ADVANCED TELEMETRY PROCESSING AND DISPLAY SYSTEM (ATPDS)

Ted Leichner
Stephen J. Nicolo
Ed Snyder
Mark Stacy
Charles Ziegler
GDP Space Systems
300 Welsh Road
Horsham, PA 19044

ABSTRACT

This paper describes a PC-based Advanced Telemetry Processing and Display System (ATPDS)- a high-end, real-time telemetry processing and display system implemented on a COTS PC platform. for A network-centric architecture was chosen from candidate architectures as the most viable for the ATPDS. The network-centric architecture is Windows NT-based, client/server based, supporting clients and servers on both local or remote PC workstations. The architecture supports distributing processing loads across multiple workstations, optimizing mission processing requirements. The advantage of this system is its flexibility and expandability with low acquisition and life-cycle support costs. The ATPDS allows the user to configure one or more small systems into a larger high-end system based on varying mission requirements.

KEYWORDS

Windows NT, Network, Telemetry Processor, Object-oriented, Client-Server, Real-Time

INTRODUCTION

Historically, telemetry data acquisition, processing and display systems have been implemented using custom-developed hardware and software architectures. Many of these systems have been installed in flight test facilities and telemetry stations worldwide and are providing high performance acquisition processing and display of multiple streams of telemetry data. These customized telemetry systems tend to be expensive because they use proprietary, high-speed hardware busses and system software.

Another significant issue is their inherent inflexibility. Telemetry processing requirements can change from mission to mission, or may even change during the life of a single program. The ability to modify and reconfigure a customized telemetry system is nearly non-existent, or at best is achievable only at the penalty of high costs.

With the introduction of the personal computer, alternative approaches to the processing of telemetry data began to evolve. The early PC-based systems were burdened by low-speed processors and data rate restrictions imposed by the hardware platform (e.g., ISA bus). These limitations made the processing of multiple high-rate data streams virtually impossible. As the processing power of the PC increased, these systems also increased in capability but were still limited by a single processor, single task operating system with little or no networking capability. Today this is no longer the case and industry has finally made it possible to build a truly open, PC-based, next-generation system with all the power of the traditional systems for a fraction of the cost.

The ATPDS is a PC-based system and is the result of work done by GDP Space Systems for Eglin Air Force Base.

CANDIDATE SYSTEM ARCHITECTURES

GDP Space Systems evaluated three different candidate system architectures for possible use in the ATPDS. The three architectures differ primarily in how telemetry data is controlled and distributed. All three assume the use of IBM compatible PCs, Microsoft's Windows NT Operating System, Intranet and Internet accessibility to the telemetry data, and adherence to relevant Federal and ad-hoc standards (e.g., IRIG-STD-106, IRIG-STD-107, TMATS).

The three candidate architectures are listed below.

1. Reflective Memory
2. Software Centric
3. Network Centric

The first architecture is an implementation using reflective memory in each core PC within the system for distributing telemetry data. With the reflective memory architecture, each core PC contains a bank of reflective memory. The data within the reflective memory is common to all the PCs. Thus, when a PC changes, data within the reflective memory that change is mirrored in all the other PCs almost instantaneously. As a result, data flows through the system quickly and transparently.

The second architecture uses remote procedure calls for distributing telemetry data. Remote procedure calls enable software to call functions regardless of where those functions reside. The called functions can be on the same PC as the calling software or on different PCs. Where the function actually resides is transparent to the calling software. In this "software centric" approach telemetry data consumers use the remote procedure calls to obtain their telemetry data.

The last architecture is an implementation of network addressing for distributing telemetry data. In this "network centric" approach, telemetry data servers either multicast or direct address (unicast) the

telemetry data to the telemetry data consumers. Which addressing scheme is used depends upon the commonality of the data being distributed to specific telemetry data consumers. Where there is a large degree of commonality, multicast addressing is used; where there is little or no commonality unicast addressing is used. The network centric approach allows the use of multiple addressing schemes simultaneously. As a result, some telemetry data consumers may be receiving their data via multicast addressing; others may be receiving their data via unicast addressing.

REFLECTIVE MEMORY APPROACH

In this approach, each PC has a portion of memory that is shared with other PCs in the system. (See Figure 1). The data within this shared memory is accessible by all participating PCs. Thus, when a PC changes data within the shared memory, those changes are immediately available to the other participating PCs. Because data is being distributed via shared memory delays associated with importing and exporting data through traditional routes are eliminated.

