Real-Time Open Control Architectures and System Performance

Yoram Koren[1] (1), Zbigniew J. Pasek[1], A. Galip Ulsoy[1] (2), Uri Benchetrit[2]
[1] The University of Michigan, Ann Arbor, MI, USA
[2] Technion, Haifa, Israel
Received on January 3, 1996

Abstract
This paper analyzes the effect of control architectures and communication networks on a manufacturing system's performance in terms of part precision and productivity; the network bandwidth requirement for a distributed control system is also included. The objective is to design the system such that the control and communications (both hardware and software) would not be the limiting factors in system performance. For simplicity we analyze the performance of a machining center control system. The base-line for comparison is a conventional computerized numerical controlled (CNC) with discrete event management/adaptive system.

Keywords: Control, Machining, Real-time

1. Introduction

Motivation. In the recent years, there are increasing efforts around the world to introduce open-architecture systems for industrial controls. The main efforts are being carried out in the USA, Germany, and recently in Japan. The drive towards the open systems is motivated by the need to implement a base of systems capabilities that is reliable, economical, and provides a stable foundation for adding more functionality as controls needs grow and change. Open systems are the only path for implementing distributed systems, in which complexities of distributed computing and multi-vendor environments play a major role.

Other Research. The major research efforts in the area of open architecture control (OAC) systems include the following:

- The OSACA (Open System Architecture for Controls within Automation systems; ESPRIT III project 6379) project [1] may be one of the largest-scale projects for OAC, in which almost all of standardization matters including networking, application software as well as hardware, have been considered.
- The National Institute of Standards and Technology (NIST) proposed and used the RCS (Real-time Control System) reference model architecture over the past 15 years [2].
- The Next Generation Controller (NGC) Program, based on the RCS reference model, co-sponsored by the National Center for Manufacturing Sciences (NCMS), the U.S. Air Force and Martin Marietta, organized industry requirements and prepared a specification for an open systems architecture standard (SOSAS) [3].
- The Enhanced Machine Controller Architecture (ECA) is the next step beyond NGC/SOSAS by NIST. In the ECA project, an open machine tool has been implemented based on the NGC/SOSAS and RCS reference model [4].
- Other research projects like the Chimera project at Carnegie Mellon University [5], the Multiprocessor Database Architecture for Real-Time Systems (MDARTS) [6] at the University of Michigan, and the Hierarchical Open Architecture Multi-Processor Motion Control System (HOAM-CNC) [7] at the University of British Columbia, have demonstrated a variety of approaches to the OAC.

Evolutionary Testbed Controller. Research on the next-generation CNC controllers has been conducted at the University of Michigan for a number of years [8, 9, 10]. To effectively perform research in that area, an open and readily modifiable cotrol system was needed - features, which were not posessed by any of the commercially available CNC systems. Hence, an original experimental controller testbed was created. The original system configuration consisted of: (a) a 5-axis CNC milling machine, (b) a general purpose Intel i486/33MHz computer, (c) multiple sensors, (d) multiple sensor interfaces, (e) commercial CNC controller. Over the past two years two new elements have been added: (f) open-architecture VME-based real-time controller, (g) a DSP-based multi-axis controller.

While providing openness necessary for research, our experimental system (see Fig. 1) exhibited a number of drawbacks. Its performance would vary, depending on the programmer's skills; for example, execution times of the subroutines are a function of the length of the code. Therefore execution of critical real-time tasks cannot be strictly enforced. This issue becomes even more important with the computational load increased by a growing number of involved control routines and their complexity. Also, change of the control algorithm required recompilation of the whole source code which, in turn, changes the interrupt frequency, as discussed below.

