A LIGHTWEIGHT SOFTWARE COMMUNICATIONS ARCHITECTURE (SCA) LAUNCHER IMPLEMENTATION FOR EMBEDDED RADIOS

David Murotake (SCA Technica, Inc., Nashua, NH USA, dmurotak@scatechnica.com)
Alden Fuchs (SCA Technica, Inc., Nashua NH USA, aldenfuchs@comcast.net)
Antonio Martin (SCA Technica, Inc., Nashua NH USA, friendsglobal@hotmail.com)
Bruce Fette (General Dynamics Decision Systems, Scottsdale AZ USA, Bruce.Fette@gd-decisionsystems.com)
Jeffrey Reed, Virginia Tech, Blacksburg VA USA, reedjh@vt.edu}
Max Robert, Virginia Tech, Blacksburg VA USA, probert@vt.edu)

ABSTRACT

The United States Department of Defense has specified a CORBA based Software Communications Architecture (SCA) as part of a procurement specification for the family of Joint Tactical Radio System (JTRS) Cluster radios. The JTRS Cluster radios require a high-assurance, secure architecture employing NSA Type 1 security, Common Criteria (CC) EAL 4+, and other stringent security architecture requirements [2]. The JTRS Cluster radio family includes embedded radios, such as Cluster 5, in which a fully featured, CORBA based SCA may not be economically or operationally feasible. In these situations, a lightweight SCA should be employed. This paper examines one approach – use of a remote waveform launcher – to dramatically reduce SCA run-time memory footprints required in the embedded radio component. The majority of the SCA resides on the remote host, typically a laptop computer of personal digital assistant (PDA). The remote host is less processor- and memory-constrained than the embedded radio(s), and more capable of hosting the SCA core framework (CF), operating environment (OE) and waveform resources. The launcher was prototyped using the open source SCA Reference Implementation (SCARI).

1. INTRODUCTION

Numerous approaches have been examined to reduce the memory footprint requirements for the deployment of the SCA Lite in embedded radios. These include:

1. Functionality reductions in the SCA, including limitation to "static" vs. "dynamic" waveform deployment and limitations of domain profile detail.

2. Classification of programmable resources as an SCA device, allowing the use of downloads designed for

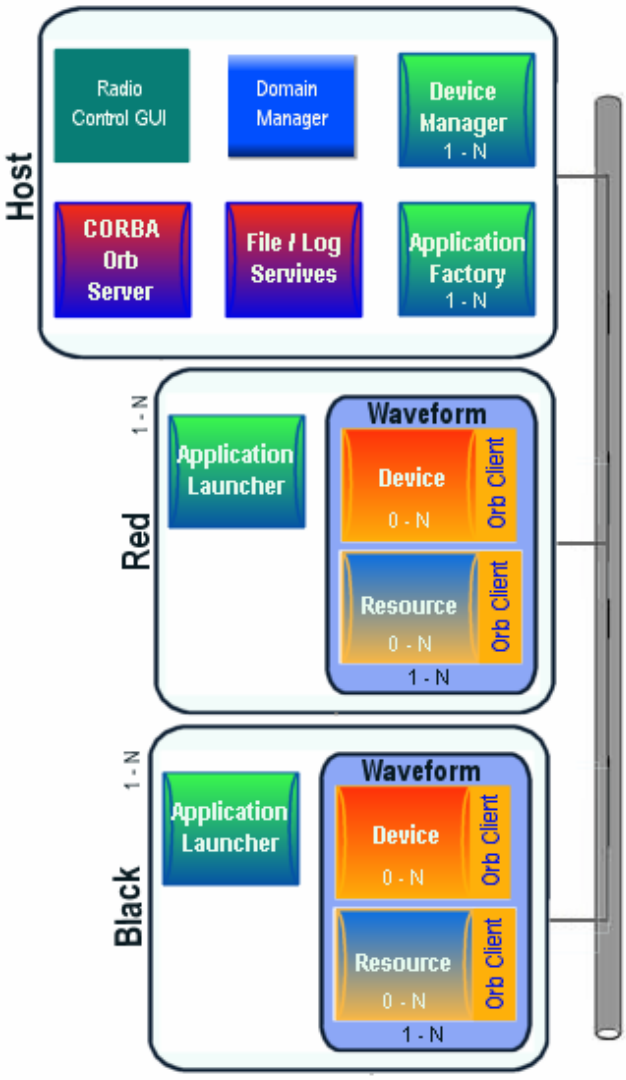

Figure 1. Application Launcher for SCA 2.2 red/black embedded radio.