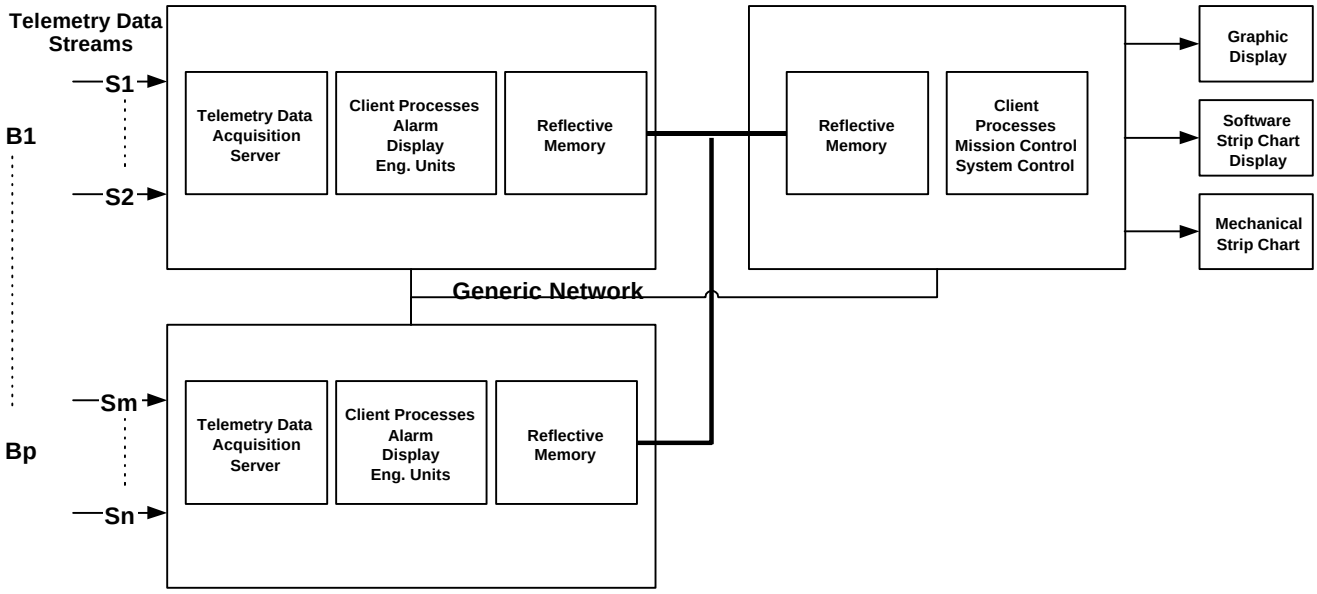

Figure 1 Example of Reflective Memory Architecture

Traditionally, reflective memory uses specialized memory cards connected together by a high-speed network. A memory card goes into each participating PC and changes to the memory on any one card are distributed via the network to the remaining cards. The PCs are unaware of the connecting network's presence.

The advantages of reflective memory include quick, efficient, and transparent transport of telemetry data throughout the system. The system is scalable. Additional PCs containing reflective memory are easily added to the system. Since the PCs are networked together, they needn't be co-located; the PCs may be as distant as the supporting network and memory cards permit.

The disadvantage of reflective memory is the market is fractured with little commonality between different vendors' products. Should a chosen vendor's product be unavailable in the future, switching to or adding another vendor's product necessitates rewriting the system software, an expensive and time-consuming process. Adding new capabilities or incorporating information from other systems is likely to require adding more reflective memory PCs and rewriting the system software. The cost of the system is high, primarily because of the cost of specialized memory cards. Developing the system is expensive. Specialized software is necessary to fully exploit reflective memory's capabilities.

SOFTWARE CENTRIC APPROACH

The software centric architecture uses remote procedure calls to distribute telemetry data. (See Figure 2). Remote Procedure Call (RPC) is a technique for building distributed systems. In essence, RPC allows a program on one machine to call a function on another machine without knowing that it is remote. RPC makes the use of existing network communications transparent to the user. That is, the RPC user is unaware the PC-to-PC communications is occurring. This transparency is one of the great strengths of RPC. Because the application software does not contain any communication code, it is independent of the particular communications hardware and protocols used, the operating system used and the calling sequence needed to use the underlying communications software.

This means the design and writing of application software can occur before these choices have even been made. RPC handles any required data reformatting and provides transparency to byte ordering and differences in data representation (real number formats, etc). RPC has been in existence since before 1976 and has been in use in academic and commercial areas for many years. Several defacto and international standards exist regarding RPC and even permit their user over the internet (e.g., SUN-RPC, RFC 1057, RFC 1833, and RFC 1831).

In this "software centric" approach telemetry data consumers use the remote procedure calls to obtain their telemetry data. Plus, the total processing load is easily distributed across multiple PCs, reducing or eliminating excessive loading on any one PC.

The advantages of the software centric architecture include scalability and load balancing across multiple PCs. When the processing load becomes too great, additional processing power can be added simply by adding a PC. RPC is standardized in several forms so it is possible to have different PCs with different operating systems all working together. Microsoft also offers RPC which rends RPC with ubiquity.

