© 1991 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Modern Computer Networks & Distributed Intelligence in Accelerator Controls

C. Briegel
Fermi National Accelerator Laboratory
Batavia, IL 60510

Abstract

Appropriate hardware and software network protocols are surveyed for accelerator control environments. Accelerator controls network topologies are discussed with respect to the following criteria: vertical versus horizontal and distributed versus centralized. Decision-making considerations are provided for accelerator network architecture specification. Current trends and implementations at Fermilab are discussed.

I. INTRODUCTION

Today's modern computer networks are characterized by "standard hardware protocols"; both proprietary and "standard software protocols"; and a variety of processor platforms. These platforms are characterized by heterogeneous processor families, operating systems, and I/O busses. The resulting system architecture must inter-operate and provide transparent network services between tasks. In general, this implies a peer-to-peer protocol enabling arbitrary messages to move through a network and arrive at its destination in a well understood form.

The architectural specifications for accelerator computer networks vary depending on the size of the machine, the worse-case amount of data to be retrieved, the performance expectations to the user, the monetary cost in support and hardware, and the supporting infrastructure of the laboratory. All these considerations must be evaluated to arrive at a reasonable solution. The data flow and its impact on the user are the most important requirements and should be either measured or modeled before committing to an architecture.

II. STANDARDS

Unfortunately, there are many types of standards: official standards such as ANSI, IEEE, IEC, CCITT, ISO; vendor standards such as SUN's SPARC; vendor committee standards such as LIM 4.0, EISA; *De Facto* standards such as Arcnet; and elephant (a big vendor) standards such as IBM's SNA, DEC's DECNet. Many standards such as IEEE 802.3 and IEEE 802.5 have jumped through several standard types before becoming official. While it may be easy and safe to use an official standard, the problem may best be solved by an alternative. The reality of standards is "standardization is good," but it must solve the problem and then the community must standardize on the selected standard. Utilization of many standards achieving the same goal, minimizes the effectiveness of standards. [1]

Standards are an attempt to bring order to a complex and volatile industry. While hardware standards such as IEEE 802.x has achieved a high degree of interoperability, software standards have had less success. For instance, there are currently 200 OSI standards spread over the 7 ISO layers. Software standards operate on a tight-rope between interoperability and flexible options which may distinguish one vendor from another. For instance, OSI standards consist of a "base standard," "International Standard Profiles," and "Functional Profiles." The International Standard Profile enables a standardization of standards as in the implementations of MAP and TOP (OSI-based standards). The Functional Profiles are an opportunity to tailor the standard for a specific user function. Conformance testing is the key to assure the interoperability of heterogeneous systems.[2]

Hardware leads software in both development and the standardization process. Previous architectures suffered from proprietary hardware which limited the connectivity among multiple vendors and increased its cost. As hardware standards have played an important role in integrating heterogeneous systems, so **will** standard software protocols and so **can** standard processor platforms. The standard processor platform is a controversial issue, since the user community has complete control but little global organization over these decisions. The decision to support a new platform should be based on need, innovation, and merit weighed against the implementation costs, rather than a user's preference.

III. NETWORK SURVEY

The following network survey is discussed with respect to implementation's in a control's environment.

A. ETHERNET

IEEE 802.3, (Ethernet) is currently the most mature and cost effective network implementation. The recent addition of 10baseT specification enables nodes to be easily connected similar to Token Ring. IEEE 802.3 specification is a CSMA/CD (collision sense multiple access/collision detection) protocol and generally operates at 10 Mbps. The disadvantage tö IEEE 802.3 is the decrease in network bandwidth for heavily loaded or synchronous networks. Networks can minimize the affect by segmenting traffic via bridges, but this solution implies a preferred communication path between nodes to isolate traffic to a local segment. Also, while the normal frame size for accelerator controls is nominally small (approximately 100 bytes), there are numerous applications for larger (<1500 byte) frames. Higher level software can provide larger frames via packeting protocols but with additional overhead and complexity. The advantages to IEEE 802.3 is its availability and support across

Operated by the Universities Research Association, Inc. under contract with the U.S. Department of Energy.

