Fermi National Accelerator Laboratory

FERMILAB-Conf-89/133

Uniform Communications Software Using TCP/IP *

Mark Bernett and Gene Oleynik
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510 U.S.A.

May 1989

* Presented at "Real Time Computer Applications in Nuclear, Particle, and Plasma Physics," Williamsburg, Virginia, May 16-19, 1989.

Operated by Universities Research Association, Inc., under contract with the United States Department of Energy

UNIFORM COMMUNICATIONS SOFTWARE USING TCP/IP[1]

Mark Bernett, Gene Oleynik
Online and Data Acquisition Software Groups
Fermi National Accelerator Laboratory
Batavia, IL 60510

Abstract

Data acquisition applications at Fermilab require a reliable, distributed communication system for downloading, diagnostics, control, and data distribution . TCP/IP over Ethernet[2] was chosen because of its uniform user interface and commercial availability for a number of processors and operating systems. This paper describes our software and hardware support for TCP/IP on VAX/VMS[3], VME/pSOS[4], FASTBUS/pSOS, and Unix systems. It includes plans to provide a portable, hardware independent implementation of TCP/IP based on Berkeley BSD software.

I. INTRODUCTION

Experiments at Fermilab are experiencing a virtual explosion in the number, power, and variety of processors and intelligent hardware devices available for their online systems. The PAN-DA data-acquisition currently under development will integrate a number of these processors and hardware devices. In particular, FASTBUS will be used for data acquisition, VME based Motorola and ACP processors for VME control, event buffering and filtering, VME based 8mm EXABYTE[5] tape technology for logging events, and VAX/VMS or Unix based processors for overall system control [1,2,3,4,5,6].

A uniform system for communication between these devices is needed for system control, program downloading and initialization, program snapshot uploading for debugging/crash analysis, and event distribution for monitoring. In addition, the PAN-DA system design includes a Remote Procedure eXecution (RPX) package that requires a base communications protocol that can be supported on all systems that RPX will run on [7].

We have selected Ethernet based TCP/IP as a major part of the PAN-DA communications backbone; other communications systems, such as OSI, were deemed to be too much in their infancy for our needs. TCP/IP is commercially available on a wide variety of systems. Most systems have a uniform "socket" user interface, for which we can provide portable software. TCP is a reliable protocol, and the 10 Mbits/sec Ethernet speed is adequate for our needs. A suite of "standard" applications software such as FTP, TELNET, and essentially public domain software that can be used to port TCP/IP to new systems is available. IP based protocols can coexist with other protocols on the Ethernet, such as DECnet, so the already existing Ethernet network at Fermilab can be used. A typical configuration at an experiment is shown in figures 1 and 2.

