Developing Software for a Distributed, Synchronous, Real-Time System

Mr. E. Augustus Cooper, Jr., and Mr. A. Lee Philpott, Jr.

The submarine launched ballistic missile (SLBM) system has been a key element in U.S. national defense for nearly 40 years, and the Naval Surface Warfare Center, Dahlgren Division (NSWCDD) has been a major contributor to its success from the beginning of the program. The SLBM Program, like all defense programs, is evolving in an environment that includes budget constraints and changes in world situation and system design. To meet the demands of the 21st century, the SLBM Fire-Control Life-Cycle Cost Control (LCCC) Program is changing the fire-control system (FCS) from a centralized, point-to-point system of SLBM-developed hardware and software to a distributed, networked system containing a mixture of nondevelopmental items (NDI), commercial off-the-shelf (COTS), and SLBM-developed hardware and software. This article describes the architecture of the SLBM fire-control, real-time software and its evolution to a distributed system.

Introduction

The SLBM system has been an important cog in U.S. national defense for nearly 40 years, and NSWCDD has been a major contributor to its success from the beginning of the program.[1] NSWCDD has experienced the evolution of the SLBM system from its initial capabilities to its current features and is part of a program that will prepare the SLBM system for the 21st century. The SLBM Fire-Control LCCC Program is changing the FCS from a centralized, point-to-point system of SLBM-developed hardware and software to a distributed, networked system containing a mixture of NDI, COTS, and SLBM-developed hardware and software.

This article describes the architecture of the SLBM fire-control prelaunch software and its evolution from a centralized system to a distributed system. The fire-control prelaunch software satisfies the same real-time requirements in both the centralized and distributed systems; yet communication in the distributed system is designed over a commercial operating system (OS) using standard internet protocols. This article shows how the incorporation of a commercial OS supporting network access can be used to solve a distributed, real-time problem.

The FCS, as described in this article, has been simplified in places for brevity and simplicity. For example, redundant equipment and multiple missile communications are omitted.

PROBLEM DEFINITION

The SLBM shipboard subsystems prepare and launch missiles that contain a guidance system, a flight-control system, and multiple independently targeted reentry bodies. Once the missile is launched, the guidance and flight-control systems steer the missile so that it releases the reentry bodies at the position and velocity required to fly on a ballistic trajectory to the desired impact points. The FCS participates in the preparation of the missiles before launch. It performs three major tasks: system maintenance, targeting, and launch sequence-related computations and control. System maintenance is ongoing, non-real-time work prior to the launch sequence. It is intended to ensure that the system is operational. Targeting tasks, which are also non-real-time, include the updating of targeting information in the mission data base by FCS operators and the assignment of targets to missiles. The prelaunch FCS software, under operator control,

- ✦ Computes missile targeting and steering data.
- ✦ Prepares the missile inertial guidance system for flight through a process that includes the continuous exchange of information among the guidance system, the FCS, and other weapon-system components.
- ✦ Coordinates the events associated with launching the missile.

The FCS is being modernized to prepare it to meet future requirements and to reduce life-cycle costs by replacing expensive obsolete components. The prelaunch software, however, must satisfy the same communication requirements in the distributed system as it does in the centralized one. Figure 1 depicts the major components that participate in prelaunch communication. The periodic communication between these components is synchronous since they all receive 1-pulse-per-second (1PPS) signals that allow the coordination of the exchange of information at timing marks within the 1PPS interval. The guidance system, for example, can require that the FCS transmit a specific data list at a defined mark after the 1PPS signal. The 8PPS (one pulse per 0.125 seconds) and the 2PPS (one pulse every half second) periods exist synchronously with the 1PPS for the purpose of computing prelaunch communication (see Figure 2). Within each period, the FCS software establishes timing marks as reference points for performing computations and communication.

Several examples of communications performed by the FCS are developed in order to allow comparison of the centralized and distributed architectures. These examples are communication with the Operators Control Panel (OCP), the submarine navigation system, and the missile guidance system.

