DISTRIBUTED MOTION CONTROL

Jacob Hefer
Elmo Motion Control
Westford, MA

Abstract *– Distributed motion control is a reality; today's processing power, deterministic protocols and network technology make that possible.*

I. CENTRALIZED OR DISTRIBUTED MOTION CONTROL

There are several interpretations for the term “distributed” in the motion control business. The first interpretation is an extension of the traditional centralized motion controller approach, which uses non-intelligent servo amps with basic analog commands. The extended topology still has centralized controllers, which communicates with digital servo drives over networks with defined protocols. In this interpretation, the majority of the motion control tasks (synchronized motion, position, ECAM, position control and speed in several cases) are still performed by the central controller.

The second interpretation is based on intelligent servo drives whose activities are coordinated by a multi-axis supervisor (MAS). The MAS may be a PC, PLC or stand-alone dedicate unit which communicates across standard networks using standard protocols.

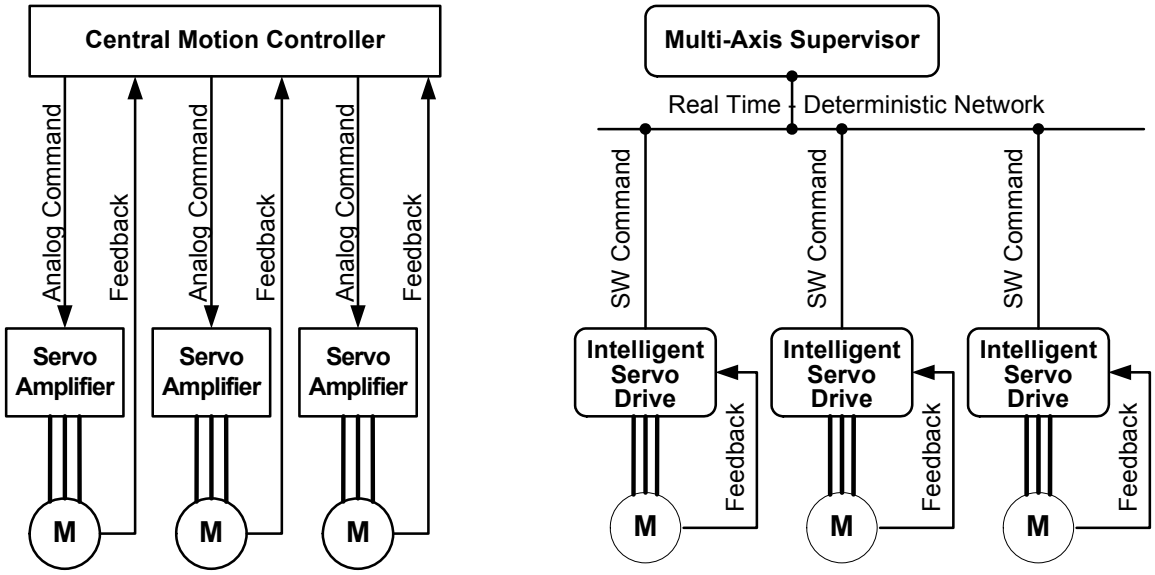

Figure 1 Centralized vs. Distributed Control

II. ERROR CALCULATIONS

The dilemma faced by motion control engineers is how much “Error” is acceptable in their application. That error is a combination of mechanical limits, feedback resolution, motor, drive and motion controller performance. “Networking” and “trajectory interpolation” should also be added to the error equation when calculating the error in a distributed Motion Control configuration.

Figure 2 The Error Chain

This article will clarify the networking and trajectory error added (if any) to application error when working in a distributed motion control environment.

In a distributed motion control environment, trajectories for synchronized motion are sent by the multi-axis controller, via a deterministic network (and protocols), to intelligent servo drives. The trajectories are sent by a method called "data-streaming". Interpolation is performed by the intelligent servo drive.

