INTEGRATION OF COMMUNICATIONS AND TRACKING DATA PROCESSING SIMULATION FOR SPACE STATION

Robert C. Lacovara, E. E.
Affiliate Instructor
Department of Electrical Engineering and Computer Science
Stevens Institute of Technology
Hoboken, New Jersey

A simplified model of the communications network for the Communications and Tracking Data Processing System (CTDP) is developed. It is simulated by use of programs running on several on-site computers. These programs communicate with one another by means of both local area networks and direct serial connections. The domain of the model and its simulation is from Orbital Replaceable Unit (ORU) interface to Data Management Systems (DMS).

The simulation was designed to allow status queries from remote entities across the DMS networks to be propagated through the model to several simulated ORU's. The ORU response is then propagated back to the remote entity which originated the request. Response times at the various levels were investigated in a multi-tasking, multi-user operating system environment.

Results indicate that the effective bandwidth of the system may be too low to support expected data volume requirements under conventional operating systems. Instead, some form of embedded process control program may be required on the node computers.

NASA Colleague: Oron L. Schmidt EE7 X6301

Introduction

Whatever the form of flight hardware which eventually comprises Space Station, distributed processors linked by local area networks are certainly to be incorporated. This study addresses one such distribution which supports the Communications and Tracking Data Processor (CTDP). This system is complex and incorporates many functions, but may be considered a data portal to other satellites and earth stations. Consequently, CTDP accepts and encodes on-board data for transmission, receives data from external sources, and points antennas as required. Additionally, various housekeeping tasks are performed. Some of these are computationally intensive. As it progresses in orbit, for example, Station presents parts of itself as obstacles to antenna patterns. A real time obscuration map must be maintained so that transmit antennas do not illuminate Station structure, experimental apparatus, or extra-vehicular crew.

Baseline concepts of the CTDP computer network have been proposed, and work on the implementation has started. This study examines a simplified version of the network, reduced in number of nodes and branches. This is not a graph-theoretic study. Instead, a network of processors have been linked in an approximation of expected requirements and simple programs have been installed at the computational nodes.

The objectives of the study are straightforward. The ability of the network to handle data loads by measuring data throughputs is of interest. So is the ability of off-shelf multi-tasking systems to operate in an environment of mixed local area networks (LANs) and dedicated point to point links. Lastly, some judgements are made concerning the applicability of different operating systems and programming languages.

The approach of the study is direct. The linked computers executed simple communication programs. Data loads of the magnitude estimated for some Space Station equipment were applied to the system. The coding was performed in C and Pascal under the Ultrix and VMS operating systems running on a Digital Micro-Vax and a Digital 11/780. Conclusions are drawn from the outcome.

Hardware Environment

The simplified model of the CTDP network consists of a tree whose nodes are computers and whoses branches are either dedicated RS-232 links or a common LAN, TCP-IP. The overall system which was modeled consisted of 3 layers in this tree.

The root of the tree is the main CTDP processor. It communicates via LAN with some number of subordinate processors called comtrollers (short for communication controllers).

The comtrollers in turn communicate via point to point links such as RS-232 with hardware modules such as signal processors, antenna gimbal systems, and so on. These hardware modules are termed orbital replaceable units (ORU) and are considered physically proximate to the associated comtroller.

At the time of this study the CTDP main computer (a VAX 11/750), a single comtroller (Micro-VAX), and a software simulation of three ORU's on a VAX 11/780 are available.

The study utilized the comtroller and the ORU simulation under simulated loads. The CTDP main computer was not included in this study at the time of writing. Therefore the hardware under consideration is a Micro-VAX connected via three dedicated RS-232 lines to three simulations of ORU's on a VAX 11/780.

The structure of parent (comtroller) with three children (ORU's) may be further elaborated. Specifically, the RS-232 line characteristics become important. These three lines are connected at each computer to a multiplexer, such as a Digital DZ. The salient characteristic of the multiplexer is that the four to eight lines which it supports are handled by a single processor local to the multiplexer. As a result, the multiplexer throughput is significantly less than the individual line rates supported by the device.

In this case, the three lines are operated asynchronously at 19,200 baud, with 8 bit data and single stop bit. After accounting for parity and so on, each line could support 1,920 characters per second. The effect of having a multiplexer for the three lines is that a total throughput of about 775 characters per second is attained in aggregate, as opposed to an ideal aggregate rate of 3 times 1,920 or about 6,000 characters per second. This will be seen to be a major limitation of the simulation.