Example 1: OCP Communication

The OCP reads operator selections, provides the status of various systems, and maintains important system-state information. It receives and displays this information using a collection of color-coded actuator/indicators (*buttons*) and lights. The OCP receives information from the operator, FCS applications, and other weapon-system elements. All operator input and output are predefined. This allows the input and output lists to be of a fixed length, and sent and received synchronously on a specified timing mark. System-state information is a set of variables *(state terms)*, some of which are examined by applications to determine the current state *(state input terms)*, while some are assigned values by applications to indicate their state *(state output terms)*. OCP communication requires reading input associated with operator selections, determining system state by executing logical equations, and updating operator display status using the output of these equations.

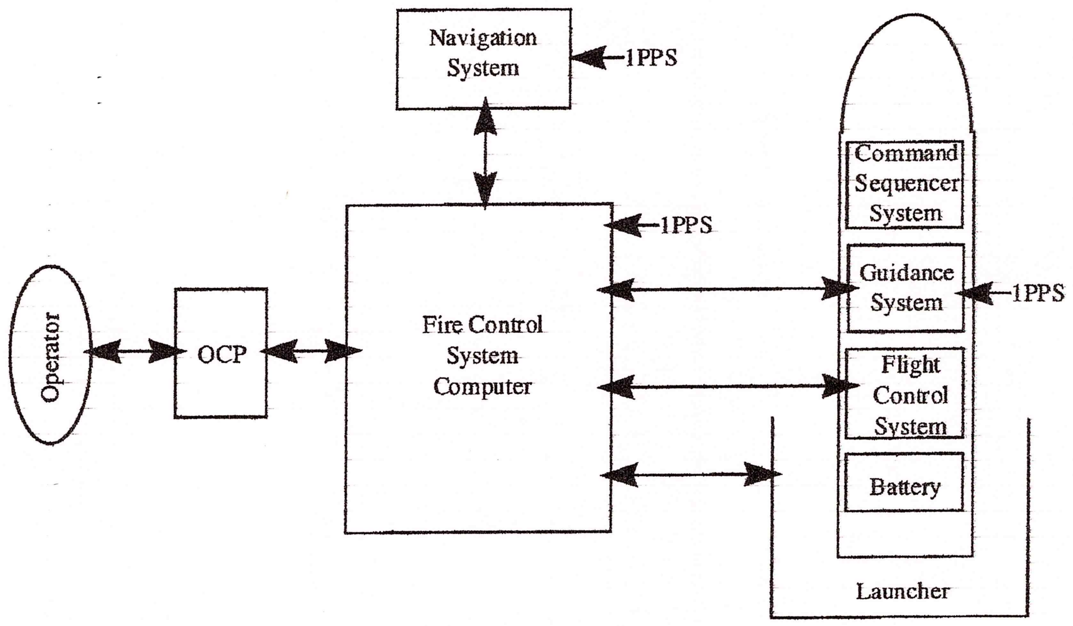

Figure 1—Major Components of a Prelaunch Sequence

Example 2: Navigation Communication

Navigation communication requires the reading of information from the ship's navigation system (e.g., position and velocity) and conversion to FCS formats. This is done at a specified interval.

Example 3: Guidance Communication

// There are many types of communication between fire control and guidance. This article specifically considers two: communication of position data to guidance and status communication from guidance. These are performed at a specified interval and require computing guidance position data based on navigation inputs, writing these data to guidance, reading status data from guidance, and computing guidance status as required for the OCP logical equations.

CENTRALIZED AND DISTRIBUTED FCS

In a centralized system, the FCS computer is connected to the other systems by point-to-point connections, as shown in Figure 3. All of the launch-sequence-related, targeting, and system-maintenance software executes on the FCS computer. The current computer, which is of 1970s vintage, was designed specifically for the FCS and includes one megabyte of main memory; 32-bit, fixed-length instructions; 16-bit virtual addresses, specialized real-number support for computations, several Programmable Interrupt Generators (PIG), and a programmable Direct Memory Access (DMA) input/output controller for interfacing the FCS to other systems.

In a distributed system, the FCS software executes in multiple computational nodes connected by a network (see Figure 4). The FCS

+ FC + ~~FCon~~
Flying Control System

in this figure includes the OCP for controlling the launch sequence; the Display and Control Subsystem (DCSS), the Enhanced Guidance Interface Subsystem (EGISS), the Navigation/ Missile Interface Subsystem (NMISS), and the Data Entry Subsystem (DESS) for controlling targeting and system maintenance software; and the Fiber-Distributed Data Interface (FDDI) network. What was previously computed in one computer is now computed in a networked environment. The primary computing nodes for the three examples are the DCSS, the EGISS, and the NMISS. Each node is Versa Module Eurocard (VME)-based and has components as shown in Figure 5. The fire-control software executes in the commercial PowerPC module; the commercial FDDI module provides access to the network; and the specialized Input/ Output (I/O) module provides access to SLBM components, such as the guidance and navigation systems, and the OCP.