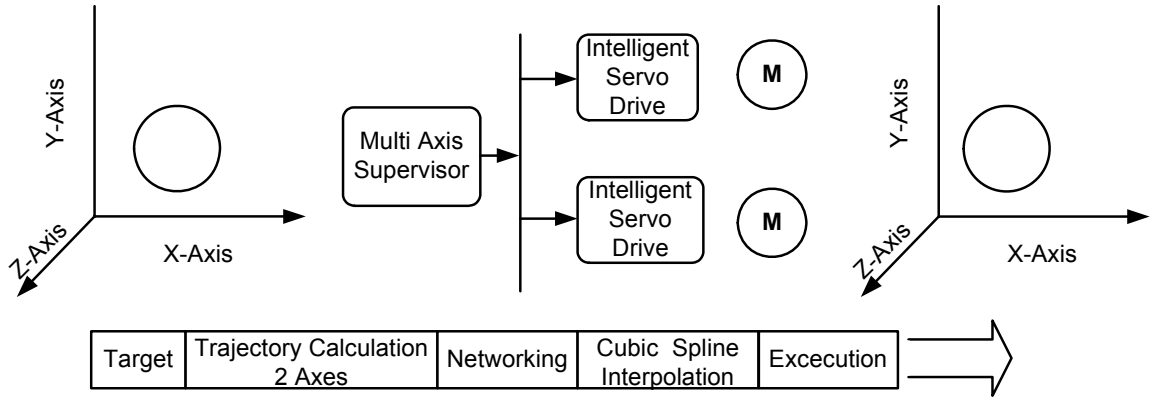

Figure 3 Distributed Processing Model

For example, to create a “circle” trajectory, 2 axes are needed (X and Y). The command that executes a circle will look something like: “circle (2000, 0,360)” where 2000 is the number of counts and 0, 360 are the start and end angles.

The multi axis manager (controller) translates the command to a pair of single axis position commands (P1 , P2 , P3 ….Pn) based on the trigonometric equation of circle $x^2 + y^2 = r^2$ where $x(t) = r \cos \alpha (t)$, $y(t) = r \sin \alpha (t)$ … t represents a synchronized time stamp.

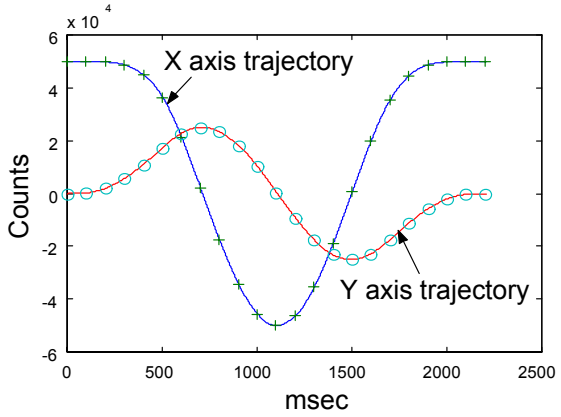

Figure 4 X-Y Trajectories

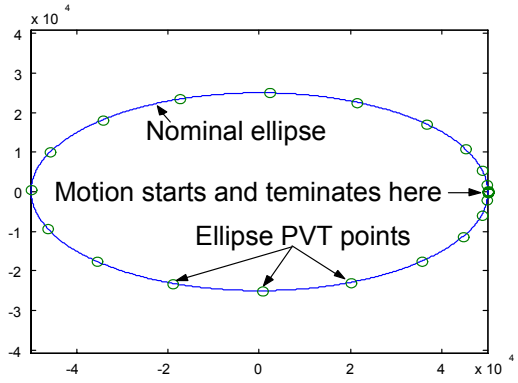

Figure 5 Elliptical Path

The trigonometric information above can be displayed as a PVT table.

X axis			
Point	Position	Velocity	Time
1	Px1	Vx1	T1
2	Px2	Vx2	T2
3	Px3	Vx3	T3
.			
.			
n	Pxn	Vyn	Tn

Y axis			
Point	Position	Velocity	Time
1	Py1	Vy1	T1
2	Py2	Vy2	T2
3	Py3	Vy3	T3
.			
.			
n	Pyn	Vyn	Tn

With deterministic networks and protocols such as CANopen, the information above can be sent to the axes in several ways: “On the Fly” or “Buffered”. In each case the axes execute the trajectory with the same time by interpolating the trajectory with a linear or 3rd order equation.

3rd order interpolation using 4 points of position data to minimize error, looks like this:

$P(t) = a(t-t0)^3 + b(t-t0)^2 + c(t-t0) + d$

$V(t) = 3a(t-t0)^2 + 2\,b(t-t0) + c$

The differences in the interpolation “level” can be dramatic (see the figure below).