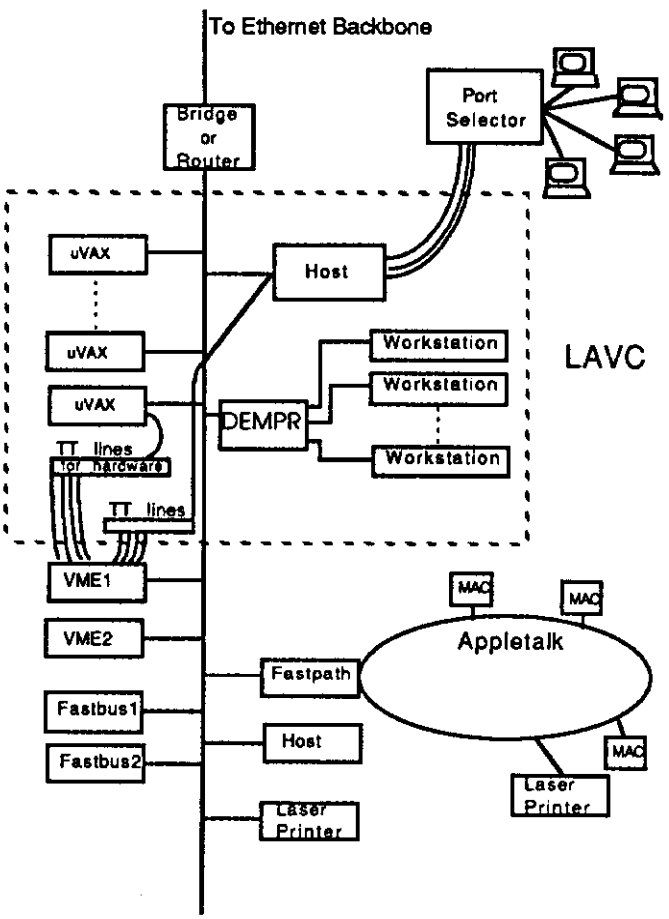

Figure 1: Typical experiment network

We use commercially available TCP/IP hardware and software where available as platforms for our applications.

---

1 (*)sponsored by DOE Contract No.DE-AC02-76CH03000

2 Ethernet is a registered trademark of the Xerox Corporation.

3 VAX, VMS, MicroVAX, and DEC are trademarks of the Digital Equipment Corporation.

4 pSOS is a trademark of Software Components Group, Inc.

5 EXABYTE is a registered trademark of the EXABYTE Corporation.

This includes the SRI MultiNet[1] software package for VMS systems, Communication Machinery Corporations' (CMC) ENP-10[2] Ethernet and TCP/IP processors for VME, and modified CMC software to drive the ENP-10s. TCP/IP is already an integral part of Unix systems.

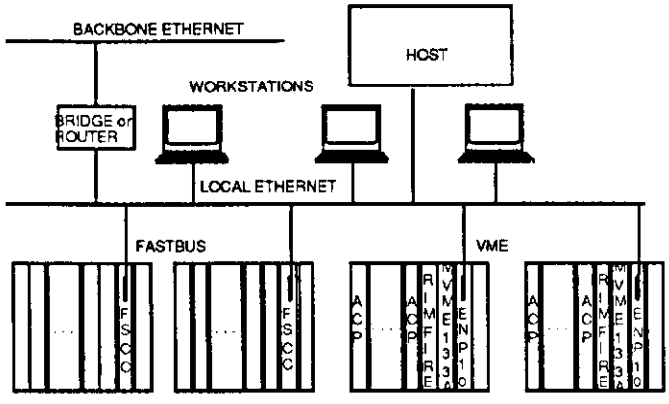

Figure 2: Data acquisition portion of network

We are now developing new 68K based intelligent front-end modules in FASTBUS which incorporate an Ethernet interface. Since no TCP/IP software is commercially available for these modules we are developing a portable implementation in-house.

II. EXISTING SYSTEMS

A. VAX/VMS

The SRI International MultiNet TCP/IP package allows VMS hosts to communicate with other TCP/IP nodes. This package provides most of the "standard" TCP and UDP services, and provides a Unix like socket interface library as well as a VMS QIO interface. This software, along with some extensions added by us, is running on a number of VMS nodes at the Fermilab site.

The MultiNet package permits users to write their own TCP or UDP clients and servers [8]. User written server processes using the socket interface can be registered with a Master Server process, which will start a detached version of the process whenever the service is requested. This allows services to be provided asynchronously, so that multiple requests can be handled simultaneously. Alternatively, asynchronous service can be provided if the server is programmed at the QIO level to provide an AST thread for each instance of a service. This last technique is more natural for a VMS system, and was used to program our VMS based Remote Procedure eXecution servers.

Besides providing a base for the RPX package on VMS machines, the MultiNet software will be used by programs running on workstations which will collect event data from VME for online monitoring purposes. Future plans also include support for downloading programs to online system processors.

TCP/IP throughput from a MicroVAX II to a CMC ENP-10 saturated the MicroVAX CPU at 140 Kbytes/sec (The ENP-10 is capable of 188 Kbytes/sec, see below).

B. VME

One of the major reasons we chose TCP/IP was the availability of a low cost TCP/IP processing VME module along with software capable of driving it from a VME processor running pSOS. We had already chosen the pSOS multi-tasking operating system kernel for PAN-DA VME modules [3].