Both the centralized and distributed FCS establish the timing marks discussed previously; however, each does it differently. The reason for the difference can be discovered by examining the communication of each FCS architecture.

Communication in the centralized FCS is performed by software executing in the one computer using its DMA controller. The DMA is programmable in a simplistic, but powerful way. Each port can have a "chain" of operations ongoing, where each operation in the chain is an output of data or commands, an input of data or commands, or a skip of the next operation. Beginning the chain requires a single Central Processing Unit (CPU) instruction to start I/O with the address of the chain. Starting I/O requires so little overhead from the CPU that it can be performed at the interrupt level. Thus, the OS of the centralized FCS establishes a system-wide set of timing marks, each with a corresponding connect point for an interrupt handler. This allows applications to connect interrupt handlers to the timing marks. The timing marks are established by using a PIG in conjunction with the 1PPS signal. When a 1PPS interrupt is received, a PIG is set up to generate an interrupt—for example, 65 milliseconds later and every 125 milliseconds thereafter. This sequence would result in interrupts at the 65-, 190-, 315-, 440-, 565-, 690-, 815-, and 940-millisecond marks. At each of these points, the OS would call the connected interrupt handler to consider sending position data to the guidance system. The prelaunch software builds the chain prior to the timing mark, and the interrupt handler starts the I/O at the proper time.

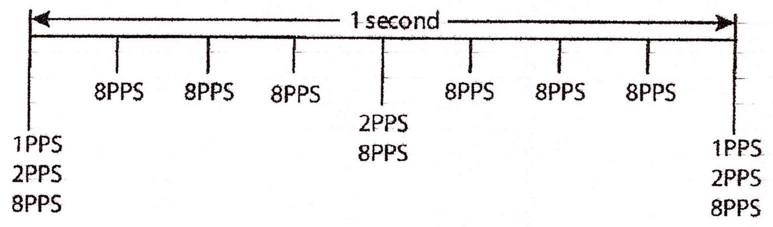

Figure 2—Timing Marks

Communication in the distributed FCS is performed by software executing in the various nodes on the network. This communication is a

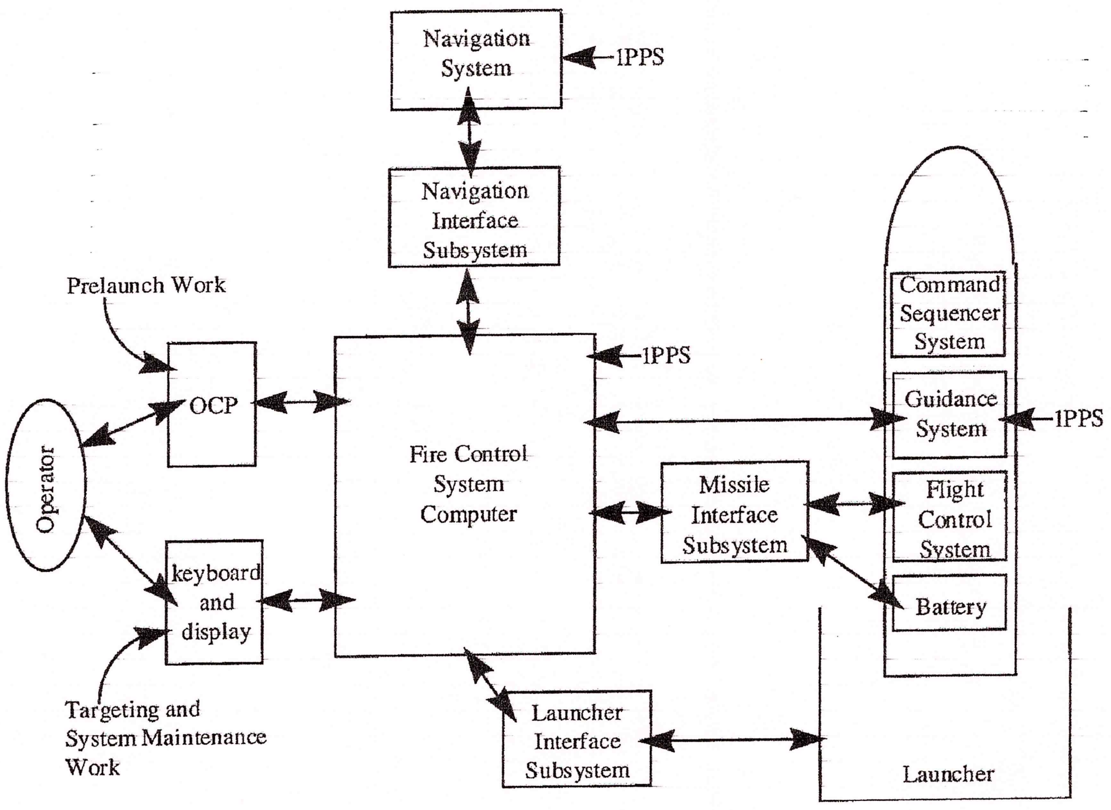
Navigation System
1PPS
Navigation Interface Subsystem
Prelaunch Work
OCP
Operator
keyboard and display
Targeting and System Maintenance Work
Fire Control System Computer
1PPS
Missile Interface Subsystem
Launcher Interface Subsystem
Command Sequencer System
Guidance System
1PPS
Flight Control System
Battery
Launcher

