The Application of a Distributed Computing Architecture to a Large Telemetry Ground Station

THE APPLICATION OF A DISTRIBUTED COMPUTING ARCHITECTURE TO A LARGE TELEMETRY GROUND STATION

Robert K. Buell
Fairchild Weston Systems Inc.
Data Systems Division
Sarasota, Florida

ABSTRACT

The evolution of telemetry ground station systems over the past twenty years has tracked the evolution of the mini-computer industry during that same time period. As the various mini-computer vendors introduced systems offering ever increasing compute power, and ever increasing capabilities to support multiple simultaneous users, the high end of the telemetry ground station systems offered by the industry evolved from single stream, single user, raw data systems to multi-user, multiple stream systems supporting real-time data processing and display functions from a single CPU or, in some cases, a closely coupled set of CPUs. In more recent years we have seen the maturation of networking and clustering concepts within the digital computer industry to a point where such systems coupled with current workstation technology, now permit the development of large telemetry ground station systems which accommodate large numbers of simultaneous users, each with his or her own dedicated computing resources. This paper discusses, at a hardware block diagram and software functional level, the architecture of such a distributed system.

Key Words: Telemetry system, Distributed architecture, Networking, Workstations

INTRODUCTION

In December of 1986 the Data Systems Division of Fairchild Weston Systems Inc. was awarded a contract by a major domestic aerospace corporation to design and build a large telemetry ground processing station. Although the system being purchased was intended to support a specific application, the specification to which the system was to be designed had been written to far exceed the requirements of the particular target application. This “overspecification” of system requirements was intentional. It was intended by the customer that the system design resulting from the contract would have the potential to be applied to future, as yet undefined, requirements without any need for redesign.

In order to most cost effectively meet the stated and unstated design goals, it was decided to implement the ground station system as a distributed network of processing nodes, each responsible for a specific subset of the overall system processing requirements. This approach offered two major advantages. First, it forced a high degree of modularity into the design, permitting the development of the major system software components to occur in parallel and to subsequently be integrated with minimal integration problems. Second, it resulted in the design of a suite of software that can be implemented on a wide range of hardware configurations, thus permitting the customer to tailor the cost of any future implementations of the system by tailoring the amount of hardware (number of processors) on which the software is hosted. This tailoring process may be accomplished without changing the functionality or user interface provided by the system software.

This paper describes, at a high level, the system design that resulted from the above referenced contract. The design and implementation of the subject system have now been completed, and the design concepts incorporated in the system architecture have all been validated.

BASIC SYSTEM REQUIREMENTS

As with any system, the design eventually implemented was driven by the requirements to be satisfied. The highest level functional requirements addressed by the subject telemetry ground station system can be subdivided into two categories; those requirements defined by interfaces external to the system, and those defined by interfaces internal to the system. There are two primary external interfaces; 1) the telemetry data to be processed, and 2) the displays and formatted reports to produced for the system users. The single primary internal interface is that between the various computers which comprise the system.

The high level functional requirements derived from the external interfaces are three in number: 1) data acquisition and storage, 2) data manipulation, and 3) data display. The basic data acquisition and storage requirement was to accept telemetry data from multiple radio links or instrumentation tape recorders, make that data available for use by several processing modules within the system, and simultaneously archive that data to disk for later recall and analysis. The basic data manipulation requirement was to perform one or more of a variety of manipulations upon the acquired data to transform that data into a form more easily understood by the system users, or into a form which more naturally supports the calculations to be performed upon the data. The basic data display requirement was to provide a convenient means for the system users to view the acquired data on a color graphic display or on paper, in a format which enables them to efficiently analyze and act on the data. The number of simultaneous users required by the written specifications to be supported by the system was twenty one, although there was a strong desire to be able to allow that number to range anywhere from one to approximately fifty.

The high level functional requirement derived from the single primary internal interface was to move the various types of information used or manipulated by the system between it's various computers in an efficient manner.

The above stated high level system requirements, and the detailed requirements that emanated from them, were addressed with a distributed computing architecture using off-the-shelf minicomputers, workstations, and networking components to provide a hardware environment in which a suite of modularized software based components could be implemented to address the specific system requirements.

