CONSIDERATIONS FOR DISTRIBUTED INDUSTRIAL CONTROL SYSTEMS

P. D. GRIEM, JR. and J. W. BERNARD
Research Department of the Foxboro Company
Foxboro, Massachusetts

Summary For a variety of reasons, industrial plant control systems have been increasingly complex and expensive. Economic and reliability considerations have initiated a trend toward distributing control system functions among communicating components, utilizing the new microprocessor and communication technologies.

Any new technology brings new capabilities, but also new engineering design considerations and trade-offs. This paper attempts to outline some of the important system design issues.

Introduction The history of industrial process control system technology can be viewed as a succession of phases, each triggered by some major advance in component technology. Each new phase was accompanied by promises of new capabilities, lower costs, higher reliability, and easier maintenance. For the most part the promises of each new technology were fulfilled, but industrial control engineers found to their dismay that each new technology also brought with it unanticipated new problems.

Let's take a brief look at the previous phases of industrial control system technology.

Twenty years ago, control instrumentation was distributed throughout the industrial plant, and in areas of high concentration was often collected onto local control panels for the convenience of plant operators. Most of the instruments were pneumatic, but some were hydraulic or electric. There was little communication between control panels because of the difficulty of transmitting pneumatic or hydraulic signals over long distance, and the expense and relatively low reliability of electric devices based on vacuum tubes (remember them?).

Then came the transistor, which made possible solid-state electronic instrumentation, which in turn led to reduced control panel size and replacement of cumbersome pneumatic switches, relays, and gauges with less expensive and more reliable electric equivalents. The ability to transmit electric control signals reliably over long distances led to control connections between different plant areas and consolidation of local control panels into

centralized control rooms, which often reduced the number of people required to operate the plant. On the other hand, a major reason for centralized control rooms might also have been that solid-state electronics didn't last too long in hot, wet, and/or dirty plant environments. Another problem was that each instrument manufacturer used a different electric control signal, making interconnection difficult. And to master the engineering and maintenance of a system required an education in grounding, electrical noise, wiring practice, and multimeters--not that this was difficult, just different.

By 1965, further advances in semiconductor technology made possible digital computers with significant new functional capabilities: process monitoring and alarming, supervisory control, direct digital control, control by mathematical models and linear programs, data gathering and reporting. These machines were rather delicate mechanically and electrically, and could only survive even in a control room environment by adding several tons of air conditioning. (Fortunately, the previous phase of industrial control technology had provided control rooms.)

Operation and maintenance required an education in parity checks, diagnostics, circuit boards, and oscilloscopes, as well as reeducation in grounding, wiring practice, and electrical noise (especially static electricity). System engineering required careful planning of core, drum, and disk sizes, as well as provision for failure of the computer (and the air conditioner) through redundancy or backup equipment.

Software was mainly the concern and responsibility of the computer vendor. The cost of a system was sufficiently high (typically 500,000 pre-inflation dollars) that a few man-years of programming did not make an appreciable difference in system cost.

And there was little commonality in hardware or software between vendors, making it difficult to interconnect systems.

About 1970, minicomputers began arriving in large numbers as a result of integrated circuit technology. Minicomputers really didn't offer any new functional capability, they just cost a lot less. They also came without much software, and customers found they had to learn about interrupts, device drivers, power failure/restart, assemblers, linkers, and loaders. There was still not much commonality of hardware or software, except that nearly all minicomputers supported FORTRAN.

Also about 1970, communications technology began growing rapidly. First, asynchronous, then synchronous communication links were established between industrial plant computers and central corporate business computers. Industrial control engineers had to become familiar with baud rates, protocols, modems, leased lines, dial-up

networks, and IBM 360's. The lack of effective standards in communications is well-known.

Today, microprocessors and communication technology have made possible distributed control systems, which promise greatly improved reliability, flexibility, and cost-effectiveness; they are now being installed in industrial plants. Perhaps it would be wise to try to look ahead, and see what new considerations distributed systems might bring.

Industrial Control System Requirements It is important to keep in mind the purposes of industrial control systems. In the final analysis, improved control system technology is not an end in itself, but only a means to improved plant profits.

Comparing a new control system technology with an existing one to determine whether any profit improvement will occur involves answering two major questions:

1. Will the new technology improve productivity? That is, we must judge whether the new technology is likely to cause more product to be produced, or higher quality product to be produced, or both. This can be done by examining the functional capabilities to determine whether existing control functions can be performed better in any sense (faster, more accurately, etc.), or whether the new technology makes possible new functions which were previously impossible or uneconomic to implement. An analysis of this sort involves a careful definition of the functional hierarchy of the system and the interfaces between each level; approaches are given in (1,2). It is usually worthwhile to make such an analysis when considering new technologies, as many functions of the existing technologies might otherwise be taken for granted and ignored. For example, the successive phases of digital computers have not yet eliminated analog strip chart recorders.