0-7803-0135-8/91$01.00 ©IEEE

multiple vendors, cost, implementation of standard protocols, and multicast support.[3][4]

B. TOKEN BUS/TOKEN RING

IEEE 802.4/IEEE 802.5, (Token Bus/Token Ring) are effectively the same concept. While the actual implementation details vary, Token Bus is simply a logical Token Ring on a physical bus. In either case, the token is passed from one node to the adjacent node providing an equal and predictable access to the medium. The most important factors distinguishing the two are cost and support. IEEE 802.4 has limited support compared to the IEEE 802.5. The disadvantage of IEEE 802.5 versus IEEE 802.3 is its maturity resulting in fewer choices for bridges, controller boards, and higher-level software. Also, the multicast functionality of IEEE 802.5 is limited to 32 addresses. Since IEEE 802.3 supports a larger set of multicast addresses and preceded Token Ring, the utilization of multicast addresses in existing higher level protocols pose a mapping problem for corresponding Token Ring implementations. The advantages of IEEE 802.5 are the effective bandwidth, predictable access, and relatively large frame size.[3]

C. FDDI

FDDI, (Fiber Distributed Data Interface) is the utilization of fiber either as a network protocol between nodes (a standard in the final stages of approval), the medium for existing protocols such as IEEE 802.3 or IEEE 802.5, or as a multiplexer of numerous protocols. All of these uses are confusing lumped into the acronym, FDDI. Today, the primary use of FDDI is as a backbone network which ties multiple network segments via bridges or concentrators. The use of FDDI as a network protocol is still restricted to mainframes or workstations with relatively high controller costs. While the controller cost are dramatically being reduced by some vendors, one should still be skeptical about the interoperability with another vendor's controller, concentrator, or bridge. While it remains difficult to fiscally justify and procure controllers for a heterogeneous network, fiber can be used as a cost effective medium for existing protocols and anticipates the higher bandwidth requirements of future applications.

D. TCP/IP and OSI

The choice of a standard communications protocol is highly desirable. The functionality required can be as simple as a connection-less datagram service or a more elaborate protocol such as a connection-oriented remote procedure call. Each protocol has corresponding overheads which may or may not be acceptable in your application. While a connectionless protocol does not guarantee communications, a connection-oriented protocol provides end-to-end reliability.

The *De Facto* standard, TCP/IP, is the most widely supported and complete set of protocols. It provides a layered architecture enabling connectionless (UDP, User Datagram Protocol) and connection-oriented (TCP, Transmission Control Protocol) communications with an underlying network routing (IP, Internet Protocol) layer. Unfortunately, OSI protocols are still being refined and lack implementations across a wide variety of platforms. Unless the set of processor platforms is restricted in the network, TCP/IP is the appropriate protocol today.

TCP/IP is not a software implementation standard, but a protocol standard. It is an artifact of the portability of the UNIX operating system that many implementations have a similar calling sequence. If required, portability can be achieved by encapsulating each implementation with a common calling sequence.[5]

E. BRIDGES

Bridges are used to tie multiple networks together. Current bridge development is attempting to combine the functionality of routers and bridges by implementing a SRT (Software Routing Transparent) protocol independent of network and transport protocols. This implies the source node does not necessarily know where the destination node resides. Further, routing information provided by network protocols such as IP are not required and can be ignored by the bridge. The lack of specific routing software for each supported protocol will make bridges simpler to operate, more efficient, and more robust.

IV. ARCHITECTURE TOPOLOGY

A. VERTICAL VS HORIZONTAL

Most network architectures are logically similar. The complication is achieved by including every last detail and specification of the implementation. The similarity can be viewed if the network is perceived as horizontal lines with various levels of connectivity. The addition of a horizontal line in the architecture is due to a latency caused by either a hardware or software protocol change which is not transparent to the network. For instance, an additional horizontal line is required if a gateway is placed between two networks. A gateway imposes a delay due to the mapping of one protocol to another while bridges essentially provide transparent access times to nodes. In many control architectures, software gateways are required to map two dissimilar data acquisition protocols. The result of too many horizontal lines is a vertical network with corresponding latency delays to the user. Data pools are a method of reducing a vertical network by making the data appear at a higher level. Data pools are not always a viable solution depending on the update rate, size of the pool, and synchronization or acknowledgement of a data setting or reading

B. DISTRIBUTED vs CENTRALIZED SERVICES

Distributed services are sometimes necessary for critical closed-loop control when a portion of the control system is not functioning. Distributed services can be used for reduction or real-time manipulation of data which can only be acheived with distributed processing. While the choice between distributed and centralized services in similar cases is clear, other implementations of distributed services are less clear. For instance, distributed services can be used to provide service to each level of a vertical network. While this can logically eliminate a vertical network for a distributed service (distributed data base), many services (data retrieval) can not span each level. Also, while distributed services can enable parallel access, they can also impose a severe overhead to the requestor of the service (a console requesting data retrieval from many nodes). In general, centralized services are convenient and more manageable (centralized data base), but imply potential bottlenecks. A centralized service can easily become compute or I/O bound by the number of requestors. Today's control's architecture should provide a tuneable mixture of distributed and centralized services. Tuneable implies the service can easily migrate from one processor to another or improve performance with the addition of the same service on another processor.