OVERVIEW OF SYSTEM HARDWARE ARCHITECTURE

The system hardware architecture is most clearly represented in a top level block diagram as three primary functional blocks organized in a processing pipeline. Each of the three functional blocks actually incorporates parallel processing capabilities, but the overall data flow inherent in the system design is of a pipelined nature. Figure 1 summarizes the system hardware architecture.

The first functional block in the system hardware block diagram is generically referred to as the Telemetry Front End, and includes bit synchronizers, decommutators, and preprocessors. The subject system incorporates six each bit synchronizers and decommutators, multiplexed into two preprocessors. This first functional block is responsible for data synchronization, decommutation, and all high speed processing which must be applied in real time to each sample of the acquired data. Alarm and limit checking, time tagging, engineering unit conversion, and data compression algorithms are performed within this segment of the system.

The second functional block in the system hardware block diagram is referred to as the Host Computing Complex. It includes a VAXcluster consisting of two VAX 8600 processors, two HSC50 intelligent I/O processors, a set of cluster common disk and tape drives, and several black and white and color printers. This functional block is responsible for the acquisition of the decommutated data into the digital computer environment, calculation of complex or involved data manipulations, control of all data flow to disks, tapes, and printers, and distribution of data across the network.

The third functional block in the hardware block diagram consists of the network of color graphic workstations. The system includes twenty one workstations, each dedicated to a single user. The workstation functional block is responsible for all data processing related to the display of data, to the capture of screen images, and to the translation of those images into formats compatible with the device on which they are to be printed.

The first two functional blocks are interconnected via a set of 16 bit wide parallel interfaces and RS-232 lines. The RS-232 lines and some of the parallel interfaces are used to program the various pieces of equipment in the Telemetry Front End from one or both host computers. The remaining parallel interfaces are used to provide high speed data paths to transfer acquired data from the Telemetry Front End to the host computers.

The second and third functional blocks are connected via an Ethernet based network running two simultaneous and independent data protocols. The first protocol, DECNET, provides a wide range of file transfer, remote file access, task to task communication, and virtual terminal capabilities. The second protocol, a special protocol implemented to support system data distribution, provides much less inherent capability than DECNET but operates at much higher effective data transfer rates. Both data transfer protocols run simultaneously on the same Ethernet cable.

OVERVIEW OF SYSTEM SOFTWARE ARCHITECTURE

The system software architecture mimics, at the top level, the system hardware architecture and may be represented as three pipelined functional blocks of processing.

The software functions associated with the first functional block, the Telemetry Front End, include time tagging the input data, performing alarm and limit checking, performing engineering unit conversion calculations, and performing data compression calculations. These functions are implemented in the preprocessors as sets of independent sequential algorithms that are applied to the data samples as they move through the preprocessors. Each of the two preprocessors is fully capable of supporting all six input data streams. Two preprocessors are included in the system to support operations in either of two modes. The first mode utilizes the two preprocessors to simultaneously process the input data streams through two independent hardware paths to provide an element of failsafe redundancy. The second mode utilizes the two preprocessors to support the simultaneous processing of independent data streams, which amounts to utilizing the one large system as two smaller and independent systems.

The software functions associated with the second functional block, the Host Computing Complex, include the acquisition of data into the host computers, recording of archive data to disk and subsequent retrieval from disk, executing user defined subroutines to manipulate acquired data into derived data, and distributing both real time display data and data recalled from archive disks to the network of workstations.

The software functions associated with the third functional block, the network of workstations, include the creation and maintenance of display formats, the displaying of data using those formats, the capture of screen images, the translation of those images into

formats required by various hardcopy devices, and the general support of the mouse driven pop-up menu style user interface.

The software environment in both the second and third functional blocks is the multi-tasking environment provided by the VMS operating system. Within each functional block, the major software functions are implemented as independent processes that communicate with each other via standardized interfaces. To the extent that certain classes of processing are performed in the host computers, and other classes of processing are performed in the workstations, a significant amount of parallelism is achieved regardless of the operational mode in which the system is used. When the system is operated in a “hot backup” mode of operation where the data streams are acquired simultaneously through two independent hardware paths, or when the system is operated as two independent subsystems each acquiring and processing independent data streams, then the parallelism exhibited by the system is in effect doubled. In both cases, the ability of the system to operate in different modes is a direct result of the implementation of distributed architecture concepts within both the system hardware and software. Figure 2 illustrates the major software functions implemented in the system and the primary data flows occurring between those functions.