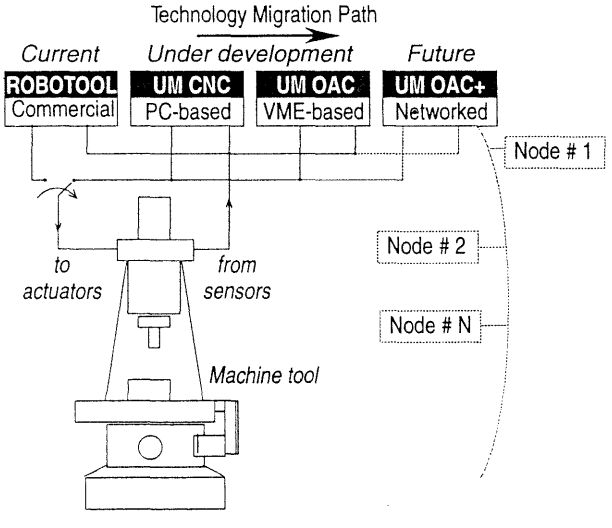

Figure 1. UMOAC Hardware Configuration

2. Basic Controller

The basic computerized controller for production machines may contain four types of algorithms, as discussed below.

A. Adaptive Compensation and Event Management. Adaptive Compensation and Event Management system may contain several types of algorithms that can improve the system performance in terms of (i) productivity; (e.g., feed adaptation to cutting force measurements with Adaptive Control Constraint -ACC- algorithm), (ii) part quality and precision (e.g., error compensation due to changes in machine temperature); and (iii) system reliability (e.g., if oil temperature exceeds a certain level, stop the machine). Inputs to these algorithms are either continuous measurements (e.g., temperature, force, etc.) or discrete events that do not need an immediate response.

B. Interpolator. The interpolator coordinates the motion of the individual machine axes to achieve a desired spatial trajectory with required precision. The interpolator operates at certain time intervals, during which the next interpolation step calculation is performed and a new position command is sent to the servo-control loops. The interpolator time interval cannot be smaller than the time interval during which the servo control executes its algorithms.

C. Servo Control. The servo-control loops operate at fixed time intervals [8]. Each loop compares the command received from the interpolator with its position feedback, and sends a velocity command to the motor to drive the corresponding machine axis.

D. Emergency Control. Emergency control responds to discrete events that require immediate attention such as stopping the machine in emergency situations (e.g., pressing a limit switch). It has the highest priority and must override any other control operation.

Timing. All of the above four levels may be executed with a single microprocessor at constant time intervals T. The time T is adjusted according to the worst case, namely, the longest possible cycle to execute successfully all algorithms. If the worst case for the Adaptive Compensator is T3, and one for the Interpolator is T_2, and the one for the Servo Controller is T_1, the time T is the sum of these three times. However, the execution times of the Adaptive Compensator, the Interpolator, and the Servo Control are not necessarily equal at each iteration. For example, the time slot given to the control loops is based on the assumption that all machine axes move simultaneously, even if this case only rarely occurs.

The position resolution, D, with this control architecture is given by the equation

$$D = VT \qquad (1)$$

where V is the velocity along the trajectory (i.e., the tool velocity in a milling machine). For example, if V = 40 mm/sec and T = 5 msec, then D = 0.2 mm. During this D = 0.2 mm interval the system actually operates in open loop and cannot make corrections to disturbances (such as cutting forces). The longer this period is, the worst the repeatable precision that can be obtained by the system. The designer would like to keep the period T as small as possible, but this, in turn, depends on the complexity of the algorithms, the speed of the control computer, and the total number of controlled axes.

3. Hierarchical Controller

Hierarchical controllers provide different rates of execution (i.e., sampling rates) for each algorithm type, where the rates are adjusted according to the priority of the algorithm and its worst-case execution time. This section discusses an hierarchical controller that is controlled by a single microprocessor. As mentioned above, the controller contains four types of algorithms. The first three algorithms are executed at different rates coordinated by a programmable clock. The Emergency Control algorithm is executed by a priority event-driven interrupt. Figure 2 shows an example in which the main clock provides three clock signals to the three algorithms, where

$$f_3 < f_2 < f_1$$

For example, f_3 = 10 cps, f_2 = 100 cps, f_1 = 1000 cps. In this example the adaptive compensation algorithm is executed every 100 msec, the interpolator every 10 msec, etc.