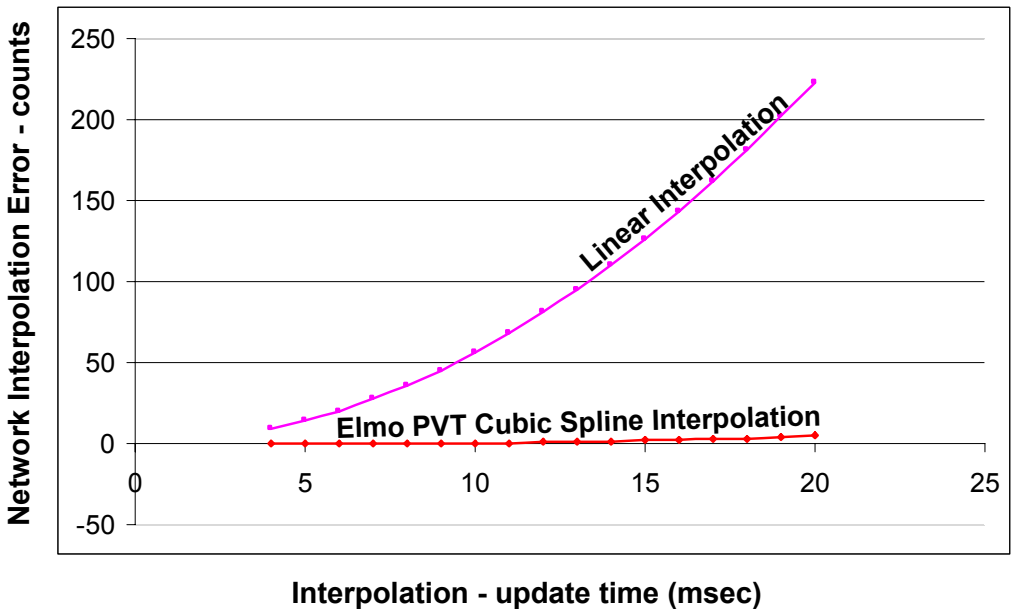

Figure 6 Comparison of Interpolation Errors

The entire distributed motion control equation represents the relationships between:

- Update time (network speed)
- Interpolation algorithm
- Data density (number of points)
- Bus load

Taking all the above into consideration will provide a fair estimate of “performance’ of the distributed system and give a numerical value of the error added by using a network- based configuration.

III. FASTER NETWORK ≠ BETTER PERFORMANCE

A common approach taken when trying to improve performance is to run after a “faster” network(s). Unfortunately, this approach does not necessarily improve performance because "deterministic" behavior is not automatically improved.

In motion control systems, the time it takes to transfer data, and how the data is transferred must be known. So, to improve the performance of the network, deterministic behavior must also be improved.

Faster networks are a good option when they are supported by a "reach" protocol. In a motion control environment there is no time for “acknowledgments” during data transmissions. Therefore, open loop “ send and forget” protocols that can deliver 100% of the data, 100% of the time, on-time, guaranteed is required.

IV. TYPES OF DETERMINISTIC NETWORKS

The CANopen protocol on CAN networks is one approach that offers deterministic behavior. There are several others protocols and network types on the market which provide similar deterministic performance.

The CAN bus, by nature, is not a “fast” network when compared with Ethernet, FireWire and other, more exotic approaches.

V. Distributed Motion Control Example

The following example shows how a Distributed Motion Control Supervisor works with a pair of intelligent servo drive (of the SimplIQ family) in a CANopen environment to create a small circle.

- Velocity 150 mm/sec
- Diameter 10 mm
- Error requirements ± 2.5 um

Designers should ask the following questions:

1. Can the error requirement be achieved?
2. How many axes can be controlled simultaneously?
3. What is the expected bus load?

The following will clarify:

Simplified calculation	1 µm = 1 count
Ordinary Incremental encoder	2,500 PPR ~ 10,000 CPR
Pitch	10mm (10mm/turn)
D	10 mm = 10,000 counts
V	150 mm/sec = 150,000 counts/sec
Error	0.3 µm (0 - 1 counts)

The error calculation is based on the following: (cubic spline interpolation)

- PVT_Err = (1-cos(α /2)-sin² (α /2)/2)*D/2
- Were alpha is the interpolation segment angle of the circle and equal to: alpha=2*V*T/D
  - D: circle diameter
  - V: spatial velocity
  - T: interpolation time

The linear interpolation error can be represented by

Linear_interpolation_Err = (1-cos (alpha/2))*D/2

In answer to the Designer's questions above:

1. When creating a circle with a 10 mm diameter (10,000 counts), a 56 count error can be expected.
2. The number of axes depends upon the network baud rate, sync time and other duties. In an isolated 800 Kbps network with 10 msec update time and 10 msec sync time a 26 synchronized axes can be supported.
3. When 26 axes are supported on an 800 Kbps network with a refresh time of 10 msec, the bus load will be 70% (see the middle graph in the figure below).

