Flexible User-level Network Interface based on Embedded Processors

James C. Hoe*
MIT Laboratory for Computer Science
jhoe@abp.lcs.mit.edu

1 Introduction

Inadequate network performance of traditional bus-like LAN (Local Area Network) connections presents a performance bottleneck for parallel systems based on a cluster of workstations. The emergence of ATM (Asynchronous Transfer Mode) network, switched Ethernet and various other "switched" networks promises increased point-to-point bandwidth. However, bandwidth alone is not sufficient for exploiting fine-grained parallelization. When communication and synchronization network traffics are short and frequent, the cost of messaging overhead and latency must also be reduced through a suitable network interface design.

To reduce communication latency and overhead, the original StarT project [6] integrated an user-level network interface directly into the pipeline of a RISC microprocessor. However, the time and effort required for such a hardware-intensive project not only introduces great risks but also often results in out-dated hardware with lackadaisical performance. To close this technology gap, on-going projects such as StarT-ng [3] and Flash [4] rely on mainstream commercial microprocessors tightly coupled with external custom messaging hardware (using a direct connection to the processor). However, despite great efforts in optimizing the external hardware and its interaction with the microprocessor, the overall messaging performance remains hindered by the internal design of the microprocessor. Contemporary microprocessors are optimized for cached memory accesses while the I/O architectures are optimized for block transfers. A network interface design should take these biases into consideration and use the available features to its advantage.

FUNi2, Flexible User-level Network Interface, is a second-generation network interface design aimed at providing stock commercial workstation with an cost-effective message-passing interface [5]. FUNi2 follows a different approach to network interface design from the projects described. Instead of trying to reduce communication costs by tight-coupling of *exotic* hardware, FUNi2 attempts to reduce the effects of communication overhead by decoupling the network interface from the processor.

FUNi2 proposes an active network interface design with DMA capability. The user message interface (send and receive message queues) and network buffers are located in the host memory. The CPU and the DMA-capable interface interacts indirectly by reading and writing to the shared interface memory locations. Message passing overhead of the CPU is reduced since long-latency memory-mapped I/O accesses are not used to manipulate the network interface. Instead, the CPU uses normal memory accesses which the processor and cache are optimized for. Furthermore, the user process can enqueue and dequeue outbound and inbound messages at its own rate independent from the bandwidth that is available to FUNi2. Moving the network buffers into the user memory space allows much greater buffering capacity than in dedicated hardware. This provides the buffer size necessary to tolerate the irregular network traffic on a highly distributed parallel system.

*This research is supervised by Professor Arvind and is supported in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-92-J-1310 and by an Office of Naval Research Graduate Fellowship.

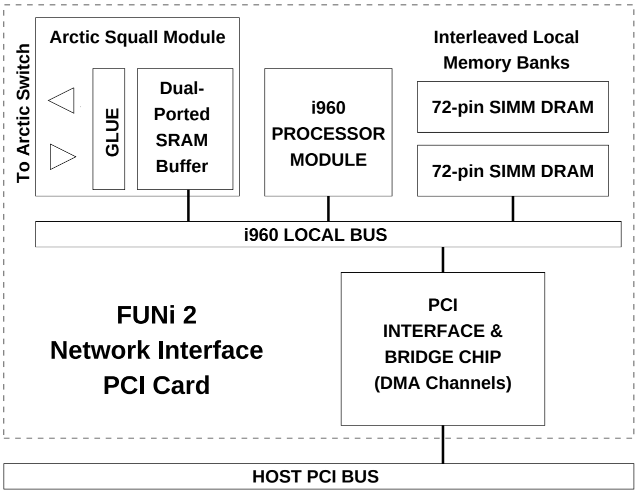

Figure 1: FUNi2 Network Interface PCI Card: Datapath

To further reduce the communication overhead of the CPU, FUNi2 proposes an *intelligent* network interface to off-load some of the message processing from the CPU. For example, remote memory accesses could be performed at the remote site by the network interface directly, without disrupting the remote CPU. Processing of light-weight Active Messages is another candidate that could be off-loaded from the CPU to the network interface. FUNi2 will provide this level of intelligent network/message services by incorporating an embedded network processor. Aside from performance, the embedded network processor has the additional advantage of greatly simplifying the hardware design effort of the FUNi2 project. The reprogrammable firmware can be easily revised to allow experimentation with different FUNi2 designs.

This paper proposes the FUNi2 project. This section gives the motivation for the project and presents an introduction to the FUNi2 concept. The next section discusses the technology for implementing FUNi2. The final section outlines the potential research stemming from the FUNi2 project.

2 Enabling Technology

Initial FUNi2 implementation will provide PCI-equipped workstations with an interface to MIT's custom Arctic Network [2]. This implementation tries to avoid costly hardware design and realization. FUNi2 will be an I/O card-based design that leverages heavily on existing technologies. Figure 1 illustrates the datapath of the FUNi2 PCI card.

2.1 Cyclone PCI80960 Card

The FUNi2 card will be based on Cyclone Microsystem's PCI80960 *Intelligent Communication Controller*. The card houses an Intel i960-based embedded system with a PCI bridge chip to interface with the host system. The card can accept a variety of i960 processor modules; FUNi2 will use a 40MHz Superscalar i960CF 32-bit RISC processor to carry out the network processing. The FUNi2 card can be configured with up to 32 MBytes of interleaved local DRAM memory. To interact with the host processor, the FUNi2 embedded processor can access the host memory through two independent bidirectional DMA channels with a peak transfer rate of up to 132 MBytes/second.