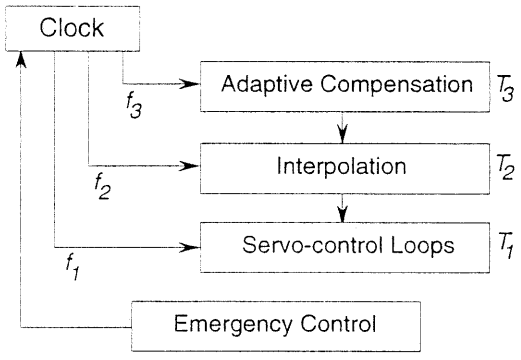

Fig. 2 Hierarchical Controller on a Single Microprocessor (f - frequency of execution; T - execution time)

The time slots given to each algorithm: T_1, T_2 and T_3 correspond to the worst-case execution times of each algorithm. It is obvious that the following conditions must always be satisfied:

$$T_i < 1 / f_i \qquad i = 1, 2, 3 \qquad (2)$$

Since all algorithms are executed by one processor an additional condition must be satisfied

$$T_3 + T_2 (f_2/f_3) + T_1 (f_1/f_3) < 1/f_3 \qquad (3)$$

Namely during the time period $1/f_3$ the adaptive algorithm is executed once, the interpolator is executed $b(f_2/f_3)$ times, etc. Equation (3) can be written as

$$T_3 f_3 + T_2 f_2 + T_1 f_1 < 1 \qquad (4)$$

For example, if T_3 = 4.0 msec, T_2 = 2.0 msec, T_1 = 0.7 msec, using the values of f_i given above, we obtain

$$4 \times 10^{-3} \times 10 + 210^{-3} \times 10^2 + 0.7\ 10^{-3} \times 10^3 = 0.94$$

and the condition in Eq. (4) is satisfied. However, for example, if T_1 = 0.8 msec (instead of 0.7 msec), the condition is violated, and the control system will not operate properly. The options in this case are either to reduce the frequencies f_i or to use a multi-processor approach. Reducing the frequency deteriorates the resolution, as explained below.

The Resolution. The basic controller has a defined resolution. By contrast, the hierarchical controller has three types of resolutions, each corresponding to one of the basic algorithms.

Position Resolution is inversely proportional to frequency f_1,

$$D_1 = V/f_1 \qquad (5)$$

For example for f_1 = 1000 cps and V = 40 mm/sec, D_1 = 0.04 mm.

The relationship between T in Eq. (1) and the worst-case execution time for the servo-control algorithm, T_1, is given by

$$T_1 + T_2 + T_3 \cong T \qquad (6)$$

which means that $T_1 << T$ and in practice

$$1 / f_1 < T \qquad (7)$$

Therefore, the position resolution in Eq. (5) is smaller than the one in the basic control system given in Eq. (1).

Interpolation Resolution. Unlike the basic control system, the hierarchical system also has resolution dictated by the

interpolation as given by

$$D_2 = V/f_2 \quad (8)$$

where V is the velocity along the trajectory.

Adaptive Compensation Resolution. A third type of resolution that inherently exists in hierarchical systems is the Adaptive-Compensation/Event-Management resolution, that can be done at time intervals of $1/f_3$, which are translated to position intervals of

$$D_3 = V/f_3 \quad (9)$$

Namely, only when the tool passes D_3 mm, the controller can execute, for example, temperature compensation, correct feed, estimate tool wear, etc. At large velocities V, a relatively large value of f_3 may cause problems. For example, if an error compensation due to changing machine temperature must be done at shorter distance intervals to maintain a certain level of precision.

