ABSTRACT

This paper describes the history, planning, analysis, design and performance specifications/results of a very fast, real time data acquisition and processing system. The heart of the system is MODCOMP’s fully pre-emptive, realtime UNIX operating system REAL/IX2.

The entire system consists of 19 intelligent communication/interface processors on a VME bus all managed by the REAL/X2 master processor.

The application for this system was developed by Arcata Assoc. of Las Vegas, NV. for use at Nellis Air Force Base.

It resides in the Nellis Range Support network as the master switching node subsystem. The Nellis Network is a data communications system which supports interactive, full-duplex communication of digital data between terminal nodes on electronic combat ranges and range user nodes at Nellis AFB.

Many obstacles to meeting the specified performance had to be overcome. When the system was delivered and installed by MODCOMP it met or exceeded the original data handling requirements and throughput.

Other system features involve communication processor products from SIMPACT Inc. a San Diego company. The paper will present their involvement in delivering this solution system to ARCATA and ultimately Nellis AFB as well as all performance data achieved from this multi-company venture.

BACKGROUND

Arcata Associates was awarded a contract to develop a customized Commercial-off-the-Shelf (COTS) network switching system. This system would integrate multi-vendor equipment into a single network which would process and route tracking, threat and
display data in a real-time environment. Vendors such as IBM, Varian, DEC, Perkin-Elmer and various 68000-based micro-computers would be integrated into a single network. An extensive trade-off study was performed to evaluate current “state-of-the-art” packet switching systems, mainframes and real-time system computers. Based on cost, schedule, risk, and technical requirements, the combination of MODCOMP tri-dimensional products and Simpact communication products was clearly our real-time network switching solution.

REQUIREMENTS

Our network switching system’s primary requirement was to interface with existing network resources while providing the ability to add new vendor products to the network. The existing network is a custom developed network loosely based around RS-449 and HDLC/LAPB levels one and two of the OSI network model. Levels three and above are custom protocol solutions which provide dynamic message routing, message prioritization and reduced message transmission times. A fully pre-emptive real-time operating system was essential to the asynchronous processing of the transmission packets. The system was developed to accommodate existing nodes and their protocols as well as future nodes. Additional requirements were as follows:

- Assign various data rates to all channels via operator.
- Assign protocol, line speeds and channel assignments while system is operational.
- Minimize the number of processor boards required for the system.
- Simultaneously run separate protocols in the same system selected from a library of available protocols (SNA,X.25, HDLC/LAPB, Custom-developed protocols, etc.)
- Simultaneously process 104 channels of HDLC/LAPB protocol at rates from 56K to 700K. Provide selectable line speeds from 50 baud to 1Mb/sec.
- Route a message from one channel to another in less than 10 milli-seconds.
- Process and route 4300 64-byte packets and over 8000 interrupts per second under SW application load.
- Provide system MTBF of over 1000 hours.
- Deliver the system within 120 days after receipt of P.O., to include the successful completion of a detailed acceptance test.
- Provide complete system (HW, SW, spares, maintenance) within ARCATA’s budgetary limits.

MODCOMP SOLUTION

The MODCOMP solution applied to our application was a customized Model 9730 with 17 Simpact communication boards, sixteen configured with 8 RS-449 ports and one supporting 4 V.35 ports. A custom device driver was developed for the MODCOMP
REAL/IX OS and the Simpact Host Interface Software was modified to optimize the interface. A custom cabinet was built to accommodate the large I/O requirements of our network switch and a standard 20 slot VME chassis was used.

Due to the interrupt processing capabilities of the 9730, a single CPU architecture was feasible. This eliminated numerous contention problems with respect to the way packets were received over the network. The 104 channel requirement could be met with a single 20-slot VME chassis, since 8 channels could be driven from a single Simpact board. The Simpact board’s 68020 processors off-loaded all HDLC/LAPB, X.25 and customer protocol processing from the MODCOMP Host system. This allowed the MODCOMP system to be dedicated to message routing, prioritization and application-level message processing. The system’s low weight limit, BTU requirements and power requirements ensured easy installation into existing facilities. MTBF was almost double the required 1000 hours and the system provides easy access to all components.

