# NETWORK UPGRADE FOR THE SLC: CONTROL SYSTEM MODIFICATIONS

M. Crane, R. MacKenzie, R. Sass, T. Himel
Stanford Linear Accelerator Center, Stanford University, Stanford CA 94309

*Abstract*

Current communications between the SLAC Linear Collider control system central host and the SLCmicros is built upon the SLAC developed SLCNET communication hardware and protocols. We will describe how the Internet Suite of protocols (TCP/IP) are used to replace the SLCNET protocol interface. The major communication pathways and their individual requirements are described. A proxy server is used to reduce the number of total system TCP/IP connections. The SLCmicros were upgraded to use Ethernet and TCP/IP as well as SLCNET. Design choices and implementation experiences are addressed.

## I. CURRENT SLC CONTROL SYSTEM

The current SLC Control System micro communication services are based on the SLCNET communications protocol. The SLCNET system was developed at SLAC in the 1980's and consists of a master node and up to 255 slave nodes connected by custom hardware and software. The master, or VMS central host is a DEC Alpha computer running the OpenVMS operating system. The slaves, or SLCmicros, are Intel ix86 single board computers in Multibus I crates running the iRMX real time operating system. The SLCNET clock rate is one megahertz and the protocol used is SDLC.

Message passing is supported between individual processes on the VMS host to individual tasks on the SLCmicros. The most prevalent VMS host process is the SLC Control Program or (SCP), of which there can be many running concurrently. The SCP provides the main user interface to the SLC Control System. There are also stand-alone processes which communicate directly to the SLCmicros to collect error reports, distribute database and timing information. All together this involves 50-100 processes on the VMS host communicating with 1-10 tasks on the SLCmicros. The basic tools to boot, debug, and query diagnostic information from the SLCmicros are also available over SLCNET.

The VMS host is equipped with Ethernet and FDDI interfaces which provides support for TCP/IP. The VMS host TCP/IP stack is Multinet which provides communications to various network nodes used in the control system such as EPICS nodes, X-terminals and others. The SLCmicros do not presently support Ethernet or TCP/IP.

## II. ENTER ETHERNET AND TCP/IP

The decision to upgrade the SLC Control System VMS host and SLCmicros to use TCP/IP protocols was based on projected bandwidth requirements, hardware availability, and maintenance issues. Please refer to the network upgrade paper for more information[1]. The upgrade of the SLC Control System and the SLCmicros would make TCP/IP the common communications protocol for all PEP-II control system nodes.

This upgrade project required that code be modified or written for both the VMS host and the SLCmicros. This code supports both SLCNET and TCP/IP communications paths since we plan to upgrade as schedule and budget constraints permit. There are some applications that require significant modifications and others that require complete new software to use TCP/IP.

The TCP stream-based protocol was chosen for nearly all the applications since a reliable data stream with flow control is required. TCP connections are setup once and remain for the lifetime of the SLCmicro task or VMS host process. This reduces connection overhead. Since the VMS host and the SLCmicros enjoy the same byte alignment, the data will be passed in little-endian order.

The Multinet package on the VMS host supports both TCP/IP BSD3.4 socket interface and the standard VMS QIO interface. The SLCmicro uses the Fusion TCP/IP stack ported to the iRMX operating system

## III. A PROXY SERVER

On the VMS host there can be 50 SCPs plus 10-20 stand-alones running concurrently while there are about 70 SLCmicros in the system. Using direct TCP connections would require 2000-3000 connected sockets on the VMS host. This number of connections would consume to many system resources. The problem was solved by implementing a proxy server that provides data flow between the VMS host and the SLCmicros. The proxy server has two connections to each SLCmicro and a single connection to each VMS host processes communicating to the SLCmicros. This limits the number of VMS host TCP connections to just the number of communicating processes.

The proxy server works by appending a destination header to the users data stream. This header has routing information which allows the proxy server to forward the data transfer to the correct destination. Proxy server 'KEEPALIVE' is also supported to allow an application to check that the proxy server connection is complete and the