Changes in the Program. A major drawback in both the basic and hierarchical controller is that adding code because of a change in the algorithm requires re-adjustment of the iteration clock. This reduces the flexibility of reconfiguring controllers according to customer needs and applications.

To further emphasize the last point, note that the times T_1 and T_2 that are needed to execute the servo-control and interpolation algorithms, respectively, are proportional to the number of axes on the machine. In many cases also T_3 increases with the number of axes. Adding a physical axis-of-motion requires therefore a major change in both the algorithm and the timing. The latter, in turn, affects resolution and precision as discussed above.

4. Multi-Processor Controller

Increasing the computational resources of the control system might be an alternative to the previous two controllers. One possible architectural structure consists of:

Processor 1: Servo-loops
Processor 2: Interpolator
Processor 3: Adaptive Compensator and Event Management.

An alternative structure combines the servo and interpolation in a single processor [7].

In each case an additional processor is needed to manage the information flow and store the part program. In terms of hardware, two possible solutions are:

I. Each microprocessor, which may be a Digital-Signal-Processor (DSP), is on a separate board, plugged into the bus of the main computer (e.g., PC) that stores the part program.
II. Distributed control system, where the various microprocessors are connected by a communications network.

With distributed control systems, intelligence and control functions can be moved out of central control units into controllers located near the controlled device. Devices with microprocessor located at the point of measurement or final control (e.g., on the motor) are being developed. They can improve signal processing and communication of the measured information. Control that is being delegated to devices may have embedded intelligence loops that can fit the application and locally adapt to process changes.

The key element that distinguishes network for distributed control from other networksis the capability to support real-time applications. Other networks that are used for applications such as electronic mail, sharing printers, file transfer among multiple users, etc. do not have the hard real-time constraint. Even distributed control systems might have different real-time restrictions. For example, if an AGV arrives at a loading station and sends a signal through the network, this signal might be delayed by a second or two. We call it a soft real-time constraint. However, if two robots are assembling a part simultaneously, synchronization signals must be transferred immediately. This is called hard real-time constraint. Factory communications networks are shown in Fig. 3, which depicts several autonomous units (such as a machine tool, an AGV and an assembly station with two robots coordinating the work) connected via a network with soft real-time constraint. The communications within each unit are done with a network or a bus with hard real-time constraint. Such architecture provides the flexibility to accommodate both types of constraints.

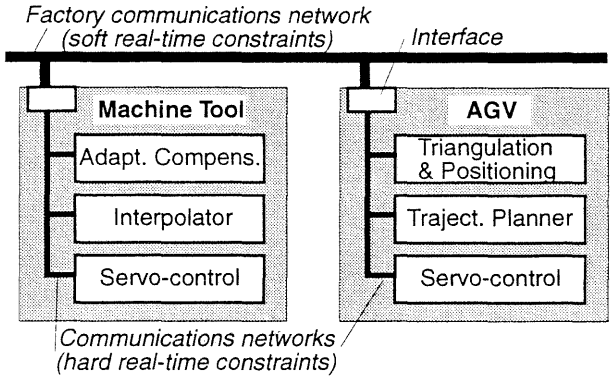

Fig. 3 Distributed communications network with soft and hard real-time constraints

The question is what is the rule that guarantees that the network has enough capability (in terms of bandwidth) to support several microprocessors requiring hard real-time constraints, such that the machine performance will not deteriorate.

Let us elaborate on the last issue with the aid of an example. A distributed control system with three microprocessors and one main computer is depicted in Fig. 6. There is a fixed and known order in the direction at which information is propagated: (III) always provides data only to (II), and (II) to (I). By contrast, in soft real-time communications networks the access to the network is arbitrary, and each station can send messages to any other station. In most cases inside the autonomous unit the information propagates in a fixed and a prior known order, and the designer might take advantage of this knowledge and implement a pipeline approach (Fig. 4 shows a point-to-point pipeline). Each internal unit (which has a dedicated processor) has input and output buffers (e.g., unit II in Fig. 4). When the output buffer is ready, it signals the next unit for the availability of the data; the next unit may retrieve the data when available, and sends acknowledgment to the previous unit. This is an asynchronous approach of a pipeline. The advantage of a pipeline approach is the small overhead in terms of time needed to prepare a medium access protocol and the extra code that must precede and follow data in regular communications networks (to indicate destination, data length, operation code, etc.).

