ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S08NC02

Multi-processor Network Implementations in Multibus II and VME

Charlie BRIEGEL, Fermilab†
P.O. Box 500, Batavia, IL 60510, USA

Abstract

ACNET (Fermilab Accelerator Controls Network), a proprietary network protocol, is implemented in a multi-processor configuration for both Multibus II and VME. The implementations are contrasted by the bus protocol and software design goals.

The Multibus II implementation provides for multiple processors running a duplicate set of tasks on each processor. For a network connected task, messages are distributed by a network round-robin scheduler. Further, messages can be stopped, continued, or re-routed for each task by user-callable commands.

The VME implementation provides for multiple processors running one task across all processors. The process can either be fixed to a particular processor or dynamically allocated to an available processor depending on the scheduling algorithm of the multi-processing operating system.

I. INTRODUCTION

The motivation for a multi-processor platform in Fermilab's Accelerator Controls Group was to support our extensive commitment to CAMAC. The goal was to provide a replacement for PDP11 front-ends improve the effective utilization of the CAMAC serial link. Since CAMAC can have only one master, the link hardware allows ownership to be passed between cooperating processors. One of the requirements for this configuration was to implement ACNET communications for several duplicate processors running identical set of tasks. Thus, processors can be transparently added to the configuration with a corresponding increase in performance. This requirement provided the impetus for a multi-processor network connection to the existing network. The VME implementation provides support of multi-processor networks by using MTOS-UX MP (multi-processor version) operating system for transparent distribution of tasks among a set of processors.

† *Operated by Universities Research Association for the Department of Energy.*

II. ACNET OVERVIEW

ACNET (Accelerator Controls Network) is Fermilab's proprietary network protocol implemented in 1980. It consists of both a software protocol and a calling sequence specification.

The software protocol consists of a 9 word header (figure 1) preceding each message and provides a specification for the routing of messages between tasks. The protocol maintains a connection between cooperating tasks through a status reply and/or cancel messages. These notifications enable tasks and the network to maintain connectivity and cleanup network resources with minimal overhead.

The calling sequence provides a consistent user interface across heterogeneous machines and enables a request/reply paradigm for communication. Both asynchronous communications (traps, signals, or event flags) and synchronous communications (polling, wait, or wait with time-out) are supported by the calling sequence.

This calling sequence has enabled Fermilab to isolate effects to users due to changes in either the software or hardware protocol. While the implementation imposes inherent software costs, there is an advantage in providing a consistent layered approach for other software protocols.

The proprietary protocol does not restrict the use of standard protocols. For instance, by tunneling or encapsulating software protocols, ACNET has been implemented through a DECNET protocol and will be implemented with TCP/IP in the near future. The primary reason for not implementing a standard protocol stack has been the lack of support by vendors for the current set of heterogeneous processors and operating systems at Fermilab.

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S08NC02

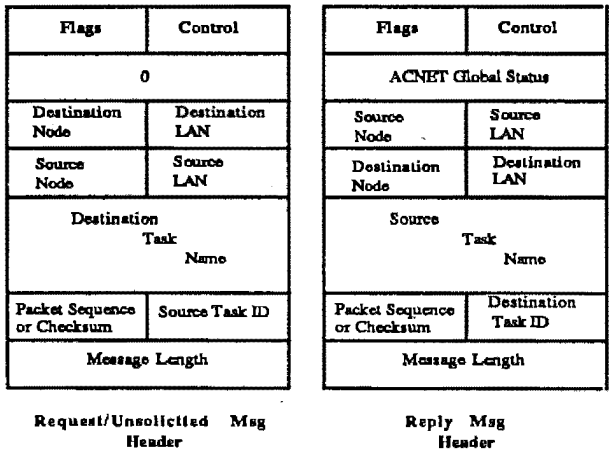

Figure 1

III. MULTIBUS II

A. Configuration

The Multibus II bus with multiple Intel 386s running MTOS-UX SP (single processor version of the operating system) is the platform for this implementation. The crate contains an IBM/PC as the bus monitor which supports diagnostics, rebooting, and console emulation. Each of the Intel 386 processors has a daughter board supporting the CAMAC link directly.