Figure 3—Centralized SLBM System

combination of network I/O and I/O using specialized ports. The commercial OS has changed the nature of I/O. The Berkeley Networking functions,[2] as well as the manner in which the specialized I/O device drivers are added to the commercial OS, require tasks (or processes) to initiate the I/O. It cannot be initiated at the interrupt level. Thus, the OS of the distributed FCS must establish a mechanism for providing timing marks to tasks. Additionally, it is not known which set of timing marks each node requires, and as a consequence, a more general mechanism is established for defining the timing marks.

For example, when a 1PPS interrupt is received in a node, a PIG is set up to generate an interrupt every 125 milliseconds thereafter (that is, at the 0-, 125-, 250-, 375-, 500-, 625-, 750-, and 875-millisecond marks). These interrupts are handled by the OS and are not passed on to application interrupt handlers. Instead, the OS allows tasks to be awakened at any millisecond marks within a 125-millisecond interval. The OS in a node maintains a data structure that describes all timing mark events established within that node. When the 125-millisecond interrupt occurs, the OS consults the data structure to determine whether any task must be awakened in the interval and, if so, when. For example, assume two tasks are to be awakened in a given interval—the first at 40 milliseconds after the interrupt and the second at 82 milliseconds. The OS establishes a second PIG to count 40 milliseconds and generate an interrupt. After the interrupt is handled by the

OS, it establishes the same second PIG to count 42 seconds, and then performs the processing for the second task.

COMPARISON OF COMMUNICATION EXAMPLES

Four examples of FCS communications were selected for specific discussion. They serve to further highlight the differences between the centralized and distributed FCS architectures. The OCP and navigation examples address the provision of system services; the guidance example considers continuous communication with a system outside of fire control.

Example 1: OCP Communication

In both the centralized and distributed FCS, the OS provides OCP communication and system-state computation as a service. In the centralized FCS, OCP communication provides one of the system timing marks. For example, the 1PPS and a PIG are used, as previously described, to establish a timing mark. The OS uses the DMA to write and read values to and from the OCP. The initial DMA program outputs a default set of values for the OCP and reads the results from the OCP. This read, which should be complete by the timing mark, is followed by the execution (by the OS) of the system-state logical equations using the input data to generate OCP output data and the