D	T	PVT	Linear
10000	20	5	223
10000	19	4.1	202
10000	18	3.3	181
10000	17	2.6	162
10000	16	2.1	143
10000	15	1.6	126
10000	14	1.2	110
10000	13	0.9	95
10000	12	0.7	81
10000	11	0.5	68
10000	**10**	**0.3**	**56**
10000	9	0.2	45
10000	8	0.1	36
10000	7	0.1	28
10000	6	0	20
10000	5	0	14
10000	4	0	9

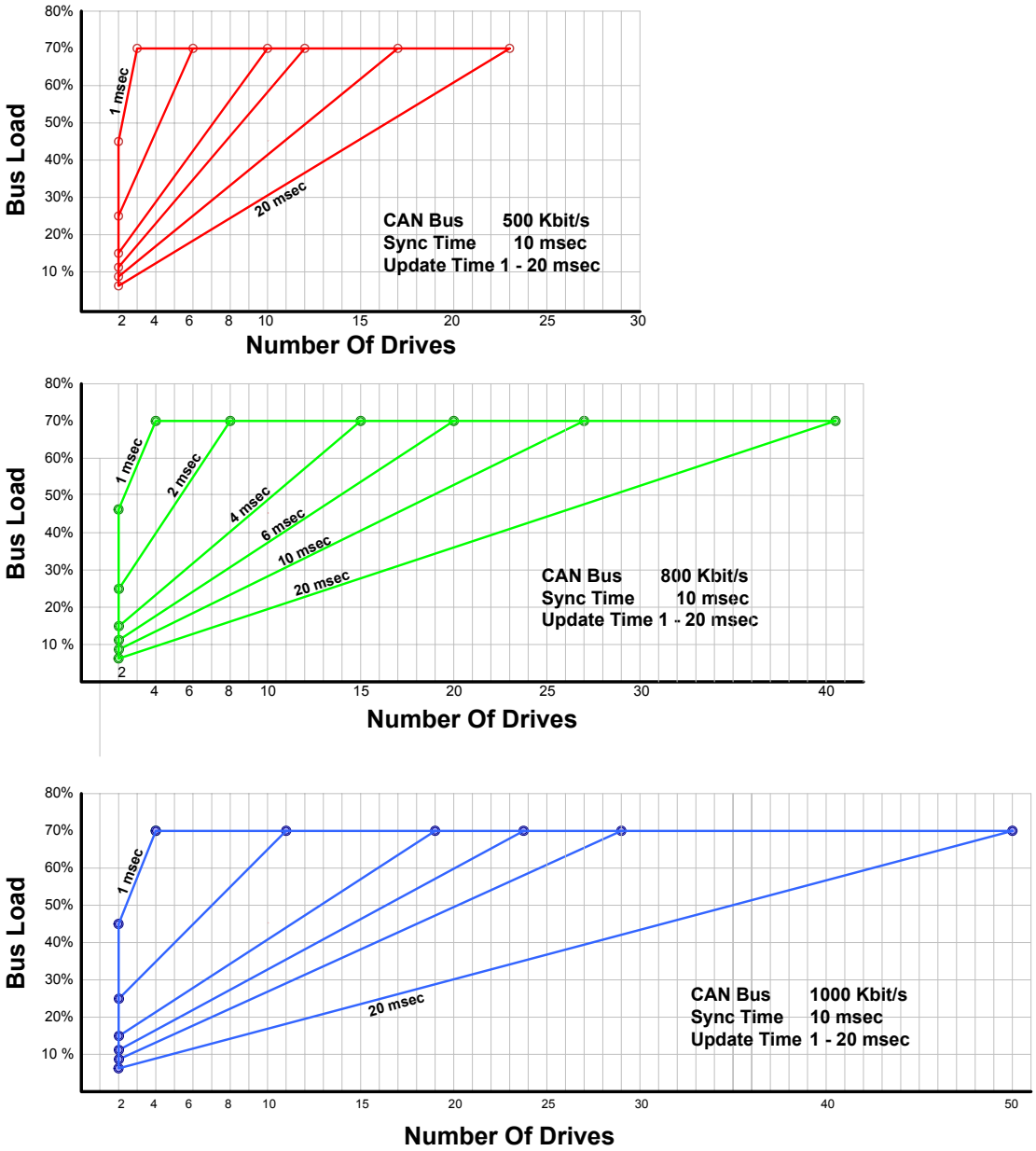

Figure 7 Comparison of Bus Load at Different Bus Speeds.

VI. Conclusions

The example above demonstrates that demanding mechanical requirements, however exotic, can be resolved with a deterministic network and protocol … but not necessarily by a “high speed” infrastructure. Excellent performance can be achieved with intelligent servo drives. The market will eventually provide “fast” networking technology based on Ethernet. But for now, most motion control applications can be created with intelligent servo drives working with Multi-Axis supervisors on standard networks.