2. Will the new technology reduce overall system design, installation, operation, maintenance, and modification costs? This question will be pursued through the remainder of this paper as various aspects of distributed systems are examined.

Similar concerns were raised at the 1975 International Telemetering Conference (3,4). In general, there will be in each area an initial cost or investment for the industrial user to learn the requirements of the new technology (i.e., how different is it), plus the ongoing costs to apply that knowledge.

Architecture The first problem in distributed system design is to determine how to distribute the system functions among the various physical units. There is a similar problem in a system based on a central computer or minicomputer, that of allocating the system functions to specific programs within the computer and deciding the priority and

residency (core or bulk storage) of each program. However, in a central computer, those decisions may be more easily changed than in a system of distributed smaller processors, since each processor is likely to be designed in accordance with its intended functions.

The choice of distribution of functions depends on the requirements of the industrial application. The most common requirement is the need to guarantee an overall system response time in a certain situation, for example, closed loop control, critical process interrupts, or in supplying data values to a process operator console (especially if the data has to be collected from several other devices). Unfortunately, "scheduling and resource allocation become more difficult in a distributed environment than in a single processor environment" (4). The main reason is that the many tasks in a single processor system do not really operate simultaneously, but rather execute one at a time under control of a common operating system which controls allocation of common resources according to strategies which can be tailored to a specific application. A distributed system often has little or no central control, which is intentional, since a failure of the central control unit could fail the entire system.

The decision on functional distribution is very tightly related to the second major architectural problem of distributed systems, which is how the physical units should be interconnected--many ring, star, bus, and network schemes have been proposed. There are many possible solutions to both problems, and each has different advantages and disadvantages from the viewpoints of reliability, flexibility, system overhead, and failure modes. A really excellent analysis and comparison of the various interconnection schemes is given in (5).

Assuming that the functional requirements are met, the major architectural consideration becomes the effect of gross failures in the system. This concerns total failure of a physical unit or communication path on the overall industrial control functions. More subtle failure modes are discussed below.

Communications The heart of a distributed system is its physical communications system. Its characteristics have a large influence on the reliability, flexibility, and performance of the distributed system.

Actually there are two levels of communication to be considered (7,8):

1. Low-level communication between physical units through the physical communication system; and

2. High-level communication between distributed control functions.

To illustrate, suppose we have a distributed system with some number of devices connected to a communication network:

(a) An operator console application function resident in device 06 wishes to invoke a control function resident in device 13 for the purpose of filling a tank, so it initiates a high-level communication transaction to send a message to the control function.

(b) The low-level communication system takes the high-level message and adds low-level information (such as device addresses and data check bits) to form a low-level message. The low-level message is then transmitted to device 13, which contains the control function.

(c) If the low-level transaction is unsuccessful for some reason, the low-level communication system will try it again; however, the high-level application functions will hot be aware or involved in the retries. If the low-level transaction is successful, low-level acknowledgement is transmitted from device 13 to device 06 to the operator console function.

At this point, the low-level transaction is complete, but the high-level transaction is not complete, since the operator console function has not yet received the expected high-level acknowledgement from the control function. Suppose it never comes--how long should the operator console function wait before giving up? And what action should it take next, since it doesn't know whether the control function (1) did not receive the message, or (2) received the message, filled the tank, but was unable to reply? If the second situation existed, an attempted retry of the highlevel communication might result in an attempt to fill the tank twice.

The points to be made from the above example are that two levels of communication exist in a distributed system, each with its own protocol and problems, and that ambiguities may occasionally arise which are impossible to resolve, at least within a short time frame (7). In fact, as pointed out in another excellent recent paper (6), the common practice of placing a time limit on transactions to prevent infinite waits actually increases the number of possible ambiguities.

Both levels of communication also have to cope with the problem of a device or function which transmits messages when it is not supposed to, thereby interfering with the communications of others. This is known as the "babbling idiot" problem.

Design of the Distributed Modules Each module should, as much as possible, be designed to carry out its application functions even in the event of failure or loss of communication with other modules in the system, perhaps by automatic functional

reconfiguration (4). However, as was discovered at Bell Laboratories (7) and on the ARPANET, this characteristic may make it virtually impossible to turn the system off!

Eventually a failed module will be repaired (or somehow heal itself), and it must then be brought back into the system. High-level and low-level resynchronization information may have to be communicated in both directions between the restored module and the rest of the system. It is apparently very difficult to do this automatically; many systems require manual intervention and reinitialization of the entire system. In the industrial environment, however, automatic restoration is usually required, which means that a module must be designed to store, transmit, and receive such information when necessary.

Handling power failures and restarts involves similar considerations. In a single computer system, the whole system stopped if the power to the computer failed. In a distributed system, especially if it is dispersed over a wide area, some of the modules might experience a power failure while others do not. Thus the design of the distributed system modules must consider the possibility of randomly losing and regaining communication with any or all of the other modules.