V. FERMILAB IMPLEMENTATIONS

A. TOKEN RING and ETHERNET

Fermilab has implemented an IEEE 802.5 network to connect consoles with front ends for the reading and writing of hardware attached to the front ends. These front ends are connected to the hardware via various methods such as CAMAC, Arcnet, 1553B, VME, MultibusII, GPIB, or RS232. The Token Ring is the primary data acquisition network. It enables a direct connection between consoles and most nodes. The IEEE 802.3 network connects consoles and provides general network services such as file manipulation.

B. ACNET and DECNet

A IEEE 802.3 network implements DECNet protocol and is used to interconnect consoles taking over higher level protocol burdens specific to the consoles such as program staging. The IEEE 802.5 network was implemented with a proprietary task-to-task message passing protocol, ACNET (Accelerator Network). ACNET is a request/reply calling sequence between cooperating tasks. This protocol will be maintained until a standard network protocol can be purchased for all platforms on the Token Ring and proven to perform adequately. At that time, the ACNET user calling sequence can access the network and transport layers of the new software protocol transparently to the user.

C. PROTOCOL SPECIFIED SERVICES

Fermilab services such as data base, file share, alarms, data acquisition, fast time plot, etc. have been specified by documenting a protocol within an ACNET message. This method provides a mechanism for layering and modularizing the service. Implementations can proceed based upon a well-defined interface similar to the arguments to a function call. The cooperating tasks, the requestor and replier, can be implemented in parallel and on multiple platforms by sending the correct protocol to the network. Further, a degree of service portability is achieved since ACNET has a common calling sequence as do many services. Rather than using a data representation protocol such as SUN's XDR, Fermilab has defined the protocols with respect to its appearance on the medium. Thus it is up to each node to provide the correct byte, word, longword, and floating point orientation as defined by each service protocol. To minimize data manipulation, this implicit protocol is optimized for the majority of our platforms and data types (integer).

D. PROCESSOR PLATFORMS

Fermilab currently supports a generic set of services on the following platforms:

Processor	Operating System	I/O Bus
VAX	VMS	Unibus/Qbus
VAX	VMS	SCSI
PDP11	RSX11M	Unibus/Qbus
68K	MTOS	VME
68K	PSOS	VME
I386	MTOS	MultibusII

Future plans for generic support include a variety of Unix platforms. The multitude of platforms has a tremendous effect on support services at Fermilab. While the addition of future platforms is inevitable, support can be minimized by implementing the generic services in portable languages and maintaining consistency in the above categories.

V. REMARKS

Fermilab goals for network and distributed intelligence is to move the data closer to the user on a console, provide an open network for a variety of processor platforms, and supply a generic set of protocol services. While these goals have been successful, Fermilab has several areas of concern. First, the multitude of processor platforms are continuing to grow. The utilization of a portable language does not completely mask implementation details of an operating system or bus structure. Second, utilization of software standards must be included into the architecture with minimal impact to the users. This can be achieved by encapsulating the standard with the current calling sequence. A viable network protocol is currently not commercial available for all supported processor platforms.

VI. REFERENCES

[1] D. Danielle, "There Is No Standard for Standards," Network Computing, March, 1991, pp.100-101.

[2] D. Ledrick, M. Spring, "Standards: Bringing Order to Complex and Volatile Industry or Stumbling Block to Creativity," Computer Standards and Interfaces, V11, pp.95-103,#2, 1990.

[3] D. Hutchision, *Local Area Network Architectures*, Addison-Wesley Publishing Co., 1988.

[4] W. Stallings, "Local Networks," ACM Computing Surveys, pp. 3-41, March, 1984.

[5] C. Comer, *Internetworking with TCP/IP VI: Principles, Protocols, and Architecture*, Prentice Hall 1991.

[6] C. Briegel, et al, "The Fermilab Acnet Upgrade", in *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems*, Vancouver, Canada, pp. 235-238, October 1989.