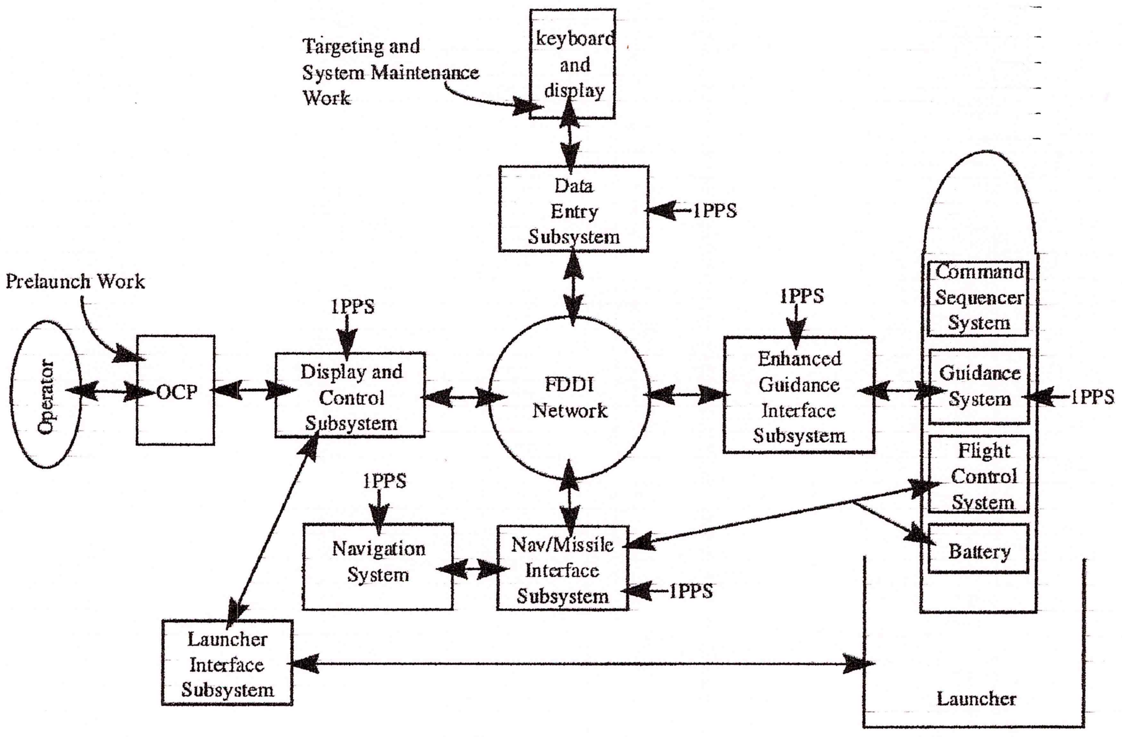

Figure 4—Distributed SLBM System

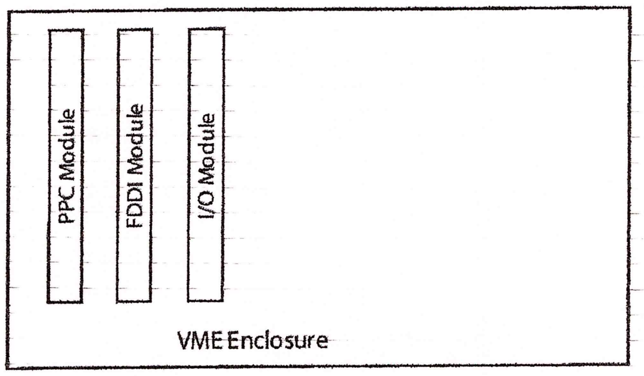

Figure 5—Fire Control Computing Node

restarting of the DMA program. This process is repeated at the remaining millisecond marks.

Since OCP communication involves the computation of system-state information, the OS provides two interrupt-level connect points that allow applications immediate access to the system-state information and precise timing marks. At the first connect point, the OS calls an application interrupt handler just prior to the computation of the system-state logical equations. The OS calls an application interrupt handler, at the second connect point, just after the system-state logical equation computation. The system-state information is contained in memory that is accessible to applications.

In the distributed FCS, the OCP communication provides one of the system timing marks, but it does it in a different manner. OCP communication and system-state computation are done in the DCSS node; applications execute in other nodes. Targeting applications, for example, execute in a DESS node, and the launch-sequence computations and control execute in the EGISS nodes. These applications require access to fire-control information. For example, the prelaunch software needs to know when a launch has been commanded and requires the ability to change system status information when elements (such as guidance) are bad or not available.