SUMMARY OF MAJOR COMPUTER SOFTWARE COMPONENTS

As illustrated in Figure 2, the software running in the Host Computing Complex and the network of workstations is divided into a number of major components. Each of those components is responsible for implementing a specific subset of the overall software functionality. The following paragraphs summarize the major software components and the functions assigned to each.

Data Acquisition and Recording - The Data Acquisition and Recording function runs in the Host Computing Complex functional block. It is responsible for controlling the acquisition of data into the particular computer in which it is running, and for writing the acquired data out to tape or disk.

Data Distribution - The Data Distribution function consists of processes that run on both the host computers and the workstations. This function is responsible for moving the real time display data from a host computer resident Current Value Table to duplicate tables resident on the workstations. This transmission of data occurs over the Ethernet via a broadcast technique. Real time display data therefore is placed on the network only once per broadcast cycle, regardless of how many workstations wish to use that data. Since the Data Distribution function is itself distributed around the network, it is able to completely handle the distribution of real time data and maintain Current Value Tables in each of the workstations. All application software modules running in the hosts and workstations are therefore insulated from any requirements to deal directly with the network.

Data Manipulation - The Data Manipulation function consists of a number of different processes each of which performs specific types of data manipulation on data either in real-time or on data recalled from an archive disk. These processes will typically run in one of the host computers. They may however, at the option of the system operator, additionally run in one or more workstations to support data manipulation requirements that are unique to particular users. This function is responsible for providing a shell under which user written code executes both during real-time operations and during data recall operations to perform mission or user unique calculations on the acquired data. The results of these calculations are made available to the Current Value Table during real-time operations for display purposes and are also archived to disk to support later recall. Additional functionality provided by Data Recall includes the capability to subset and recombine archive disk files, and the capability to edit archive disk files.

Data Recall - The Data Recall function normally runs in one of the host computers. This function is responsible for accepting data requests, in the form of standardized message packets, from other applications for subsets of data from the archive disk files. Once located, the requested data is formatted into a series of data packets which are then sent back to the requester. Because DECNET was used to implement the communications interface between the Data Recall function and it's client applications, the fact that the Data Recall process is running in a remote computer is transparent to the requesting application. It is therefore possible for the Data Recall process and the requesting application to be running in the same processor without any requirement for modification of either. In a normal operational scenario, the Data Recall process will run in one or more host computers and act as a server process which provides data packets of recalled data to all requesting workstations. At the system operator's option however, a copy of the Data Recall process may be distributed to one or more workstations and used locally to recall data resident on local workstation disks.

Message Handling - The Message Handling function is similar to the Data Distribution function in that it consists of multiple processes that run on both the host computers and the workstations. This function is responsible for providing a standardized mechanism to be used by all applications within the system to issue error and warning messages, and in some cases to pass text messages between themselves. Again as with Data Distribution, the existence of the network is transparent to the application making use of Message Handler services. Once a message is generated by an application it is passed to the Message Handler by a simple subroutine call interface, and the subsequent transmission of that message to one or more destinations in the system, including a common system log, is automatically performed. An added advantage provided by the Message Handler function is that it implicitly enforces a standardized message format on all applications, making error and warning messages appear consistent no matter where within the system they were generated.

Data Display - The Data Display function runs exclusively on the workstations. It is responsible for accepting data either from the Current Value Table or from Data Recall data packets, and displaying that data in user defined formats. The processing associated with the display of data may be simple or complex depending upon the level of complication of the particular display that the user has designed. By having the data distributed to the workstations, and letting the individual workstations be responsible for all display processing, system users are effectively isolated from each other and are protected from suffering any degradation of display performance based upon the display processing initiated by any other user.

Screen Image Processing - The Screen Image Processing function also runs exclusively on the workstations. This function is responsible for translating the bit mapped images copied from the display screen in response to user hardcopy requests into formats that are compatible with the various hardcopy devices available on the system. All screen images are in color. They may however be translated into formats consistent with color printers or black and white printers. Once translated, the resultant files are sent to either local or remote print queues that are associated with the destination device.