As one could guess, the success of the Arcata system depends on the ability to respond to events in a fast time frame. Instruction execution, I/O throughput and interrupt handling have to be at their optimum levels in order to achieve the real-time environment necessary. Time critical processes must receive the resources they need without regard for other executing processes. In the past, real-time processing was handled by proprietary operating systems. MODCOMP has been able to add the real-time functionality to UNIX System V making it a real-time UNIX operating system called REAL/IX. The paragraphs below describe how the Arcata system utilizes the real-time features of REAL/IX.

**PRIORITY SCHEDULING**

In a standard UNIX environment, a process has control of the CPU until it uses its entire time slice or until it needs to wait for a software or hardware event, such as an I/O completion. This is known as a non-preemptive kernel. Under REAL/IX, the process will give up control of the CPU during the same conditions, but will also give up the CPU when a higher priority process is ready for execution. This is a feature of a fully preemptive kernel. The Arcata system must always handle a received packet immediately because it is the most critical activity in the network. Scheduling processes is done by a REAL/IX process called the Fixed-Priority Scheduler and is transparent to the user process. The user process needs only to use a system service establishing its priority. The Transmit/Receive process of the Arcata system has the highest priority on the MODCOMP system.
INTERPROCESS COMMUNICATION

One of the fastest ways of interprocess communication is through the use of shared memory. Shared memory is available to all processes that are able to map to it. This mechanism eliminates the system overhead that normally is incurred with message passing mechanisms. One process can immediately access the memory after another process has written to it. The time required is no more than a normal memory reference. REAL/IX allows a process to expand its virtual address space through the use of a system service that maps the shared memory into the expanded virtual address space. Arcata’s switching system utilizes shared memory between the Simpact device driver and the application software for buffer reception/transmission. It also utilizes shared memory for error handling and for updating statistical information.

COMMON EVENT NOTIFICATION

Under standard UNIX, the method of notifying a process of an event is the use of signals. Signals occur asynchronously and must be expected by the receiving process or that process will terminate upon receipt of a signal. When a signal is received, one cannot determine who sent the signal and another signal is ignored if the process is currently handling a signal. REAL/IX has implemented a feature known as the Common Event Notification mechanism which can receive events synchronously as well as asynchronously. Events may also be queued to a process so that they cannot be lost. The Arcata switching system utilizes the event mechanism in a queued synchronous manner. The Transmit/Receive process receives events notifying the process of packet reception, packet transmission and packet transmission acknowledgement from 17 possible Simpact communications cards. While each event is processed, events are continuously posted in an event queue. Thus, no packets are lost since each is tied to an event.

MEMORY MANAGEMENT

Under standard UNIX, physical memory is divided into pieces of fixed length called pages. UNIX brings in portions of the executing program from disk in page increments. Pages that are no longer needed are written back out to disk until they are accessed the next time and then they are read back into memory. The obvious advantage to this is that a program can be bigger than the amount of memory available on the machine because only a portion of the program is in memory at any given time. This process is called paging. Paging is an expensive system overhead for time-critical processes. REAL/IX, through the use of a system service call, allows you to lock in all pages necessary into memory so that paging will not occur. However, this limits the amount of available memory in the system for other executing processes that are not time critical. But it
reduces processing time by eliminating paging. The Arcata system locks all real-time processing into memory.

**TIMER MECHANISMS**

Typically, UNIX time functions operate in intervals of one second. This is usually the best resolution that can be achieved under standard UNIX. Real-time processes often need a finer resolution to control specific events that occur in time critical applications. REAL/IX has provided a choice of selectable resolutions that are 1/64, 1/128, or 1/256 seconds. The Arcata system requires a resolution of at least 10 milliseconds. Process interval timers are used which can be set to expire based on a time value relative to the current system time or a time value that represents a time in the future. The best feature of the interval timer is that each expiration is tied to REAL/IX event mechanism rather than the SIGALRM signal under standard UNIX. This means that a timer can expire while waiting for an I/O completion without prematurely completing the I/O. This premature completion occurs when a SIGALRM signal is received and the I/O then needs to be restarted. The event mechanism does not harm the pending I/O while interval processing occurs.