The ENP-10 Ethernet controller board is a high performance, intelligent, VME based interface to the Ethernet. A host VME processor residing on the same VMEbus can access the Ethernet via the ENP-10. The ENP-10's 68010 processor, Ethernet processing LANCE chip set, and 512 Kbytes of DRAM are used to offload network protocol processing from the host processor, freeing it to do other work. In particular, TCP/IP processing is performed on the ENP-10 with the optional TCP/IP firmware. The firmware supports the entire Internet suite of protocols: TCP, UDP, ARP, ICMP, and raw Ethernet.

We have extensively modified a pSOS driver provided by CMC to drive the ENP-10 from a single Motorola MVME133A host processor in the same crate. pSOS processes on the MVME133 are able to communicate with other systems on the same Ethernet through a Unix style socket library [8] (the original CMC driver was a Unix driver modified for pSOS). This library contains Microtec[3] object modules for use in the Microtec cross-development environment that we have extended for PAN-DA [3].

Client and server processes are written with the same routine calls used with the socket interface on the VMS MultiNet system. Services are provided asynchronously by pSOS server processes which spawn processes for each instance of the service.

TCP/IP provides one of the communications media for PAN-DA RPX applications split between host VMS or Unix systems and VME processors, and will also be used to distribute events buffered in VME to workstations for analysis and monitoring. Future plans include TCP/IP

[1] MultiNet is a registered trademark of TGV, Inc.

[2] ENP and CMC are trademarks of the Communication Machinery Corporation.

[3] Microtec is a registered trademark of Microtec Research, Inc.

support for downloading programs and uploading system snapshots (as part of the PAN-DA static debugging tool [3]). Since the MVME133A/pSOS environment is non-interactive and diskless, TCP/IP and UDP services such as TELNET and FTP are not supported

TCP/IP throughput has been measured at 188 Kbytes/sec between two ENP-10s on an isolated Ethernet. The CPU overhead on the MVME133A-20 hosts controlling the ENP-10s was measured to be approximately 1.1 us per transferred byte.

III. A PORTABLE ETHERNET TCP/IP IMPLEMENTATION (PRETI)

Fermilab is developing in-house front end embedded processors for data acquisition systems which will need communication capabilities for program download, debugging, control, and a secondary data path. RS-232, used for this purpose previously, is too slow for the current technology of modules being developed. Commercial communications software is not available for a new module. The time required to get a complex TCP/IP package ported and working on a new module is considerable. We decided to address these issues by adapting an existing implementation to explicitly make it easy to port to a new operating system and hardware environment. This is being done initially in support of the Fermilab FASTBUS Smart Crate Controller, a new high speed readout controller.

The communications package is being adapted from a public domain version of TCP/IP software developed by the University of California at Berkley (UCB). Many commercial communication packages are based on this software. Using the UCB version offers a very good chance for compatibility with other software on processors on the far side of the Ethernet. The system will provide communication services using TCP, UDP, IP , ARP, and Raw Ethernet protocols. Figure 3 shows a block diagram of the primary system components.

In support of the goals of providing as much hardware and software independence as possible the software will:

- o Be position independent so that it can accommodate any specific system address map.
- o Be usable from read-only memory.
- o Easily accommodate different Ethernet controller chips.
- o Be usable on different or no operating systems. This is particularly useful when one wishes to download an operating system where none is previously active.
- o Be layered to allow offloading of the lower level processing on to a separate CPU.
- o Easily be adapted to add a new protocol. In the future there will be a migration to ISO/OSI communication protocols. This migration should be able to be accommodated with minimal effort.