CONCLUSION

A distributed computing architecture has been applied in the design of a large telemetry ground station. The resulting system, which incorporates twenty one MicroVAX based color workstations, two VAX 8600 host computers, and six PCM input links, provides a flexible telemetry data processing tool applicable to many potential problems. The distributed nature of the design facilitates the replication of the system on different sized suites of hardware without impacting the functionality available to system users.

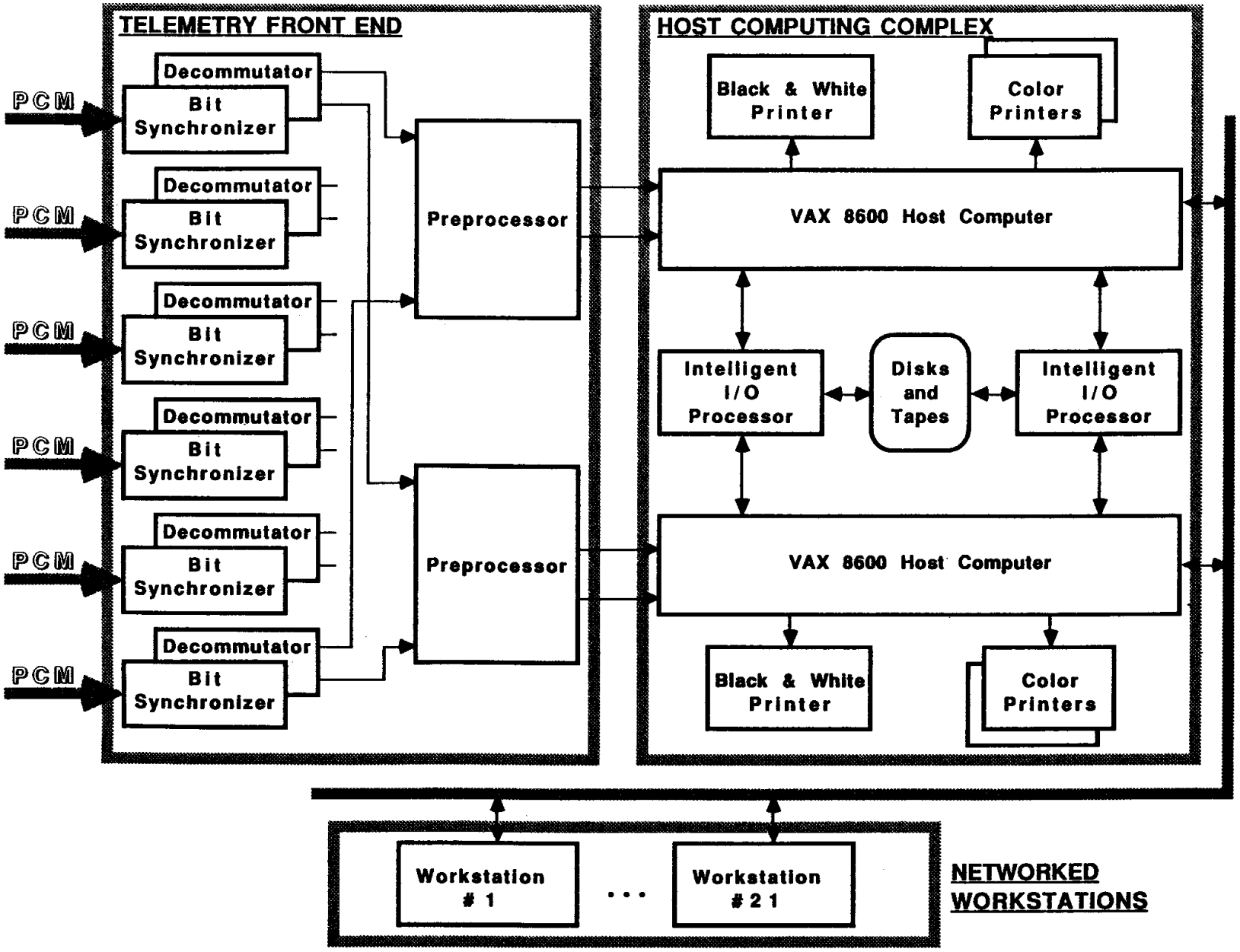

FIGURE 1 - HARDWARE BLOCK DIAGRAM

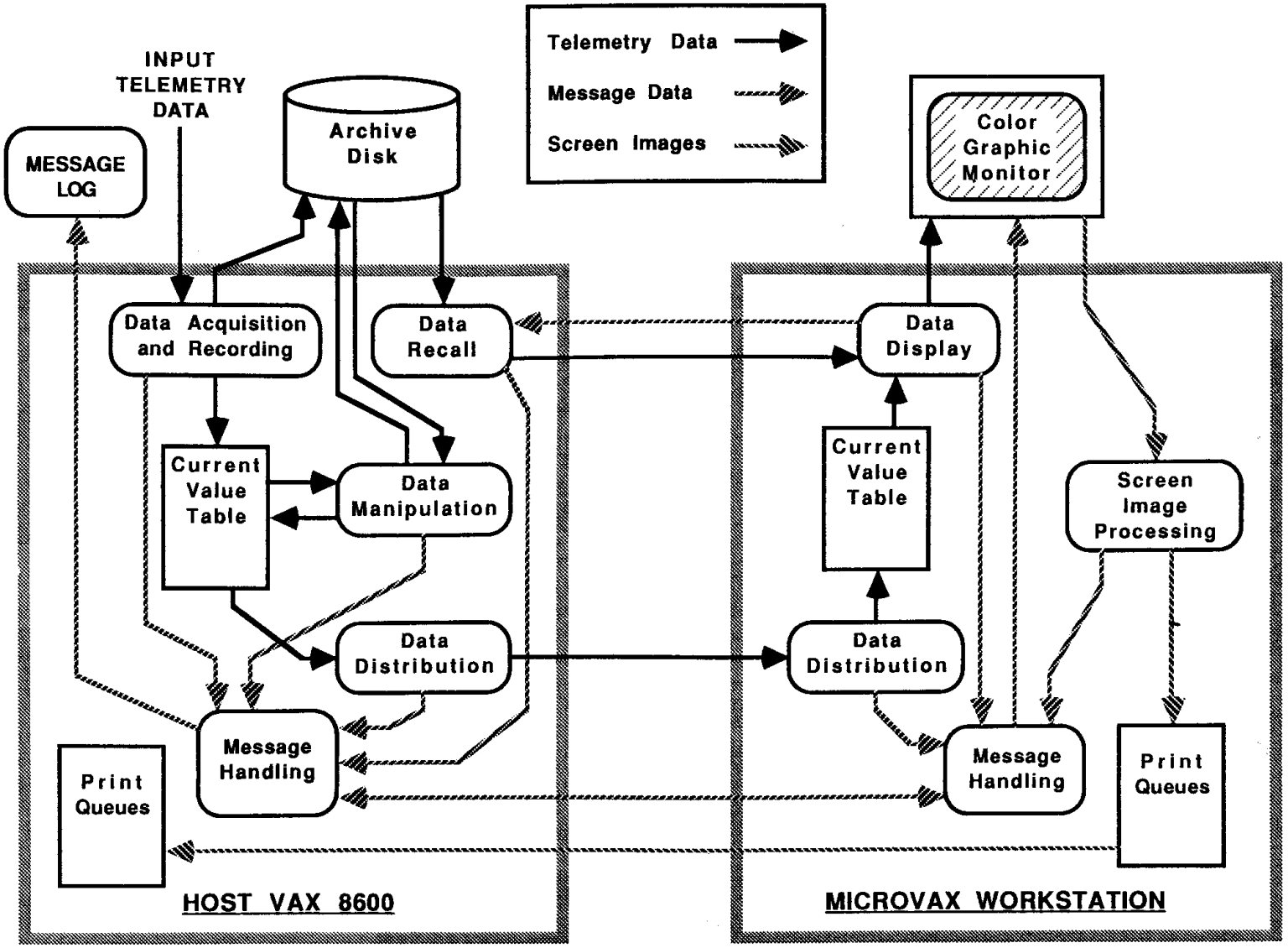

FIGURE 2 - MAJOR SOFTWARE FUNCTIONS AND DATA FLOWS