

Analysis and Modeling of Servo Motor Controls in Military Vehicle Control Systems Update

Christopher B. Mushenski

PM-MGV Common Systems, U. S. Army, TACOM-Warren

ABSTRACT

This paper covers further analysis of various concepts that are used to implement Servo Motor Control in military vehicular control systems. A survey of existing “hard real-time” servo controls and applicable design patterns is presented. These design patterns along with their critical parameters are identified and described. Potential solutions are created from combinations of design patterns and parameter choices. The solutions are modeled with plausible parameters to identify parameter sensitivities. As the modeling progresses, the solutions’ frequency responses, sensitivities and functional performance are also evaluated. This update paper concludes with a summary of architectural guidelines.

INTRODUCTION

There is an increasing trend in recent years to use decentralized “hard real-time” control implementation in military and commercial vehicles. The driving forces for this include the following:

- Advances and cost reductions in network technology
- Demands for commonality
- Demands for higher reliability
- Improved limp home capabilities

The advances and cost reduction in network technology are real. The networks are now faster and more reliable than ever. Examples of this are given in the “Rotorcraft” presentation by Dale Johnson, reference [11], which shows the past and predicted evolution of network technology. The advances are continuous. Ethernet for example was 100 Mega-bit per second in 2000, reached 10 Giga-bit per second in 2005 and is expect to reach 100 Giga-bit per second in 2010. The possibilities of using remote input / outputs and remote actuators over the networks are now feasible.

The implementation however is not always straight forward and has a number of pitfalls especially with

regard to latency and jitter. This paper addresses a few of the problems and some of the associated solutions.

Many solutions turn out to be concepts that have been under development for years in universities and research facilities. In other cases, there are new areas to be researched and documented.

COMMON BUILDING BLOCKS

The demand for commonality comes from architectural desires to use common “building blocks”. It is thought that common units, when properly designed, can be used across the vehicle product line. While the use of “product line” development is used in commercial application, the commonality approach is strongly desired by the US Army. The reason for the emphasis on commonality is the desire to be able to take common parts from one vehicle and use them in a totally different control in another vehicle.

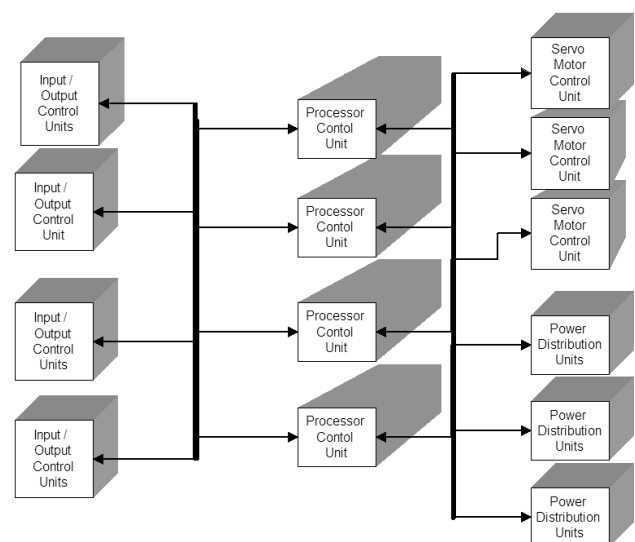


Figure 1 – Decentralized Real-Time Control

The most common division of systems is into the basic building blocks of input / output units, computing units, power control units and servo controllers. This vision of simplicity in the future is illustrated in Figure 1 – Decentralized Real-Time Control. Multiples of each building block type can be used to facilitate redundancy, “limp home” capabilities, expansion, and reconfiguration. The redundancy and the reconfiguration facilities provide the increased reliability needed for the survivability of the soldiers in the field.

EXISTING TECHNIQUES

The following sections describe a survey of the real-time control techniques and explanations of each technique. The optimal solution for a specific problem will come from using a set of these techniques to accomplish the goals of the system.

SURVEY OF EXISTING INDUSTRIAL TECHNOLOGY

The goal of this survey was to establish what techniques were used in existing control system technology. The area of industrial controls and robotics were of special interest because of the similarities to the military vehicle applications. The survey was generated primarily from the searching Internet documents and university libraries. The documents that were the most helpful were Doctoral Theses (such as reference [4] and [9]) and technical studies from industrial control groups such as IAONA (Industrial Automation Open Networking Alliance) [10].