Included in the configuration is a Token Ring board for communications. The board has an 8 MHz Intel 186 with a Token Ring daughter board attached to the ADMA controller of the processor board. The board implements the TMS380C16/04 chipset at either 4 or 16 Mbps and will be commercially available in the near future.

B. Software Implementation

To effectively communicate across a Multibus II backplane, each board contains a message passing co-processor and a software protocol called "transport" to provide a messaging facility similar to a LAN protocol. The Token Ring board's software takes an incoming transport message and synchronously transmits it onto the Token Ring. Upon receiving a Token Ring frame, a transport message containing the frame is sent to the appropriate processor. The distribution of a message is based on a connected task name for a given processor, the node number, and the message type.

ACNET requests are delivered in round-robin order. When a task called 'TSKA' connects to the network from CPU1, a transport message provides this information to the Token Ring processor. Another processor, CPU2, can connect with the same task name, 'TSKA'. Since replies must go to the originating request's processor, only requests are distributed in this way. The ACNET header's source node is altered to remember the specific processor which initiated the request so the reply is routed correctly.

ACNET header contains a 16-bit word to specify a lan/node number for both the destination and the source of the message. For this implementation, a set of node numbers are used as a logical nodes. One node number is used as a global logical node and implies the request can be distributed to any of the available processors which have the connected task specified in the ACNET header. For each processor, a specific logical node enables a request to be directed to the corresponding processor.

When a task is terminated, a cancel message is sent to delete each outstanding request. Since the requests are distributed in round-robin order, the Token Ring processor needs to deliver this cancel to the same processor which received the original request. Since cancels are infrequent and can be uniquely matched to original request through the ACNET header, the Token Ring processor broadcasts the message to all processors.

The byte orientation on the Token Ring wire was dictated by early implementations on VME, UNIBUS and QBUS platforms. The TMS380 chipset enables either Motorola or Intel logic interface which implies correct text, while integers are byte swapped. These early controllers used the Motorola interface and mapped the Big Endian format directly to the little endian processor bus. The result was an automatic byte swap. While this appeared to optimize our message format, the Multibus II Token Ring board using an Intel interface must byte swap each message.

The ACNET software for Multibus II and VME processors is written in "C" and is ported between the two platforms. The portable code is conditionally compiled to distinguish the use of transport as a vehicle to and from the Token Ring board.

C. ACNET extensions

Three new calls were added to aid in flow control. The round-robin scheduling of requests could be made ineffective if requests with greater life-times funneled into the same processor. The processor could be starved by servicing these requests with multiple replies while other processors would be relatively idle.

If such a case can be detected by the user, a NTSTOP call can be executed which provides a temporary block for future requests to be delivered to this task's processor. A subsequent NTUSTP call will resume normal round-robin schedule.

Since Multibus II implements a flexible messaging facility, a request can be re-routed to another processor based

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S08NC02

on a user statistic, memory utilization, or idle time. The user can execute the NTMREQ call which will re-route the message to a specified processor. It is the user's responsibility to prevent a message storm on the bus.

IV. VME

A. Configuration

The test configuration used two Motorola 68020s running MTOS-UX MP operating system in a VME backplane. The platform implements one of the following three Token Ring boards: a Fermilab designed board at 4 Mbps, a Proteon p1542 at 4 Mbps, or a Formation fv1600 at 4/16 Mbps. All of the boards use the TMS380 family of Token Ring chipsets.

B. Software Implementation

The same ACNET software package implemented for MTOS-UX SP is used for the MP version. The operating system allows multiple tasks to run concurrently and transparently on multiple processors. A copy of the operating system is placed on each processor to improve performance. The operating system allows the user to specify the processor a task must run on or whether the task can be globally run on any of the available processors. If the task is global, the user must make modifiable data accessible to all processors. In the case of the test configuration, the operating system's tables, global task's data, and stack was contained in a shared memory segment accessed through the VME bus. Thus, performance is decreased due to the multi-processing operating system's overhead and each global task's data access over the VME bus.

