areas, the send and receive operations are paralleled and no processes race the NI memory. The Myrinet control program (MCP) is running on LANai in the network interface card. It is responsible for transmitting

the data in send buffers out and putting the incoming messages in the appropriate receive buffer.

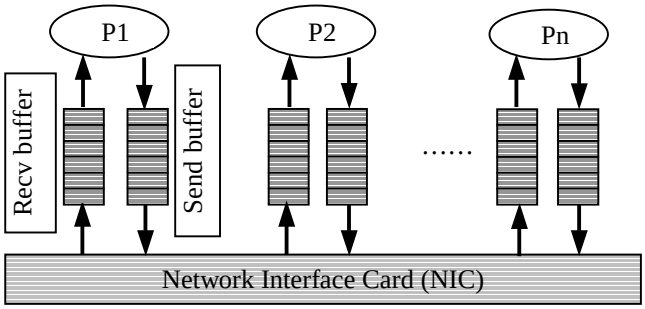

Fig. 4. Message buffers in FMP

The intro-host processes communication is based on the mechanism of shared memory. During the process of system initialization, a block of shared memory is statically allocated for the maxim processes and each process has a send queue and a receive queue. Processes communicate with each other by directly accessing the corresponding buffer queues through process ID.

4 Performance results and evaluation

The validity of our design and implementation of FMP and MPI-FMP/PVM-FMP is evaluated by the communication performance in our PC cluster based on Myrinet. The PCs are connected with a 1.28Gbits/s Myrinet switch. Each PC is a SMP with two Pentium III 550 MHz running Windows NT, and a Myrinet network interface card on PCI Bus.

We measure the inter-host point-to-point communication performance of FMP and MPI-FMP/PVM-FMP, and compare it with that of TCP/IP. All the results of latency and bandwidth are tested by applications directly using send and receive interface functions. The latency tests are in the general roundtrip model and the bandwidth in the stream model. The performance results are illustrated in the following figures.