Fire-control information is computed in the DCSS node and broadcast on the FDDI network to all other nodes. Each node computes its own system status information and sends it on the FDDI network to the DCSS node. Thus, the OCP communication timing marks provided in the distributed FCS are those used when the fire-control information is received in a node. The OS in a given node uses its own timing mark mechanisms to establish the mark at which broadcast fire-control information is required in the node. An OS task is awakened at that point to perform a nonblocking network read to receive the broadcast fire-control information. The OS task then calls an application connect handler in the context of the task that performed the read, not as an interrupt.

Example 2: Navigation Communication

In both the centralized and distributed FCS, the OS provides the navigation communication and data computation as a service. They are provided in a manner similar to that of OCP communication and fire-control computation.

Navigation communication provides one of the system timing marks in the centralized system. The 1PPS interrupt and a PIG are used to establish two timing marks (different from those provided by OCP communication). The OS interrupt handler, operating at these times, reads navigation data, converts it to fire-control format, and provides a connect point that allows an application interrupt handler to be called. The navigation data are contained in memory that is accessible to applications. The prelaunch sequence application uses this connect point to be notified when navigation data are available.

In the distributed FCS, navigation communication again provides one of the system timing marks. Navigation communication and data computation are performed in the NMISS node. The NMISS software broadcasts navigation data on the FDDI network to make it available to other nodes. Analogously to the OCP case, the timing marks provided by the distributed FCS are those used when the data are received in a node. The OS in a particular node uses it own timing mark mechanisms to establish the mark at which broadcast navigation data are needed within the node. A nonblocking network-read OS task is awakened at the appropriate time.

Example 3: Guidance Communication

This example shows how the prelaunch application software uses the timing marks provided by the OS to perform the required communication with the guidance system. The discussion will consider the communication of position data to guidance and the receipt of status data from guidance. These are only two of many guidance communications; however, all are handled similarly.

Position data are continuously provided to guidance throughout the prelaunch sequence. The data generated in the navigation system are read into the FCS, converted into guidance formats, and written to the guidance system. Guidance requires that this be performed in an interval following the 2PPS mark established in navigation communications. In the centralized FCS, the prelaunch software and the OS software, which reads from the navigation system, are in the same computer. In the distributed system, the prelaunch software is in the EGISS node, and navigation data are read by the OS in the NMISS node. In spite of this, the prelaunch software has the same design in both the centralized and distributed FCS. The difference is the level at which the guidance position processing connect point is called. In the centralized system, it is called at the interrupt level; in the distributed FCS, it is called at the task level.

Collecting guidance status also occurs continuously throughout the prelaunch sequence. The prelaunch software uses the OCP connect point in the EGISS to perform all guidance communication except for position data. Guidance status communication occurs at the OCP connect point prior to the end of the 2PPS interval.

Summary

The SLBM FCS prelaunch real-time software provides position data to missile guidance systems, computes missile mission parameters and provides them to the guidance systems, and coordinates the final events in the launch sequence. The prelaunch real-time software executes in a synchronous system that uses a 1PPS signal to allow subsystems to exchange information on coordinated timing marks.

The prelaunch software was originally designed to execute in an FCS with a centralized architecture. The OS in the centralized system provides two synchronous timing marks using interrupt-level connect points. One of these marks is established by the availability of navigation data, the other by the availability of OCP and fire-control information. The prelaunch software uses these timing marks in its guidance communications design. A key element of this design is that guidance communication is performed by DMA access, thus allowing the prelaunch interrupt connect points to start I/O and check it for completion at a subsequent connect point.

As the prelaunch software is moved to a distributed FCS, similar design concepts are being developed in the context of a commercial OS with network access. The prelaunch software is not distributed across the nodes of the FCS; however, the supporting infrastructure is. The same two timing marks available in the centralized FCS are present in the node where the prelaunch software executes. The navigation data and fire-control information are available in that node when they are received by broadcast transmission on the FDDI network from the NMISS and DCSS, respectively. The timing-mark connect points are provided within the context of the OS tasks that receive the broadcast information.

The shift from interrupt-level timing marks to task-level timing marks is necessary because the commercial OS requires network access through task context. Guidance communication must also be performed at the task level. In order to support task-level timing marks that may be different in the various FCS nodes, a generalized timing mark package has been developed.

REFERENCES

1. Gates, Robert V., "Strategic Systems Fire Control," *Naval Surface Warfare Center, Dahlgren Division Technical Digest*, 1995.

2. Stevens, W. Richard, *UNIX Network Programming*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.