A disadvantage of the software centric architecture is reduced performance resulting from requesting and waiting for data from another PC. The cycle time, as estimated by Microsoft, is on the order of only several hundred requests/waits per second. The high cost of overhead associated with each transfer significantly throttles this architecture.

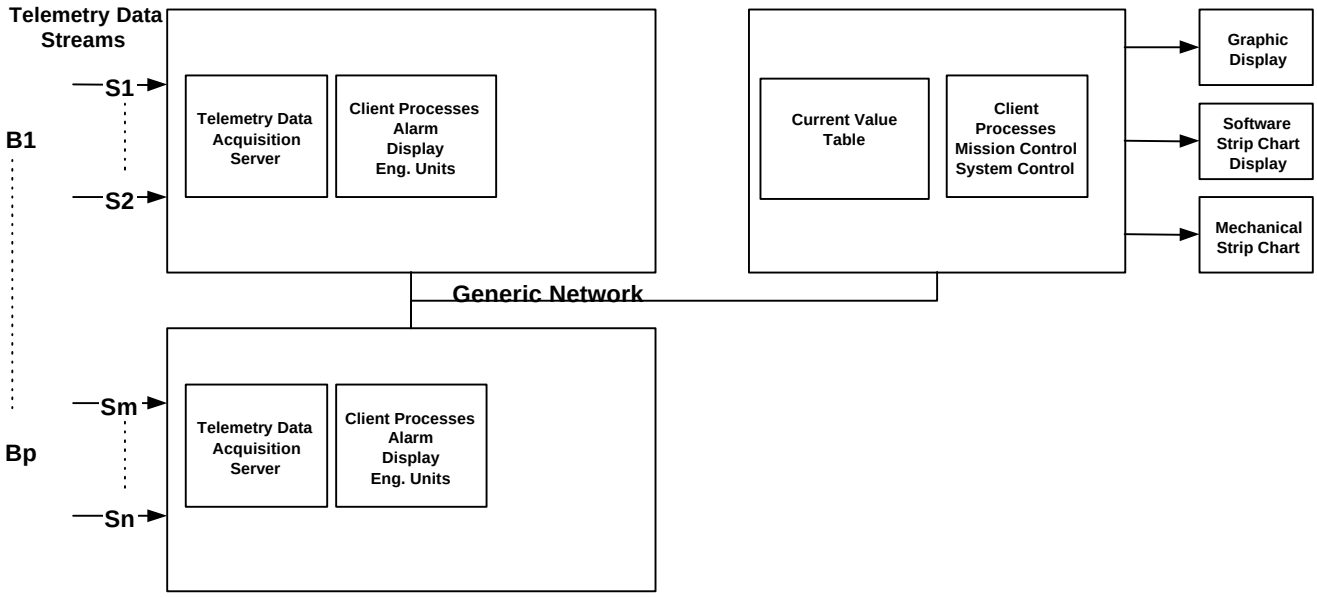

Figure 2 Software Centric Architecture Example

NETWORK CENTRIC APPROACH

The network centric architecture uses network addressing for distributing telemetry data. In this "network centric" approach, telemetry data servers either multicast, or direct address (unicast) the telemetry data to the telemetry data consumers. (See Figure 3.)

In this approach "streams" of telemetry data flow from server to consumer. Once a stream is established, the server automatically and continually feeds that stream until told to stop. It is up to the consumers (e.g., displays, archives, and strip charts) to extract from these streams the information they need in a timely fashion. Within this approach, a PC that injects processed data into the network for consumption by some other PC can also process raw telemetry data.

Secondly, by using a standardized stream format, new stream types may be easily introduced and automatically handled by the ATPDS without any software changes to the existing system components.

The ATPDS is easily expanded to carry large bandwidth, raw telemetry data while also supporting a large client base. When this is true, multiple networks, high-speed and low-speed, may be used to support, respectively, the bandwidth-hungry raw telemetry data consumers and the less demanding display-type consumers.

The network centric architecture scales easily from a single PC to multiple PCs. It is eminently flexible and extensible. The user may tailor the system to a particular application using resources already owned.