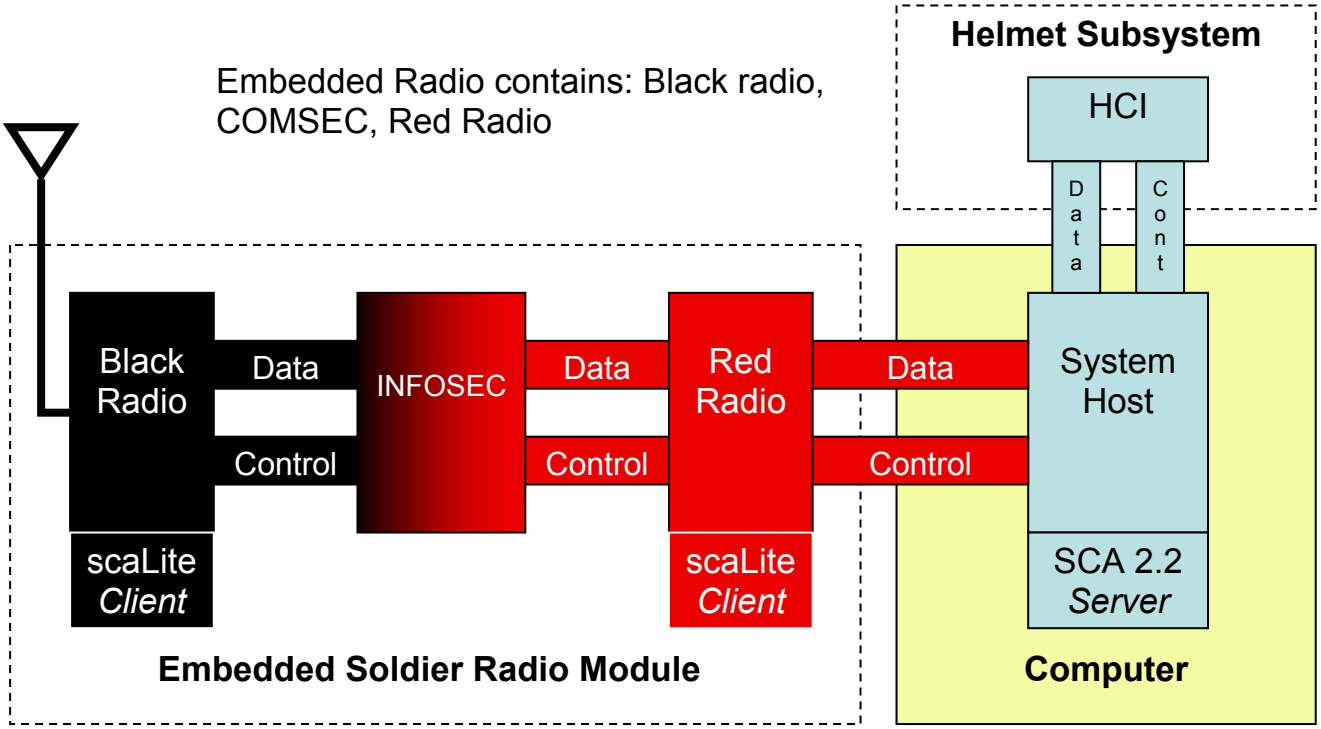

Figure 2. Embedded SCA 2.2 radio example (Land Warrior Soldier Radio Subsystem).

specific hardware (e.g. a field programmable gate array) at the expense of waveform portability

.

3. Selection of the lightest possible POSIX operating systems and minimum CORBA 2.2 object request brokers (ORBs) for dynamic SCA 2.2 compliance.

4. Replacement of CORBA with customized middleware.

5. Use of remote application launchers to deploy SCA compliant applications, such as waveforms, to embedded red/black radio subsystems, which remained logically connected to a remote red host (Figure 1).