In a single processor system all tasks can be synchronized against a single clock; in a distributed system, many of the modules may have individual clocks, so periodic resynchronization of the system with respect to clock time is another consideration.

Instrumentation of Distributed Systems How does one determine the state of a system of many distributed processors? Which modules or communication paths are alive, dead, or sick? Single processor systems have a single hardware panel on which the status of all the processors in the system (i.e., one) can be displayed, and often have an on-line utility program which can report on and allow interaction with the scheduling of all the tasks in the system. With a distributed system, particularly if the modules are widely dispersed, this need would seem to be even more important (9).

Good instrumentation practice would require that such instrumentation be totally independent of the distributed system itself, but this would mean in effect installing a second distributed system, which would be uneconomic. Where such instrumentation is provided, it will probably use portions of the distributed system being measured (certainly the communication system), so it is important to understand what kinds of system failures also cause a failure of the instrumentation.

Modification of Distributed Systems Very often multiple copies of application and system configuration information reside in several different modules, and often in different forms. This is done to improve the operating performance of the system, by avoiding the need to transmit such information continuously to all modules which need it. (This

situation also occurs within single processor systems, but to a lesser extent, since the information can be placed in common core or bulk memory which can be rapidly accessed by all programs.)

Sooner or later it will be necessary to modify or augment such information because of application or design changes, and the implications of updating multiple copies have been previously described (10). Ideally this would be accomplished by catching the entire system in a quiescent state, freezing it temporarily while all the multiple copies are modified, and then releasing the system to operate on the new information. This is actually what is done on existing single processor systems, usually in a few seconds or less, since one task can easily take over the whole system, examine the status of all other tasks, and if an acceptable state is found, make the modifications. In a distributed system of independent processors, however, this may be much more difficult for reasons mentioned above, so the mechanism for accomplishing such changes should be an important consideration.

A related consideration is how to make modifications to the software in any particular physical unit of a distributed system. With central computer systems, operations such as generation, software configuration, addition or deletion of programs and files, and patching could be performed on-line or off-line with varying degrees of ease, depending on the size and sophistication of the system. Most processors in a distributed system will probably not have these capabilities, but there may exist one processor on the system capable of generating and “down-loading” software for the others. The extent of the software modification capability, the mechanism for effecting the changes (especially if many copies of the same software exist in the system), and the documentation control are still important considerations.

When adding new modules to the system, it is necessary to make a now physical connection into the existing communication network. How this is accomplished, and any possible disruptions to the system, should be considered.

Finally, if it is desired to interconnect distributed systems of different manufacture, the previous experience with new technologies would ,lead us to expect that this would not be easily accomplished.

Conclusions Distributed control systems offer many potential benefits to industrial plants, and we are all eager to partake of them. To ease the transition into this new technology, it is helpful to try to anticipate any new considerations for installation, operation, maintenance, and modification. Since there are relatively few installations at present,- additional experience will reveal aspects which were overlooked in this paper. These. will probably present even more interesting design challenges in the future.

REFERENCES

1. M. C. Beaverstock, “3 R’s of Control System Design,” Third Annual Control Engineering-Purdue Advanced Control Conference, Purdue University, April 26-28, 1976.

2. J. W. Bernard and G. M. Howard, “Organizing Multi-Level Process Control Systems,” IEEE Systems and Cybernetics Conference, San Francisco, October 16, 1968.

3. D. J. Karleskint, “The Common Element Approach to Telemetry Processing Systems,” Proceedings 1975 International Telemetering Conference.

4. R. L. Larsen, “A Distributed Systems Approach to Mission Operations Support,” Proceedings 1975 International Telemetering Conference.

5. G. A. Anderson and E. D. Jensen, “Computer Interconnection Structures: Taxonomy, Characteristics, and Examples,” ACM Computing Surveys, Vol. 7, No. 4, December 1975.

6. E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber, “Some Constraints and Tradeoffs in the Design of Network Communications,” Proceedings 5th Symposium on Operating Systems Principles, University of Texas at Austin, November 19-21, 1975, published in ACM Operating Systems Review, Vol. 9, No. 5.

7. A. G. Fraser, “A Virtual Channel Network,” DATAMATION, February 1975.

8. H. Halling, “A Distributed Control System Using the CAMAC Serial Highway and Microprocessor Modules,” Proceedings 1975 IFAC/IFIP Workshop on Real-Time Programming, Boston, August 21-22, 1975.

9. “Distributed Systems: General Discussion,” Proceedings 1975 IFAC/IFIP Workshop on Real-Time Programming, Boston, August 21-22, 1975.

10. P. D. Griem, Jr., “On the Principle of Unique Definition,” Proceedings 1975 National Computer Conference, Anaheim, California.