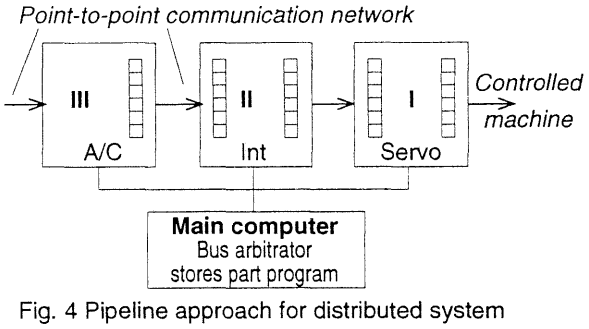

Fig. 4 Pipeline approach for distributed system

The asynchronous pipeline approach might have three architectures: Point-to-point (the one shown in Fig. 4), a common bus (e.g., VME bus or PC bus), and a common network (also called backbone). The latter might have two possible architectures: Token ring network (Fig. 5) and a regular network one example of which may be the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) [described in IEEE 802.3 standard]. The latter architecture, shown in Fig. 6, is discussed below.

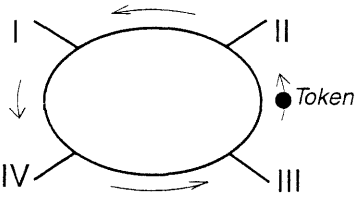

Fig. 5 A token passing ring real-time network

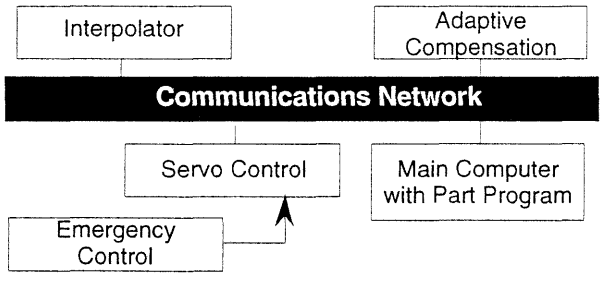

Fig. 6 Processors communicating via a local network

5. Network Bandwidth

In designing a distributed control hierarchical architecture, a new constraint must be accommodated - the limited bandwidth of the communications network. There are three factors that affect the load on the network: (i) the sampling rates at which the various microprocessors send information to the network; (ii) the number of entities that require synchronous operation (e.g., number of controlled axes requiring synchronization), and (iii) the size of the information packet (more accurate machines might require larger packets).

To analyze the effect of this constraint on the controller performance we assume that information from and to the various microprocessors is sent serially at packets consisting of 16-bit command (or data) and additional 40 bits header and trailer.

For simplicity we may assume that there is only one message at a time on the network and that the physical length of the network is relatively small. During the execution of the segment the average frequency of information on the network is

$$f = (f_2 + f_3)\,(40 + 16N) \qquad (10)$$

The network bandwidth f_W must be larger than f, i.e.,

$$f_W > f \qquad (11)$$

Combining Eqs. (10) and (11) yields an upper bound on $(f_2 + f_3)$:

$$f_W \geq (f_2 + f_3)\,(40 + 16N) \qquad (12)$$

The frequencies f_2 and f_3 are given by

$$f_2 \leq 1/(T_2 + DT) \qquad (13)$$

$$f3 \leq 1/(T_3 + DT) \qquad (14)$$

where DT is the overhead time. Eqs. (12) - (14) express the constraints in selecting the sampling frequencies f_2 and f_3, when the network bandwidth f_W and the times T_2, T_3 and DT are given. Alternately, the bandwidth can be calculated by these equations when the sampling frequencies and times are given.