While the first four methods are feasible, all require changes (in some cases significant) to the SCA. The fifth method preserves the full functionality and interoperability of the SCA and its applications; requires no changes to the SCA specification; requires minimal code additions to the software deployed on the host; and significantly reduces the run-time footprint of SCA components deployed on the embedded radio. The application launcher neither restricts operation to "static" (pre-compiled) waveforms, nor does it affect portability of waveforms and applications across different platforms.

An example of an embedded SCA radio set is the Land Warrior Soldier Radio Subsystem (Figure 2).

2. SCA APPLICATION LAUNCHER

A Remote Launcher is a device / server that resides on a remote client and is separate from the SCA Core Framework. It provides mechanisms to allow an application to be started on a processor or system that is remote from the heavy core framework. This requires that all needed files for the application and all libraries must either exist on a Lightweight client or the Launcher must provide a means of transferring and saving the files. The remote launcher represents the only overhead to the lightweight client that is beyond the need to run the required applications. All SCA functionality resides at the host and does not need to be replicated for each Core / Red / Black processor. The only running processes on the embedded radio cores will be the required waveforms and a very compact Launcher. The ORB and all other SCA 2.2 systems will operate on the host.

No changes to the SCA itself are required to implement remote launching of waveforms. Since the means and location of storing waveforms and the means of instantiating waveforms are platform dependent and are not specified in the SCA 2.2, doing so by the prescribed method remains fully SCA 2.2 compliant.

Remote launching of waveforms requires a launcher server on a remote location from the SCA CF that:

- Allow for built/compiled waveform files or byte-code to be saved/copied to the remote location.
- Allow for dynamic command line parameters to be passed to the remote location and then used to execute or launch the waveform or application.

A single call could support both functions. The launching server utilizes the same middleware as the waveform and SCA CF.

In the target SCA core framework, two points of entry for code modification are identified:

- The point at which the application file(s), byte-code(s), XML or etc. is installed (typically by the Domain manager).
- The point at which the application or waveform is instantiated (typically by the Application Factory helpers).

The location of these "entry points" could reside at the same location in the code base, depending on the SCA implementation. For the first entry point, the SCA CF identifies the location where the waveform file(s) must be copied. For the second entry point, the SCA CF identifies the location where the execution parameters must be passed for launching the waveform / application.

To identify the target location, the XML parameters in the waveform are examined. These parameters contain the information necessary to support the middleware being utilized and have been processed by the XML parser. The proper variables or objects are accessed for this information and depends on the core framework implementation.

In the case of CORBA, the XML <simple> tag with the id='EXECPARAM' in the xxx_PRF.xml file for the waveform contains two tags:

- <kind> with a value of "CORBA ORB" to signify this is the remote name reference when the ORB used is CORBA
- <values> with a value of the remote launcher name registered with the CORBA ORB name server. I.e., Red1, Black1, Red2, etc.

Once the location and means of making a call to the remote server at the target location are identified, the SCA CF makes that remote call dependant on the current point of entry action (Save files or start the application).

The data is then passed via the middleware to the remote server (port). Depending on the point of entry action, the remote server performs one of two actions:

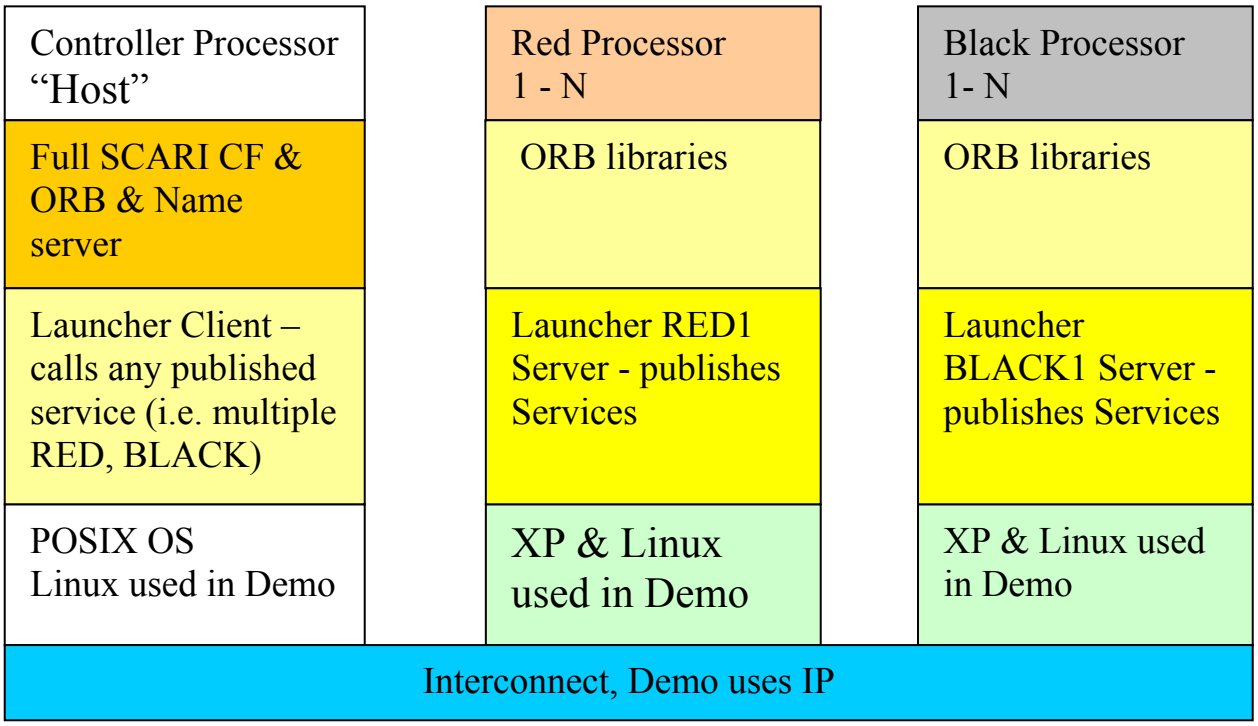

Figure 3. SCA Launcher demonstration architecture.