We gathered considerable information from the European industrial control standards groups who were using some techniques that appeared to be more advanced than what was being considered in our initial efforts. This is especially true in the use of Ethernet Data Packet Packing to create high speed low latency data links.

Of interest was that the transmission of formatted data in the standard message data format was utilized in only three instances, Ethernet Powerlink, Power EtherNet/IP and Modbus/TCP. In three cases Ethernet Frames were constructed by multiple controllers in a non-standard manner. The results of using the “multiple computer message packet construction” was fast data update rates. The fast data update rates came with at the expense of non-standard hardware, which is required to support the multiple computer construction of message packets.

One of the other techniques that were widely used was the use “Time Division Multiple Access” (TDMA). This came in the form of either “isochronous message packets” or “time triggered protocol” (TTP). Table 1 – Applicable Real-Time Control Techniques for Control over Distributed Computers shows the list of applicable techniques that were determined for common use, their

benefits and drawbacks. To apply these techniques, a topology has to be implemented.

Goals of Servo Network Implementation

Our goal is to apply these techniques to develop servo system implementations that would meet the needs of lethality, propulsion and survivability systems. To implement this properly requires the following characteristics:

- Determinism
- Bounded and Minimized Latency *
- Bounded and Minimized Jitter *
- “Highly Dependable” communications
- Support for timed message transmissions

* indicates that these are covered in depth in this paper.

Highly desirable capabilities include the following:

- Guaranteed delivery of data packets
- Composability
- Minimum Weight
- Minimum Volume
- Reconfigurable
- High Reliability
- Guaranteed End-to-End Latency
- Local clock synchronization to global clock
- Support for time triggered tasks

Composability is the ability to add additional control units to a data network without affecting the existing units. One test for this is to add a “babbling idiot” control unit, e.g. a continuously sending transmitter, to the network and verify that it has not affected other control units’ operations. The truth is that composability is difficult to achieve without the use of Time Division Multiple Access e.g. Time Triggered Protocol or Isochronous communications. These will be discussed later in the paper.

Simple Models of the Systems

The basic control system is modeled in Figure 2 - Simple Control Loop and shows a simple model of a closed loop control. In our study we utilize the distributed computer system, which requires a remote input-unit, high level computer processor and a servo actuator controller, a servo actuator, an actuator and a feedback sensor. This is shown in Figure 3. We will use a sensor input device, a high level controller, a servo motor controller and a 3 phase brushless DC motor with an attached resolver as our test case. The plant in this example shall be a simple inertial and friction load. The system will be modeled with ANSOFT Simplorer.

Table 1 - Applicable Real-Time Control Techniques for Control over Distributed Computers

<i>Technique</i>	<i>Pros</i>	<i>Cons</i>	<i>Common Existing Technology</i>
Highest Priority Processing	Functional for one high priority application	Does not work well on multiple high priorities	Yes
Time Triggered Protocol	Has composability, simplifies formal verification, and reduces latency when compared to Event Triggered protocol	Requires complete knowledge of resource allocations. Requires special timer hardware and synchronization	Yes
Timed Task Activation	Minimizes latency	Requires task activation timers	Yes
Dedicated Dual Controller Hardware	Uses common LRUs	Has a minor size impact.	Yes
Single Controller Hardware	Minimum Latency and maximum control	Least amount of fail safe and reconfiguration	Yes
Latency Compensation	Significantly reduces effect of latency.	In practice will still leave a certain level of noise.	Yes
Data Packet Packing	Synchronizes by start of message packet.	Limits the number of devices on the bus	Yes
Data Switches	Pseudo Deterministic, large number of units on bus.	Cost, weight and volume plus additional hardware.	Yes
Global Time Synchronization	Facilitates time stamping of data and events.	Requires a limited use of the network periodically to update the remotes to compensate for remote clock drifts.	Yes
Non-periodic control law	Eliminates jitter.	Not well supported by current control simulation tools but has sound mathematical basis.	No
Segmented topology	Limits the number of nodes on the bus to minimize interaction	Limits some of the opportunities for reconfiguration	Yes
Star topology	Has extensive re-configurability and back up	Causes issues with composability and weight	Yes