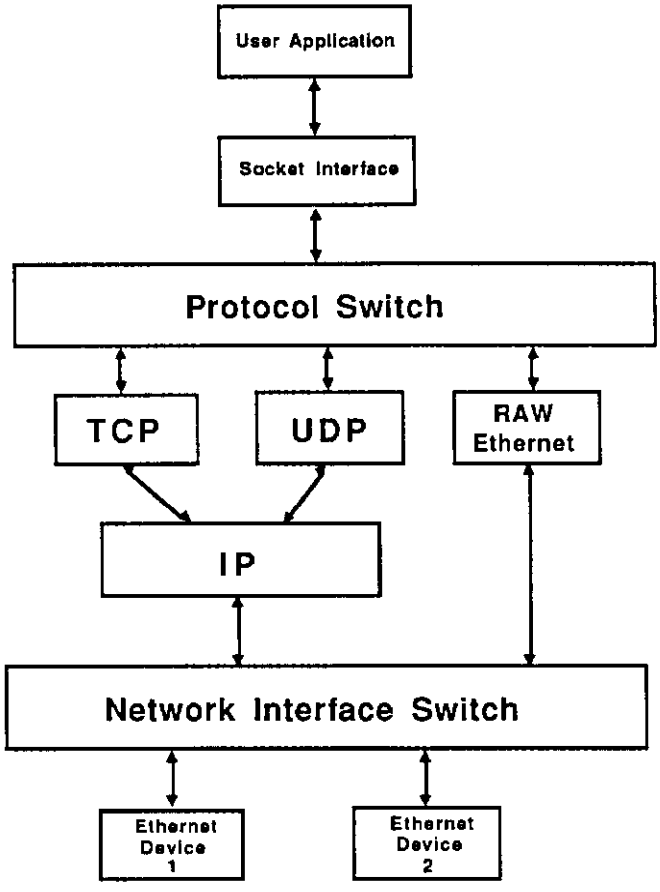

Figure 3: Major elements of PRETI

Ease of porting PRETI to new modules is achieved by implementing system dependent parts of the TCP/IP code as "callout functions" and "callable services". For the former the system independent routines make well defined subroutine calls to perform each system dependent function. For the latter the system dependent code (e.g. interrupt service routines) make subroutine calls to well defined routines in the PRETI kernel.

The interface between the communications system and the specific hardware of the module to which the package is being ported is implemented with user written software. The design of the PRETI implementation results in these routines being small in number and size and independent of the main body of the code.

The user supplied software handles the following functions: process blocking (e.g. when a process waits for a read to complete); memory allocation /deallocation;

interrupt enabling /disabling; timer services; obtaining the Ethernet/Internet address; event tracing and diagnostic message handling; statistics; mutual exclusion; initialization. Often more than one callout is required to implement a particular function. Process blocking, for example, is implemented as two callouts which will include system dependent code: one to block the process and the other to release it. Depending on the services available under the specific operating system, the blocking callouts might be implemented as: wait on an event flag / set an event flag; wait on a mailbox / send a message to the mailbox; spin loop wait on a memory location to be cleared / clearing the memory location. This last method can be used even where no operating system exists.

An initialization callout is provided in which the user can initialize any parameters and static memory needed by the other callout routines.

A UCB 4.2 BSD socket interface is provided as part of PRETI for the application programmer. We are implementing this to provide a common user interface between this package and the other TCP/IP packages that we use.

IV ACKNOWLEDGEMENTS

We appreciate the collaborative professional support and help of Greg Chartrand, Phil Demar, and Cary Dowat from the Computing Department's Communication Group.

V REFERENCES

[1] See accompanying paper: "Software for FASTBUS and Motorola 68K Based Readout Controllers for Data Acquisition", Ruth Pordes, et al.

[2] See accompanying paper: "The PAN-DA Data Acquisition System", D. Petravick, et al.

[3] See accompanying paper: "A Real Time Integrated Environment for Motorola 680xx-based VME and FASTBUS Modules", David Berg, et al.

[4] See accompanying paper: "FEREAD - Front End Readout Software for the Fermilab PAN-DA Data Acquisition System", Terry Dorries et al.

[5] See accompanying paper: "Software for EXABYTE Helical Scan Devices", Penelope Constanta-Fanourakis, et al.

[6] See accompanying paper: "ACP/R3000 Processors in Data Acquisition Systems", J. Deppe, et al.

[7] See accompanying paper: "Remote Procedure Execution Software for Distributed Systems", Eileen Berman et al.

[8] M. Bernett and Gene Oleynik, "TCP/IP socket level programmers guide for the PANDA system using the SRI_MULTINET and ENP10 products", Fermilab Computing Department Note PN-370.