V. PERFORMANCE

In a multi-tasking operating system, many factors determine the performance of the network to deliver messages to a task for useful work. Each of these implementations provide adequate performance with several opportunities for improvements. The Multibus II configuration can deliver 200 byte messages at approximately 110 messages per second for one processor and 180 messages per second for two processors. This measurement is for a task communicating with itself and in the case of two processors the messages are distributed to two duplicate tasks on separate processors.

The VME configuration was tested with one global task initiating a 200 byte request to a separate global task in the same VME backplane. The receiver of the request then sent an echoed reply of equal length. A single processor running MTOS-UX SP generated approximately 200 messages per second. A single processor running MTOS-UX MP generated approximately 160 messages per second. A dual processor running MTOS-UX MP generated approximately 170 messages per second. As expected, the additional processors should not improve network performance in such an architecture, but could provide additional compute power with ACNET communications.

The VME test was implemented with the ACNET network task running on the first processor. The same test was attempted with ACNET running as a global task on two processors. The result generated approximately 8 messages per second. The second processor had the Token Ring interrupts masked off. I believe the tasks were thrashing from processor to processor, but interrupt delivery to a task could only be delivered to the same processor which received the hardware interrupt. While further investigation is required, the results caution the user of MP to understand the system architecture or unusual results could occur.

Improvements in performance can be implemented with the following techniques:

1. Implement multiple Token Ring controllers.
2. Upgrade processors.
3. Overlap Token Ring I/O with transport I/O.
4. Eliminate unnecessary copy of received frames.
5. Optimize C code.
6. Upgrade the Token Ring to 16 Mbps.
7. Upgrade the TMS380C16/04 firmware.

VI. CONCLUSIONS

The Multibus II implementation has been running at Fermilab's Central Detector Facility. While the system has limited utilization compared to Fermilab Accelerator Controls, it has been valuable for detecting bugs and predicting performance characteristics. In particular, the round-robin scheduling with user invoked flow control appears to be adequate and manageable.

The throughput of the Token Ring board is a concern. Since the Token Ring software must byte swap each message, much of the performance is tied to the processor. The board will soon be available with a 16 MHz Intel 186 and should improve performance. Further, the software to enable multiple Token Ring boards on the same Multibus II backplane is easily implemented. Unfortunately, the ability to allocate these resources for incoming messages is a management problem without a stable solution.

Currently, their are no multi-processor implementations of MTOS-UX MP. While the operating system provides a high degree of transparency, the improvement in performance is difficult to quantify. In general, a master/slave or co-processor configuration with simple mailboxes is more predictable and manageable for real-time processing.

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S08NC02

While both implementations use MTOS-UX as its operating system, the multi-processor version is inefficient for Multibus II. The operating system's global tables implies a time consuming access in Multibus II. The Multibus II implementation does enable tasks to be statically distributed across multiple processors. A VME implementation to enable duplicate tasks distributed across multiple processors requires mutual exclusion techniques on network structures and additional software. Thus, these two multi-processor network implementations provides a non-competitive solution solving two different problems.

VII. ACKNOWLEDGEMENTS

Multibus II Token Ring board was designed and built by Jim Zagel, John Smolucha, and Bob Marquart. Transport protocol for the Intel 186 was developed by John Smolucha and transport protocol for MTOS-UX was developed by Mike Glass and Bill Marsh. Kevin McGuire and Duane Voy (SSC) generated MTOS-UX MP for the MVME133 board.

VIII. REFERENCES

[1] Briegel, C., Johnson G., and Winterowd, L. (1989). The Fermilab ACNET Upgrade, Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems, pp. 235-238

[2] Glass, M., Zagel., J., Smith, P., Marsh, W., and Smolucha, J., (1989). The Upgraded Tevatron Front End, Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems, pp. 87-92