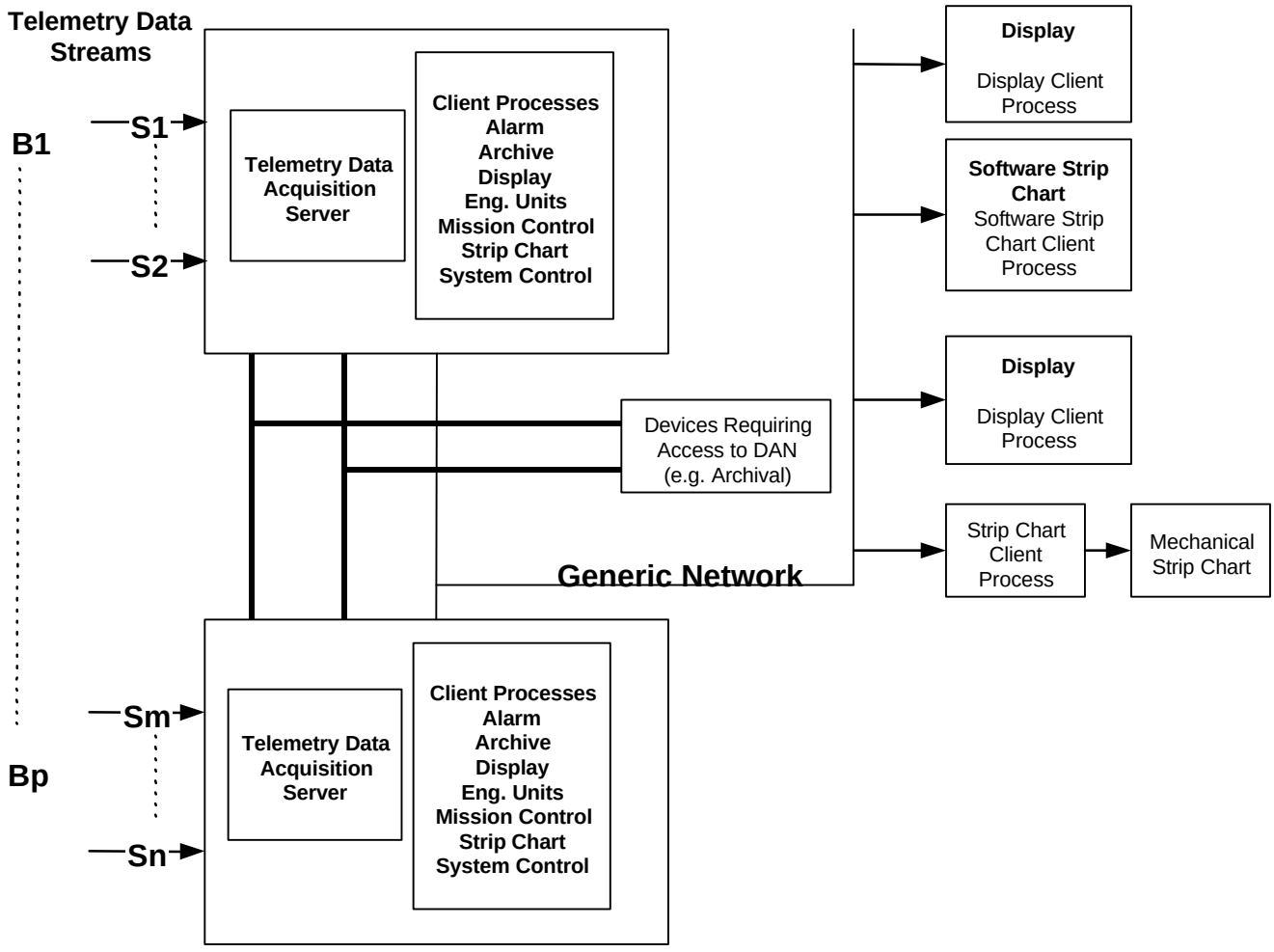

Figure 3 Generalized Network Centric Architecture

The network centric architecture provides an easy to use mechanism for moving data within the system. This mechanism enables the system to be scalable, flexible, and open to new technology. Since this approach is a software solution using standard software and network mechanisms, it is independent of the underlying hardware infrastructure. As a result, newer, faster, and better hardware platforms and networks can be readily incorporated without rewriting software.

The ATPDS when scaled beyond a single PC is largely limited by the supporting network. Greater performance is achievable by either using a higher bandwidth network, or using multiple networks in parallel.

COMPARISON

Of the three network architectures examined, the network centric architecture has the greatest scalability, best flexibility, lowest cost, and highest extensibility.

Table 1 Architectures Comparison Chart

	Reflective Memory	**Software Centric**	**Network Centric**
PC-Based	**Yes**	**Yes**	**Yes**
Scalable	**Yes**	**Yes**	**Yes**
Flexibility	Low	Medium	**High**
Open Architecture (new technology)	No	**Yes**	**Yes**
Commercial Availability / Ubiquity of Components	Low	**High**	**High**
Software Ubiquity	Low	**High**	**High**
Implementation Difficulty	High	Medium	**Low**
Throughput	**High**	Low	**High**
Fault Recovery	Low	Low	**High**
Cost	High	**Low**	**Low**