**SPECIFIC PERFORMANCE ENHANCEMENTS**

The foregoing discussion centered on the attributes of MODCOMP’s REAL/IX as compared to standard UNIX and as applied to the ARCATA system requirements. There were many other modifications to the originally supplied SIMPACT software that were required to meet the ARCATA throughput performance objectives. But this we found, conflicted with an objective to maintain the SIMPACT defined interfaces of its Host Interface Software. Consequently, the performance gains came at the expense of changes to the driver interface. The result was a driver interface that is high performance and is usable with any protocol which uses the SIMPACT Host Interface.

The following presents detailed discussion of the changes made to the interface to the SIMPACT boards to meet the performance goals.

- The original Simpact software provided non blocking I/O and the application polled to find a port where it could do successful read or write. This polling required several system calls for each packet moved. The new interface allows several I/O requests to be queued and when an I/O completes the REAL/IX events mechanism is used to notify the application that the operation has completed. This is the same concept as the REAL/IX Async I/O mechanism.
The Simpact firmware only supported transfers to physically contiguous memory. Since REAL/IX is a virtual memory system, a user’s buffer may cross page boundaries. To solve this problem Simpact allocated contiguous buffers within the kernel and had to copy the data to/from the users buffer.

To avoid this overhead the new driver interface allocates an area of physical memory and requires the application to map the buffers into the application using REAL/IX shared memory primitives. An initialization subroutine is provided which does this mapping. Since the data is not copied on writes, the contents of the buffer must not be modified by the application until the completion event is received.

The interface which Simpact used between the Unix driver and the software running on the board would interrupt the host twice for every transfer. The first interrupt notifies the host that a buffer is available to be read or written. The host would fill in mailbox registers to tell the board where to transfer the data and interrupt the board. The second interrupt indicates that the data has been transferred. This approach forced the board and host to synchronize for every packet. Any time the host delayed interrupts it was likely to slow down transfers.

To avoid the overhead of the additional interrupt and to reduce the coupling between the board and the host, changes were made to the Simpact Host I/O task running on the board and the device driver. Rather than using the mailbox registers, queues of requests and completions were implemented in memory.

Where the Unix driver was responsible for matching a host write with a read from the board the new approach is to move this function to the board. The HIO task on the board now matches requests from the protocol module with requests from the host, does the dma operation and queues the request packet back as a completion packet. When a request is made the host queues the request and interrupts the software on the board. When the board completes an operation it queues the completion packet and interrupts the host.

In each direction there is the chance that several operations may be processed in the interrupt routine for each interrupt serviced.

The driver read and write primitives were reimplemented as subroutines executing within the application instead of system calls. This was done by mapping the driver data structures into the application. This eliminates the overhead of the system call.
For this to work reliably several restrictions apply. First since the interface uses a single reader/single writer queue only one process can do I/O to a given board at any time. This happens automatically if there is only one process which interacts with the boards or if processes are matched up with boards. The same problem also affects signal handlers since they behave like separate processes in that they interrupt the normal flow of execution. The answer is not to initiate reads or writes to the Simpact boards from the signal handler. It is possible to program around these restrictions by protecting the tkdsread and tkdswrite calls using the REAL/IX binary semaphore calls.

PERFORMANCE SPECIFICATIONS

The ARCATA system configuration:
The performance migration from where MODCOMP started its modifications to the final delivered system results are presented and compared to the original requirements.

Arcata proposed two acceptance tests to be performed. Figure 1 shows performance requirements for a loop-back configuration of 50-9.6K lines, 14-56K lines, 2-500K lines and 1-1.8Mb line.

Figure 2 shows maximum performance requirement for a loop-back configuration of 100-56K lines, 2 500K lines and 1-1.8Mb line. All lines, except for the 1.8Mb line, transmit and received 70-byte packets at the specified number of packets per second (pps). The 1.8Mb line would receive 6000 byte blocks ten times per second.

The originally supplied Simpact Software performance rates were as in Figure 3.

With the many modifications to the original supplied Simpact Software we were able to exceed the required performance, as shown in Figures 4 & 5.
REFERENCES