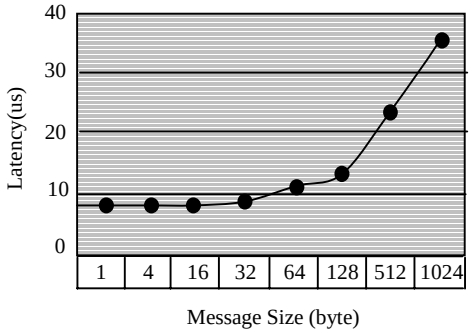

Fig. 5. Latency of FMP

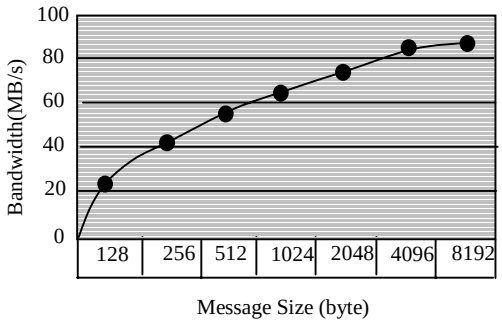

Fig. 6. Bandwidth of FMP

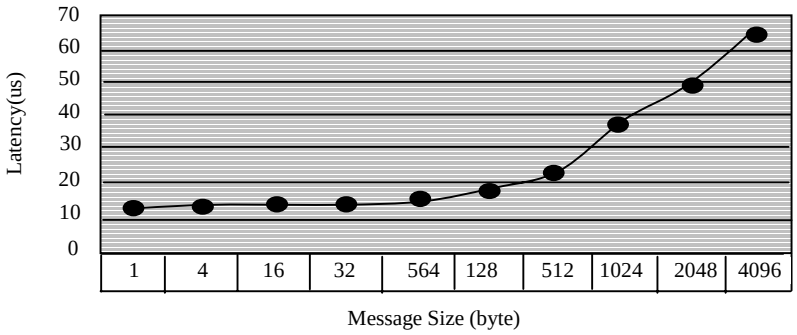

Fig. 7. Latency of MPI-FMP

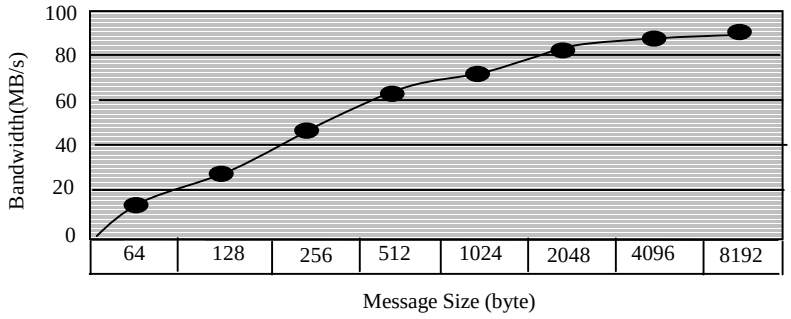

Fig. 8. Bandwidth of MPI-FMP

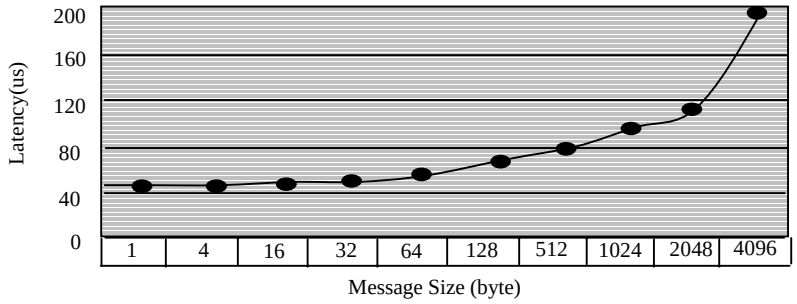

Fig. 9. Latency of PVM-FMP

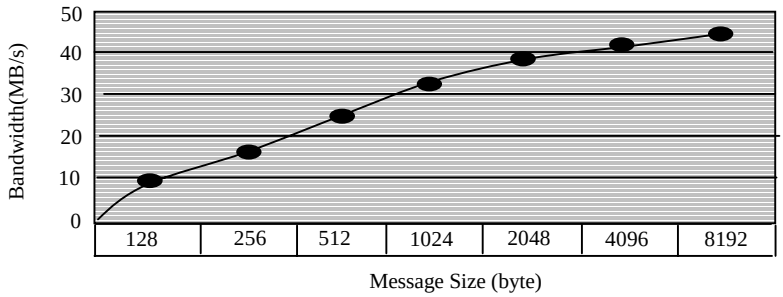

Fig. 10. Bandwidth of PVM-FMP

In our PC cluster platform, FMP achieves a minimum one-way latency of 9us and a peak bandwidth of 83 MB/s. Based on FMP, MPI-FMP achieves a one-way latency of 11us and a bandwidth of 64MB/s, and 45us and 48MB/s for PVM-FMP.

Over the past few years, many research projects have studied the design of high performance communication software, including Active Messages(AM) [8], Fast Messages(FM) [9][10], BIP [11], U-Net [12] and VMMC [13]. These communication systems are all based on Myrinet, and they differ substantially in terms of design consideration and performance. The comparison of the performance of Message Passing Systems in terms of latency and bandwidth is illustrated in Table 1.

Mssage Passing System	One-way Latency(us)	Max. Bandwidth (MB/s)	Configuration
AM-II	21	31	167MHZ UltraSPARC, SBUS
FM 2.1	11	77	200MHZ PPro, PCI
BIP	5	126	200MHZ PPro, PCI
U-Net	13	35	167MHZ UltraSPARC, SBUS
VMMC-2	20	93	166MHZ Pentium, PCI
FMP	9	83	550MHZ Pentium, PCI

Table 1. Comparison of the performance of Message Passing Systems

As we can see from the table, the performance of FMP compares well with other message passing system. Among the message passing systems, BIP stands out for its high performance, which can be explained by its limited functionalities and leaving other functionalities to the upper communication levels.

5 Conclusions and future work

High-speed networks are now providing incredible performance, while traditional communication protocol stacks are not adequate for the high-speed communication hardware. FMP is designed to exploit the performance potential of low-level network equipment.

In this paper, we discuss the general design issues for low-level message passing system and present the design and implementation of FMP. FMP is a high performance message passing protocol for Cluster of PCs connected with Myrinet network.. It achieves impressive communication performance which enables PC clusters to work with high performance of MPP systems.

Based on high-speed and highly reliable communication hardware, FMP reduces messaging layers, eliminates redundant and unnecessary functionalities of TCP/IP stack. The performance results prove the validation of the design of FMP. Although

FMP is designed specially for Myrinet-based PC clusters, the design mechanism is applicable to message passing systems for other network equipments.

Our future work will concern the optimization of higher level API (PVM/MPI) to further reduce the gap between the performance of applications and that of FMP. We'll also make efforts to enrich the functionalities of FMP such as multicast support to provide more communication services for the high-level user applications.

References

1. C.L. Dong, W.M. Zheng, et al., A scalable parallel workstation cluster system, *Proc. of APDC'97*, Shang Hai, China, 1997.
2. Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su, Myrinet: a gigabit-per-second local area network. *IEEE Micro*, February 1995.
3. Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI: The Complete Reference*. MIT Press, 1995
4. Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Mancheck, and Vaidy Sunderam. *PVM:Parallel Virtual Machine*. Scientific and Engineering Computation. MIP Press, 1994.
5. R. Bhyoediang, T. Ruhl, H. E. Bal. Design Issues for User-Level Network Interface Protocols on Myrinet. Technical Report IR-455, Vrije Universiteit, Amsterdam, November 1998.
6. R. dos Santos, R. Bianchini, and C. L. Amorim. A Survey of Messaging Software Issues and Systems for Myrinet-based Clusters. Parallel and Distributed Computing Practices, special issue on High-Performance Computing on Clusters, May 1999.
7. V. Karamcheti and A. Chien. Software Overhead in Messaging Layers: Where Does the Time Go? In Proceedings of ASPLOS-VI, San Jose, California, October 5-7,1994.
8. T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser, Active Messages: A Mechanism for Integrated Communication and Computation, *Proc. of the 19th ISCA*, May 1992.
9. S. Pakin, M. Lauria, and A. Chien. High Performance Messaging on Workstations: Illinois Fast Messges (FM) for Myrinet. In *Proceedings of Supercomputing '95*, December 1995.
10. Mario Lauria, and Andrew Chien. MPI-FM: High Performance MPI on Workstation Clusters, *Journal of Parallel and Distributed Computing*, Jan 1997.
11. L. Prylli and B.Tourancheau. BIP: A New Protocol Designed for High-Performance Networking on Myrinet. In *Proceedings of International Parallel Processing Symposium '98*, March 1998
12. T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A User-Level Network Interface for Parallel and Distributed Computing. In *Proc. 15th ACM SOSP*, Copper Mountain, CO, December 1995.

13. C. Dubnicki, A. Bilas, K. Li, and J. Philbin, Design and Implementation of Virtual Memory-Mapped Communication on Myrinet, *Proceedings of the 1997 International Parallel Processing Symposium*, April 1997.