Software Environment

The comtroller is a Micro-VAX operating under Ultrix-32m. This is a port of a Berkeley Unix operating system and supports a range of languages and utilities. Of chief interest are the following features: the operating system detached process utilities, the interprocess communication support, and the programming language used for the simulations.

Unix provides a simple mechanism to start processes detached from the user's terminal. Almost as convenient is an interprocess communication facility called a socket mechanism. The combination of these abilities makes it possible to create some number of detached processes which may still communicate with each other via sockets.

Since it was determined early in the study that these features would be required, the C language was chosen to code the programs which would process data. Other languages were available, but an unfortunate characteristic of Unix systems is an apparent disinterest in providing "equal access" from all supported languages to the operating system features. Although another language such as Pascal might have been used, the differences between C and Pascal did not justify any effort to make the Berkeley Pascal implementation compatible with all the operating system features readily accessed from C.

The comtroller software consists of a controlling process called CMTR, and three subordinate processes called RX7, RX6, and RX5. The socket mechanism provides communication between CMTR and each of the RX programs.

The three RX programs act as interfaces to the three ORU simulations. The processes are run detached, and thereby provide some degree of parallelism in the multitasking system. Unfortunately, the three processes each use the same serial line multiplexer, which essentially destroys any efficiency gained by using detached processes.

CMTR periodically querys RX7, RX6 and RX5. Each of these three programs in turn interrogates the software ORU simulations on the 11/780. Normally, the ORU simulations will return between 100 and 1000 characters as a response, and once these are received by the RX programs they are passed on to CMTR. CMTR then indicates to the user the total number of characters transferred, the time required to transfer, and calculates the rate of transfer. The number of transfers and size of data transferred may be varied.

As has been indicated, the serial communication lines pass only 775 characters per second. This is the rate limiting path in the system. The socket mechanism in comparison is far faster, and easily exceeds 50000 characters per second. As a result, of the two mechanisms used to transfer data between processes, only degradation in response due to the effective throughput rate of the serial communication lines need be considered.

On receipt of a status query, the RSP programs open an associated text file containing data representing ORU status, and write this data to an associated RS-232 port. The RSP programs do little else but housekeeping, and each program is only 30 lines of code. In practice, the programs exist as blocked tasks waiting to read a status query.

The ORU simulation programs running on the 11/780 are quite simple. There are three called RSP7, RSP6, and RSP5. These programs are like their counterparts RX7, RX6, and RX5 in that they are identical to each other except for the I/O ports to which they are attached. However, these programs run detached under the VMS operating system, and are coded in Pascal.

Simulation Behaviour

The simulation was operated for several hours of CPU time. Data rate from CMTR query through the return of the data from the three ORU's was calulated under various conditions of system loading, data block size, and program priority.

The result may be summarized easily. Using standard DZ style multiplexers and standard Digital terminal drivers allows data rates of the three lines to be less than 775 characters per second under any conditions. Data rates may be as low as 100 characters per second if task priorities are low while the system is heavily loaded. Estimates made by experienced NASA staff indicated that the traffic to be expected from three ORU's is likely to range from 1000 to 8000 characters per second. In short, the commonly available hardware and software, even with informed tuning, is unacceptable in terms of realistic data requirements.

The immediate conclusion is that specialized drivers for communication need to be developed to support block data transfer under multitasking operating systems.

Several other conclusions may be drawn from observations made during the development of the system.

Neither C nor Pascal are particularly appropriate languages in the application. Neither language inherently supports concurrency or interprocess communication. These are very desirable attributes of languages for this application. One advantage of C is that it is probably a good choice of a language in which to write a customized driver for the serial communication lines. The two languages are otherwise equally ill-suited to the task of networking support.

During the course of the study, some attention was paid to the use of ADA as a programming environment. Although availability precluded use of ADA for this study, it appears that ADA incorporates concurrency and communication features which make it a reasonable choice for an embedded control application such as the present study.

Note that since concurrency and interprocess communication in the present study are provided by the operating system, the code and techniques are non-portable. It is in fact unlikely that the C code or the Pascal code could be ported to any other operating systems and still be made to work. This unhappy result is due to the fact that the socket mechanism provided under Ultrix is not part of the C language, but of Ultrix itself. Similar comments may be made about features of the Pascal code.

Summary

The CTDP network may be simulated in non-real time by use of typical operating systems and typical driver packages. Real time simulation, and simulation of extended systems will require customized software.

Traditional modular programming languages such as C and Pascal are unsuitable when compared to ADA. Although ADA code was not written, examination of ADA features indicate that ADA is a potentially superior language in the application.