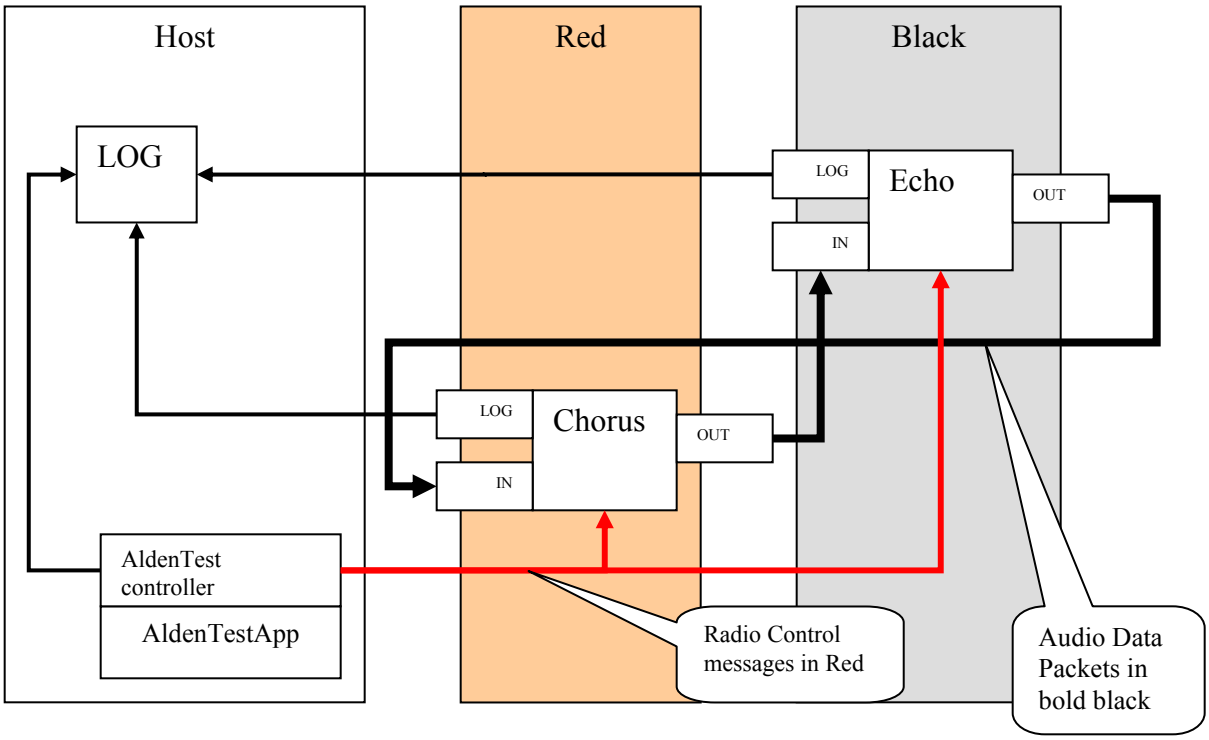

Figure 4. Sample waveform used for prototype evaluation.

- When requested to save a file, the remote launcher shall accept the data stream, save the file to the location it deems appropriate.
- When requested to instantiate a waveform or application, the launcher builds a command string from the data provided and instantiates the waveform by making the appropriate OS or POSIX call.

A remote launcher is a platform dependant component. Thus the location where files or data should be saved, and the means of instantiating or launching a waveform or application, are specifically coded for the target platform on which it executes.

3. LAUNCHER PROTOTYPE DESCRIPTION

A launcher prototype was developed using three personal computers to simulate an embedded red and black SCA radio system with a remote host (Figure 3). The prototype system architecture employs multiple launcher servers, one each for the Red1 and Black1 processor stacks. A simple test application was employed (Figure 4). One line of code in CRC's SCARI was modified to use the launcher client in the loading and starting of resources:

$SCA_HOME\demosources\devices\
ExecutableDeviceOperationsImpl.java

SCARI was also employed as the SCA CF on the host processor.

4. TEST RESULTS

Preservation of SCA functionality and interoperability with waveform applications developed for JTRS radio sets' employing "full" versions of the SCA without loss of runtime radio performance was achieved as the test waveform ran on both an un-altered SCARI single processor baseline system, and the three processor configuration of Host, RED1, and BLACK1.

Minimization of runtime memory footprint and hardware required to implement the remotely launched version of SCARI was achieved with minimum impact to the SCARI code base. The launcher was demonstrated using a platform independent approach running on a combination of WindowsXP and Linux. Using the SCA launcher with the SCARI implementation, the memory footprint requirement for run-time operation of the Red1 and Black1 embedded radio subsystems results in a 40% to 60% savings when compared to conventional methods (Figure 5 and Table 1). The ORB libraries and Java virtual machine (JVM) for Java 2 Standard Edition (J2SE) on Pentium processors occupy about 6 Mbytes on Linux, and 8 Mbytes on WindowsXP. Given the multiple JVM's invoked by the SCARI about 50 Mbyte of memory is required on the Host. SCARI's Java GUI programs add