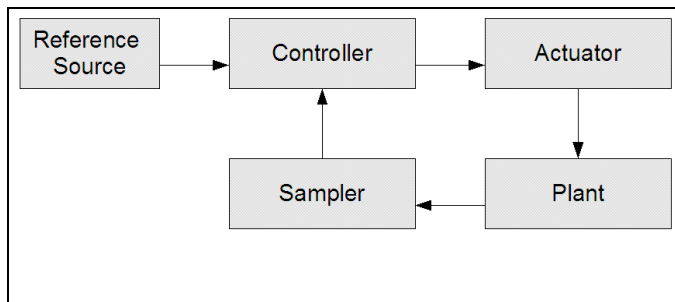


Figure 2 - Simple Control Loop

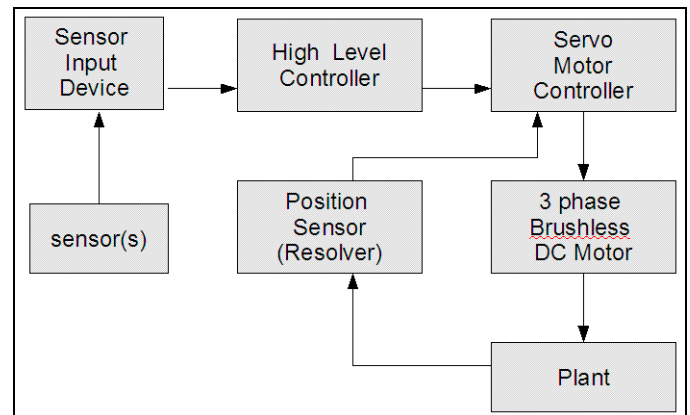


Figure 3 - Servo Control System Evaluation Model

SOURCES OF LATENCY

Latency is the delay between when something is changed and when that change is detected. In communications, it can be the time the message packet is on the bus, but that is a contributor and not the total value. When looking at a control system, the latency for a signal starts right after it is read and does not stop until its input/output value affects the value/state of the item it is trying to control. This includes all the processing for control and input/output. The latency contributors are described in the following subsections.

The delays are modeled as they are observed. The processing delays will be modeled as pure propagation delays.

Sensor Input Processing

If we assume an event driven system then, the sensor input processing delays included are defined in the following table.

Table 2 - Sensor Input Latencies

Identified Effect	Reasonable Value
Sensor Input processing	48 us
Scaling and packet building	16 us
Packet transmission Delay	64 us

The summary of these effects would be modeled as 128 microseconds of pure delay. The packet sending time is normally a function of the loop time in event triggered systems. Since input processing is normally performed at the start of loop execution cycle and the transmission of serial message is an output function it is normally done at the end of the main loop. The maximum delay loop time is 500 microseconds and the minimum is the 128 microseconds. Our model shall use a message delay of 64 as the model in this event triggered implementation. The total worst case delay is 564 microseconds.

Note that one of the common mistakes by novice working on latency studies is to not include the normal software processing time. Since embedded software commonly uses the input / control / output architecture, it will most likely be used unless specified otherwise. To not include its timing is a serious mistake.

Having read the sequencing in the above paragraphs, one realizes various ways to improve it. If we time the tasks so that the data is read and processed just before

the message is sent, it can substantially reduce the latency of the data. This is illustrated in Figure 4 and is referred to in the paper as “time tasked activation.”

By adding a timer for sensor processor interrupts, we can reduce the latency between reading the data and sending the data. A reasonable estimate for the interrupt processing is dependent on the other activities in the sensor processor. A new latency value of 128 microseconds could represent the remote sensor delay and add it to the message packet propagation delay to get a total of 192 microseconds of delay.

Time tasked activation does have a cost. It requires “time-triggered procedure-calling” facilities. Fortunately these are standard on real-time controllers under the names like “event processor array” and “timer processing units”. The timed tasked activation is illustrated in Figure 5 – Time Triggered Task Activation. There are two timed interrupts used, one on the remote unit to start the reading of raw data and another one to start the sending of the data.