Example. Let us assume typical values for T_i and DT: T_3 = 4.0 msec, DT = 0.1 msec, T_2 = 2.0 msec, and N = 8. Eqs. (13) and (14) yield:

$$476 \geq f_2 \qquad 234 \geq f_3$$

From Eq. (12)

$$f_W \geq (476 + 234) \times (40 + 16 \times 8) = 119{,}280 \approx 120{,}000 \text{ bauds}$$

When each axis has a dedicated processor for the servo-controller, Eq. (12) becomes

$$f_W \geq (f_2 + f_3)\,(40 + 16)\,N \qquad (12a)$$

and in this case result is

$$f_W > 318{,}000 \text{ bauds}$$

When there are more computers on the network or more axes of motion that require synchronous operation, the real constraint may be imposed by the network bandwidth and not by the algorithm calculation time. A possible solution would be to reduce f_2 and f_3. However, a lower value of f_2 deteriorates the resolution of the interpolator, and consequently deteriorates the performance of the controller (in terms of part precision and cutting tool velocity).

The bandwidth given for commonly used networks is relatively high, for example 10 Mbps for Ethernet with coaxial cable. However, the difference between networks used for control and those used to transfer information is the real time requirements. In control we require synchronous operation and accurate timing is a given condition. Therefore, the practical bandwidth to guarantee reliable operation is much smaller than the maximum given in the literature.

6. Conclusions

It is becoming commonly accepted that open-architecture controls will allow simple and gradual system building. They will allow the end-user to invest in several smart controllers, and integrate them into a system. Later the user will be able to expand the system gradually. This paper suggests that such thinking might paint too simplified a picture. Timing constraints currently affect performance, which calls for the development of laws relating information flow to system performance (precision, reliability and production rate).

References

[1] Pritschow G., et al., "Open system controllers - a challenge for the future of the machine tool industry," CIRP Annals, vol. 42, no. 1, pp. 449—452, 1993.

[2] Proctor F. M., et al., "Open architecture for machine control. NISTIR-5307," Tech. rep., NIST, MD, 1993.

[3] Next Generation Controller Specification for an Open Systems Architecture Standard, Manufacturing Technology Directorate Wright Laboratory, September 1994.

[4] Proctor F. M., Michaloski J., "Enhanced machine controller architecture overview. NISTIR-5331," Tech. rep., NIST, Gaithersburg, MD 20899, December 1993.

[5] Stewart D. B., Volpe R. A., Khosla P. K., "Design of dynamically reconfigurable real-time software using port-based objects. CMU-RI-TR-93-11," Tech. rep., Carnegie Mellon University, Pittsburgh, PA 15213, July 1993.

[6] V. B. Lortz V. B., Shin K. G., "MDARTS: A multiprocessor database architecture for real-time systems," Tech. rep., The University of Michigan, 1993.

[7] Altintas Y., Munasinghe W. K., "A hierarchical open-architecture CNC system for machine tools," CIRP Annals, vol. 43, no. 1, pp. 349-354, 1994.

[8] Koren Y, Lo C., "Advanced Control for Feed Drives," CIRP Annals, Vol. 41/2, pp. 689-698, 1992

[9] Ulsoy A. G., Koren Y., "Control of machining processes," J. of Dyn. Sys., Meas. and Ctrl., vol. 115/2, pp. 301-308, 1993.

[10] Park J., Pasek Z. J., Shan Y., Koren Y., Shin K. G., Ulsoy A. G., "An Open Architecture Real-Time Controller for Machining Processes," 27th CIRP Intl. Semin. on Manuf. Sys., pp. 27-34, 1995, Ann Arbor, MI