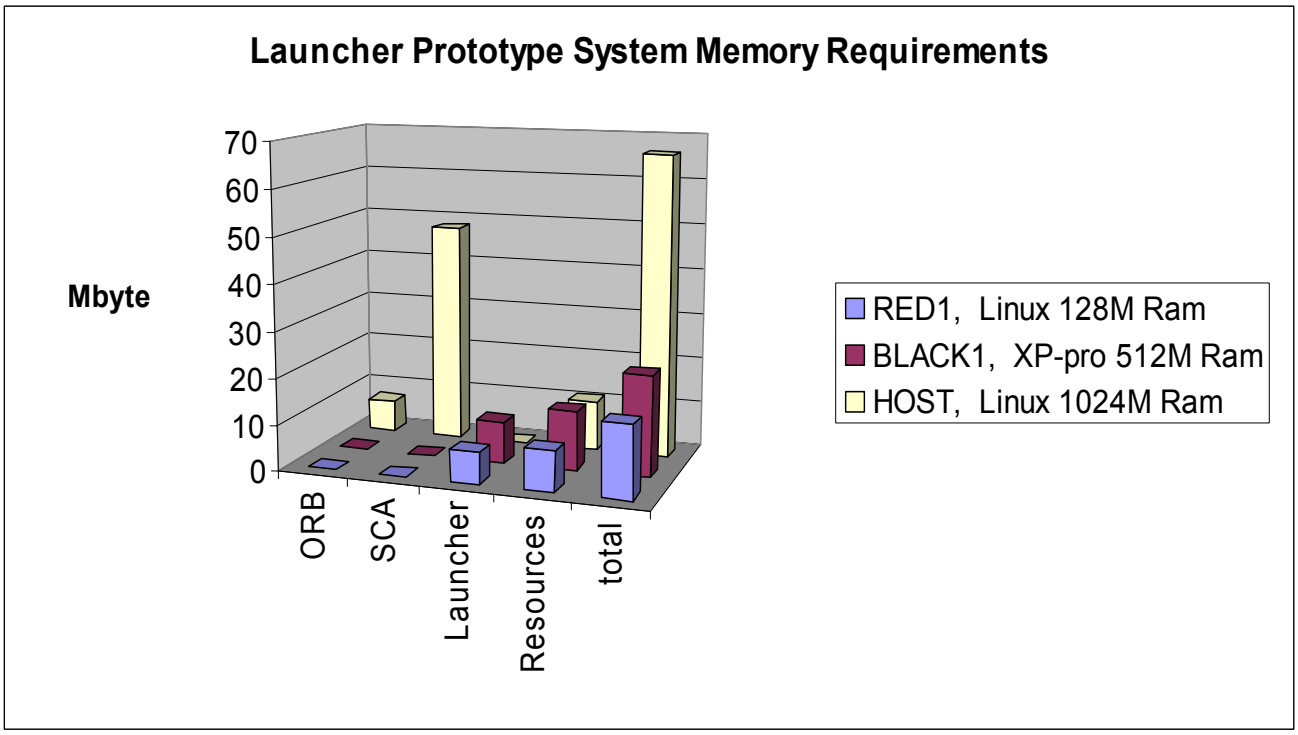

Figure 5. Memory requirements using Launcher J2SE

another 15 Mbyte of RAM requirements. The total memory required on the Host is about 65 Mbytes. By contrast, only 12 Mbyte was required on the RED1 radio, and 18 Mbytes required on the BLACK1 radio.

5. SUMMARY

Some JTRS SCA 2.2 compliant small form factor radios require a means of reducing the run-time memory requirement because of cost, power, and size constraints of the embedded radio hardware. In the case of embedded radios, which remain logically connected to an external host CPU and GUI, a set of "launcher" applications may be employed to allow remote initiation of SCA waveforms and applications by SCA CF and middleware located in the host CPU elsewhere in the system. In the case of a red/black embedded SCA radio, a compact launcher application (which includes a middleware port) and waveform resources are stored in each red and black embedded radio subsystem.

A single launcher application, located at the host CPU, is added to the SCA CF, middleware, and operating environment, distributing the SCA components and resources across the radio system. This same approach can also be used in other JTRS Clusters, and allows a single SCA 2.2 "server" to control several radio sets located throughout a platform, such as a ground vehicle, aircraft, or ship. The same approach can also be used in a data link system for unattended sensors, as long as logical connection is maintained between the sensors and host.

6. REFERENCES

[1] CRC task4_deliverable.doc October 30 2002 http://www.crc.ca/en/html/rmsc/home/sdr/projects/scari

[2] Draft Statement of work for small form factor radios, Cluster 5, JTRS JPO http://jtrs.army.mil/

[3] Jeffery H. Reed, Software Radio, a modern approach to Radio Engineering, Prentice Hall, New Jersey, 2002.

Table 1. Run-time memory requirements (MB) using SCA 2.2 Launcher and a simple demonstration waveform.

	ORB	SCA	Launcher	Resources	Total
RED1, Linux 128M Ram	0	0	6.97	8.83	**15.80**
BLACK1, XP-pro 512M Ram	0	0	9.05	12.77	**21.82**
HOST, Linux 1024M Ram	7.05	48.22	0	10.84	**66.10**
System total					*103.73*