*Work supported by Department of Energy, contract DE-AC03-76SF00515

0-7803-4376-X/98/$10.00 © 1998 IEEE

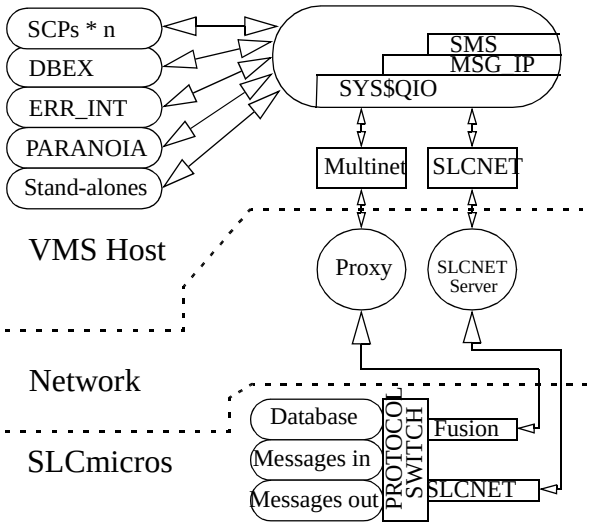


*Figure 1. SLC Control System Micro Data Communications*

proxy server is operational.

## IV. MESSAGE SERVICE

The SLC Message Service (SMS) is a message service which securely delivers messages between the VMS host and SLCmicros. It is a design goal to have TCP/IP communications fit into the existing message service in a transparent manner. Meeting this design goal means making TCP behave like SLCNET (from the user's perspective). This is tricky because SLCNET is a reliable, connectionless, datagram protocol unlike TCP which is stream based.

SMS works in a command/response mode. A VMS host computer process issues a command and one or more SLCmicros may optionally respond. A command is sent to SLCmicro(s) by a host computer process (e.g. a SCP) and the process then optionally waits for a reply.

Each SCP process on the VMS host is a "Single-Process Client" which makes an active connection request to a server. SMS supports any mixture of direct connections to SLCmicros and connections via proxy servers.

The TCP/IP communications code is implemented as a flexible MSG_IP layer below the existing SMS code. This layering makes it possible for both new and existing applications to call the MSG_IP layer directly for TCP/IP services without going through SMS.

## V. DATABASE DISTRIBUTION

DBEX is the database executive which communicates database changes between the SLCmicros and the VMS host. This VMS based stand-alone process is the source of most of the network communications traffic in the SLC Control System. DBEX also downloads the database to the SLCmicros at boot time.

DBEX receives data from VMS mailboxes (data destined for the SLCmicros), SLCNET interrupts (data from the SLCmicros) and now TCP/IP interrupts (data from the SLCmicros).

DBEX does not support any direct TCP/IP connection to SLCmicros. Instead, all messages flow through the proxy server. DBEX is a TCP/IP 'client' in that it initiates connections to the proxy server by calling MSG_IP_SEND which transparently makes the connection and sends a KEEPALIVE message. DBEX sends these messages periodically to make sure that the proxy server is alive. If the connection to the proxy server has been broken, the MSG_IP layer automatically tries to re-establish the connection. If a response to the KEEPALIVE message is not received within a given amount of time, an error message is logged in the error log stating that the proxy server is dead.

DBEX calls the MSG_IP layer to receive data using VMS asynchronous I/O methods. Mailbox data is sent to the SLCmicros using MSG_IP_SEND which performs connection management and error logging transparent to DBEX.

## VI. ERROR LOGGING

ERR_INT (error interceptor) is the VMS based stand-alone process which receives error messages from SLCmicros, logs them into the error log file and places them into the error message global section.

As was the case with DBEX updates, all error messages from TCP/IP SLCmicros flow through the proxy server. ERR_INT is also a TCP/IP 'client' and uses the MSG_IP layer to send KEEPALIVE messages to the proxy server.

Error messages are received from the SLCmicros in a similar manner to DBEX by using MSG_IP layer. When a message is received, an event flag is set and ERR_INT receives the data portion of the message. The message is then logged and placed into the global section.

## VII. PARANOIA

The VMS PARANOIA process is responsible for performing 'check' functions and for receiving unsolicited messages from the SLCmicros. Check functions are issued by PARANOIA to check whether an SLCmicro is up and communicating by sending a check request to the SLCmicro and waiting for a reply. If no reply is received, the SLCmicro is marked as off-line in the SLC database. Check functions also tell the SLCmicro to update various device information and send a reply back to the VMS host.

Unsolicited messages are service requests sent by SLCmicros to the VMS host. Typically, PARANOIA receives these requests and forwards them on to another VMS process which provides the requested service.

As is the case with DBEX database updates, all PARANOIA messages flow through the proxy server. PARANOIA is also a TCP/IP 'client'. KEEPALIVE messages are not needed because the check functions periodically send messages to the proxy server which insures that

the proxy server connection is kept open. Check function messages are sent to a list of SLCmicros using SMS calls instead of the MSG_IP layer. Check function replies and unsolicited messages are received using the MSG_IP layer methods described above.