Minimum latency exists when timed tasks schedules the reading of raw data and the transmission of message packet with the data. Note that that time for higher priority interrupts to be serviced must be for worst case conditions. This requires the timing of low level tasks and requires the use of multiple high resolution timers for task and data transmission timing. It also requires the use of Time Trigger Protocols.

Notice that this technique only works if the time is synchronized to a high resolution. In general, this resolution is less than one microsecond. Ideally is it is less than ¼ microsecond.

Global Clocks in Distributed Control Systems

In distributed computer control systems, there is a requirement to synchronize a master clock with remote clocks located on the distributed units. The reason for this is illustrated in Figure 6 – Need for Copies of Global Clock where in the left hand configurations the unified control is established with all control sections using the same synchronized clock time. When the remote I/O devices develop a local independent clock, the synchronization fails and temporal errors begin.

There is also a loss of correlating data records because timing is now local based and with out reference to the global time used by the processing section.

There are various standards used to do synchronization. The Ethernet de facto default standard, from evaluation of the different protocols, was the global clock synchronization standard, IEC 61855. This standard defines a method to match remote clocks to the global clock with less than 1 microsecond of inaccuracy, which meets the requirements of time triggered networks and

time stamping of data. The accurate time stamping of data facilitates “latency compensation” on time critical data, which can then be used to improve the overall accuracy of control system reference data (e.g. position, speed or torque command data).

Timers for task activation

Figure 4 - Recommended Time Triggered Task Hardware Concept provides a recommended task timer pattern for real time applications implementation. The functionality shown here is based on the pattern used by Intel Corporation in their 80196 series processor’s event timer.

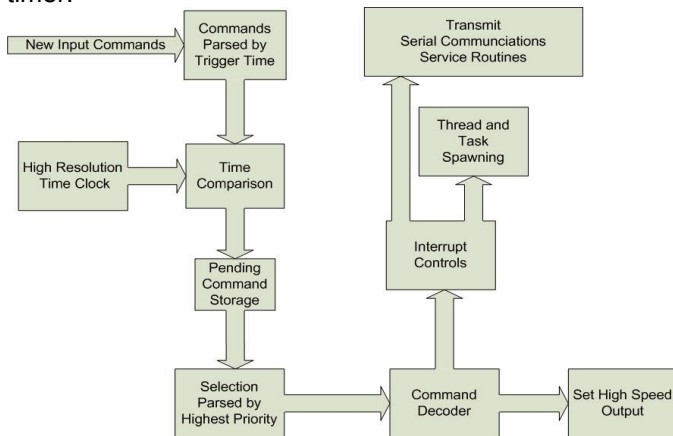


Figure 4 - Recommended Time Triggered Task Hardware Concept

New commands are entered parsed by scheduling time and stored. The commands are compared by their trigger time with the high resolution time clock value. When the times are equal to or greater than the trigger time the commands are moved to pending command storage. The highest priority command is issued. The

command either sets interrupts which are responded to by specific services or sets a high speed output or both.

Other alternatives to this approach exist. The pertinent elements are as follows:

- Trigger commands by time.
- All the highest priorities are to be operated first.
- Interrupt low priority task where required.

Only the timer, timer comparison and temporary storage need to be handled in hardware to support this element. In the case process build for hard real-time controls, this functionality is already available on chip.

High Level Controller

The purpose of the high level controller is to take the sensor inputs, process them and provide a reference to the servo motor controller. An example would be to take the handle information and inertia signals and speed of the traction drive motors.

The high level controller receives data from the joystick controller at rates of 500 microseconds. The high level controller can be run with 500 microsecond main loops also. There is a significant problem in that there is a lack of synchronization between the sensor input device and the high level controller. If one looks at the sensor input device and uses it as the time reference, the sensor input device sends the message packet but since the main loops are not synchronize the packet will not be received until the input processing of the high level controller decides to read the input. This delay is equal to the main loop time of the high level controller. Figure 7 - Effect of Lack of Distributed Controller Temporal Synchronization illustrates the worst case delay due to this lack of synchronization.