2.2 Custom Arctic Squall Module

The PCI80960's i960 local bus provides the interface to an inter-changeable I/O expansion module to implement the physical interface to various types of networks. *Squall II* I/O modules for Ethernet, SCSI, etc. are commercially available from Cyclone. The modular nature of the physical interface allows FUNi2 to adapt readily to other suitable network technologies. Initially, the FUNi2 project will build a custom Squall module to interface with Arctic. This custom module will share many of its components with the StarT-ng's NIU (Network Interface Unit) design.

2.3 Arctic Switch Fabric

FUNi2 will leverage on the interconnection technology developed for the StarT-ng project. This common interconnect encourages the possibility for heterogeneous operations of FUNi2-equipped workstations alongside with StarT-ng nodes.

Arctic is a MPP-class packet-switched router designed to implement fat-tree and other multi-stage networks. Each Arctic router provides a full cross-bar between 4 input ports and 4 output ports with 6 cycles (40MHz) of flow-through latency. A peak bandwidth of 160 MBytes/sec is available at each port. Two network traffic priorities are supported by Arctic with the guarantee that low-priority packets will never block high-priority packets. This feature is crucial to the flow-control protocol that FUNi2 uses.

3 Proposed Research

The goal of the FUNi2 project is to achieve program performance comparable to hardware-intensive projects such as StarT-ng or Flash, while maintaining a significant price-performance advantage. The research focuses on providing the correct memory-based interface paradigm. The research will take full advantage of the programmable network processor to compensate for the other hardware tradeoffs.

3.1 Intelligent Message Interface Service

By decoupling the network interface from the CPU, FUNi2 is able to reduce the overhead of message passing, but the latency of communication will increase. The success of FUNi2 must rely on the application of latency-hiding software techniques such as split-phased transaction. Part of the proposed software research involves analyzing existing programs/algorithms' communication pattern and transform such programs/algorithms into more latency tolerant ones. The analysis of communication patterns will also help in designing more suitable message interface services. By using the programmable FUNi2, the services provided can be fine-tuned on a program-by-program basis.

More intelligent network services will be developed as part of the research. In addition to the basic message sending and receiving functions, various higher-level features can be supported by the firmware programmable FUNi2 to off-load network processing tasks from the host processor. (FUNi2 firmware is dynamically loadable by the host processor.) Processing of light-weight Active Messages and remote fetches are obvious candidates. Other possibilities also include scheduled prefetches, conditional prefetches and advanced synchronizations services.

3.2 Flow-control and Fault-Tolerance Protocol

FUNi2's embedded processor-based implementation encourages research in network flow control protocols. Even non-trivial protocols can be quickly tested by recoding the firmware. Performance monitors can also be included with little effort. Initially, FUNi2 will inherit the Acknowledgment/Retry end-to-end flow control protocol proposed for the original FUNi. When FUNi absorbs an inbound packet from the network, it returns an acknowledgment packet to the originating FUNi. If FUNi cannot accept a packet, a negative acknowledgment is returned to request a retry. The protocol is useful in systems where the network is shared between multiple users since the protocol can guarantee both deadlock-freedom and worst-case performance of the network. Future work in protocols can extend to address fault tolerance and network load balancing issues.

3.3 Application to StarT-ng Project

Successful results from this project can be applied directly to the StarT-ng system. Currently, each StarT-ng SMP site incorporates 4 NES (Network Endpoint Subsystem) boards; each board contains a PPC620 CPU and a private network connection. To support the FUNi-style of communication, only one NES is required on each site to serve the role of network processor. The three other NES boards can be replaced by normal CPU boards to increase the computation power of each site and also allow larger StarT-ng system to be constructed using the same switch fabric technology.

4 Related Work

The original StarT system dedicates a second processor to handle network processing in order to minimize disturbance to the main processor. Meiko CS-2 MPP systems also uses a separate processor to process network communications. The SHRIMP multicomputer project [1] specifies an user-level network interface that involves the memory system to reduce the communication overhead in a stock commercial system.

References

[1] M. A. Blumrich, et. al. Virtual memory mapped network interface for the SHRIMP multicomputer. In *Proceedings of 21st ISCA*, April 1994.

[2] G. A. Boughton. Arctic routing chip. In *Proceedings of Hot Interconnects II*, August 1994.

[3] D. Chiou, et. al. StarT-NG: Delivering seamless parallel computing. Technical Report CSG Memo 371, MIT Laboratory for Computer Science, February 1995.

[4] J. Heinlein, et. al. Integration of message passing and shared memory in the Stanford FLASH multiprocessor. In *Proceedings of ASPLOS*, March 1994.

[5] J. C. Hoe. Network interface for message-passing parallel computation on a workstation cluster. In *Proceedings of Hot Interconnects II*, August 1994.

[6] R. S. Nikhil, G. M. Papadopoulos, and Arvind. *T: A multithreaded massively parallel architecture. In *Proceedings of 19th ISCA*, May 1992.