## VIII. DIAGNOSTICS AND TOOLS

There are SLCNET diagnostic tools provided to monitor the activities of both the SLCNET network and the SLCNET based SLCmicros. The tools which support the SLCmicro itself are modified to use both protocols, or are replaced with new TCP/IP functionality.

SLCmicro booting will be accomplished using the standard TCP/IP BOOTP and TFTP protocols which are supported by both TCP/IP stacks. The boot sequence is built into the SLCmicro EPROM and run only at bootstrap time to load and execute the image.

Standard ethernet and IP diagnostic tools are used to trouble shoot network and low level communications problems. These tools include ethernet protocol analyzers, UNIX-style tools such as ping, tcpdump, and netstat. Scripts have been developed for the protocol analyzers to trap SMS and proxy server traffic for development and debugging purposes.

SLCNET provides a means to acquire communications information from the SLCmicros which provide users and technicians needed diagnostic data. This same information is provided from a TCP/IP-based SLCmicro using an embedded HTTP server. This server simply gathers data from the ethernet driver, SNMP variables from Fusion, image filename, boot date/time, and sends it to a requestor. This information is displayed in a format similar to the SLCNET tool for consistency. Some of this information is also written to the database to make it available for display by the SCP.

Commercial TCP/IP based debuggers have been tested during the development phase of the project but the familiarity of the SLC debugger still makes it a good choice. It is possible to modify the SLC debugger to use TCP/IP. No final decision has been made yet.

## IX. SLCmicro

A new Multibus I single board computer from Radisys was chosen as the replacement card for the current SLC micro hardware. This new card, named the Skater card, or EPC (Embedded PC), is actually a PC based computer with an Ethernet interface on a Multibus I card. Fusion, a third party TPC/IP stack was integrated with iRMX and our code development system to provide a fast and maintainable communications interface for the SLCmicro.

A difficult part of the EPC upgrade was getting the current SLCmicro image to boot and run on a PC-like platform. The modifications to the memory map for the actual SLCmicro image were substantial but everything did have a place. The lack of a Non Maskable Interrupt (NMI) on the EPC caused a re-design of the most critical section of the SLCmicros code. This section of code wakes up at 360hz to set up for the next beam crossing and must do so very reliably.

The TCP/IP modifications to the SLCmicro code were straight forward since nearly all the communication needs are handled by a handful of routines. All of the I/O routines were changed to reference a single set of routine contained in one file, making a single place to modify how a SLCmicro communicates, SLCNET or TCP/IP. A flag is set at boot time specifying if this SLCmicro is a TCP/IP SLCmicro or a SLCNET SLCmicro. If the flag is set to SLCNET then message flow is as it has been for SLCNET. If the flag is set to TCP/IP new code is used to setup the TCP connections to the proxy server and then keep the connections up. There are two connections to the proxy server, one for database flow and one for messages and error reports. These connections are kept for the life time of the tasks or are re-connected when errors occur.

## X. STATUS / EXPERIENCES

All major applications have been ported to TCP/IP in both the VMS host and the SLCmicro and are being tested now. Connections from the VMS host and the SLCmicro via the proxy sever are operational and being tested using a mixed system of SLCNET and TCP/IP SLCmicros. SLCNET is still being used for timing data transfer, SLCmicro booting, and debugging.

A some issues came up during the upgrade process. The SMS message service was designed with limited functionality so that when a process had needs that were not met, shortcuts were taken. An example would be, asynchronous I/O completion where the solution was to make VMS system calls directly instead of going through the SMS layer to talk to SLCNET. This lack of good ‘layering’ made it necessary to modify individual applications instead of just modifying a unified SMS layer. The lesson learned is that good interface design and evolution saves a lot of time in the long run.

Quite a bit of time was spent porting the TCP/IP stack to our SLCmicro compilers and the iRMX operating system. It was difficult to integrate this large body of source code into our production system and then make it work well. The porting of the SLCmicro image to the EPC card was also a challenge due to the newness of the EPC card and the constraints of PC-type hardware.

## XI. REFERENCES

[1] M. Crane et.al.,Network Upgrade for the SLC: PEP-II Network, PAC, Vancouver, BC, Canada, 1997.

## XII. TRADEMARKS

*OpenVMS is a trademark of Digital Equipment Corporation.
*Alpha ia a trademark of Digital Equipment Corporation.
*Fusion is a trademark of Pacific Softworks, Inc.
*iRMX is a trademark of Radisys Corporation.