An Innovator for Software Defined Systems

Performance of SCAv4.1 vs SCAv2.2.2

Presented by
Dr. Juan Pablo Zamora Zapata

WlnnComm Europe 2017
Oulu, Finland. May 17, 2017.
Outline

About NordiaSoft

SCA v4.1 vs. SCA v2.2.2 metrics

New SCA v4.1 features
Who is NordiaSoft?

- A strategic partner for the development of complex heterogeneous embedded distributed systems (HEDS)

- Located in Gatineau, Québec, Canada
  - All started at the Communications Research Centre Canada (CRC)
  - NordiaSoft was launched in 2013
  - Team with over 15 years of R&D in embedded system software

- Specialises in high-end HEDS products
  - Military and public safety radios
  - Test and Instrumentation equipment
  - Radar, Electronic Warfare, SigInt
  - Robotics, Control rooms
  - Transport (Automobile, Avionics, Train, Ship)
NordiaSoft Technology Around the World

- Technology licensed to over 50 clients in 16 countries
  - Americas, Europe, Middle-east, Asia
  - Over 10 waveforms deployed on thousands of SCA radios
NordiaSoft Partners

- Platforms Partners
  - Cobham AvComm
  - Ettus Research
  - Spectrum Signal Processing by Vecima
  - More coming...

- Real Time Software Partners
  - Wind River
  - Green Hills Software
  - Objective Interface Systems
  - More coming...

- Certification Testing Partners
  - Reservoir Labs
  - More coming...
NordiaSoft’s Team: List of Industry Firsts...

2017
- 1st **Embedded Components (eCo) Suite for SCAv4.1**

2016
- 1st **SCA Test Instrument**: Cobham Modular Platform (CMP)

2015
- 1st **SCA OpenCL demonstration** (GPP, GPU, FPGA)

2013
- 1st **SCA on Android Handheld demonstration** (AM, FM, APCO Project 25)

2011
- 1st **Android-based SCA waveform implementations**

2010
- 1st **SCA-based Virtual Front Panel**

2008
- **SCARI-GT**: New generation Core Framework for small form factors

2007
- 1st **SCA Radio demo** using the world’s smallest computer (Gumstix)

2006
- Added support for LynxOS
- SCA Architect™ Eclipse-based integrated modeling tool
- Added support for VxWorks and QNX

2005
- 1st to introduce XML validation and code generation
- Added support for ORBexpress, INTEGRITY, and YellowDog

2004
- **SCARI++**, full C++ SCAv2.2 CF for Linux/TAO
- Open Source SCARI2, JTeL Tested (97.39%) SCAv2.2 CF

2003
- **SCARI-Hybrid**, CRC’s 1st commercial solution with modeling tools

2002
- 1st demo of a commercial SCA application (**DAB™**)

2001
- 1st **SCA Reference Implementation** (**SCA – RI**)

2000
- Introduced the concept of “Ports” and “connections” for **SCA**v1.0

1998
- Implemented **SCA**v0.3 FM-LoS demo for DND.
- Designed proprietary SDR architecture
Outline

About NordiaSoft

SCA v4.1 vs. SCA v2.2.2 metrics

New SCA v4.1 features
Test bench characteristics

▪ Processor
  - NXP iMX.6Quad 1GHz (ARM Cortex-A9 Quad Core)

▪ Memory
  - DDR3-1066, 2GByte 533Mhz, 64 bit bus

▪ Storage
  - SDHC, speed 10, 32GByte

▪ OS
  - Linux 3.14 (IMX6 QD)

▪ OE
  - NordiaSoft 3rd generation SCARI CF v2.2.2 vs NordiaSoft 1st generation Embedded Components (eCo) Hub CF v4.1
Outline

Node Boot Up: Device Registration
Assembly Deployment: Mass Connections
Application Deployment: Application Creation
Feature: Registration

- **SCAv4.1 uses push only registration**
  - SCAv2.2.2 was mostly implemented by letting key components pull the information they needed
  - SCAv4.1 is focused on allowing components to provide more information at registration to avoid pulling
  - This feature can save several interactions to copy metadata files over embedded file systems (major concern for radios with slow file systems)
  - It can also help avoid reparsing of some XML information
  - The result is a faster boot sequence
1. *Device* registers with its *DeviceManager*  
2. *DeviceManager* registers *Device* with *DomainManager*  
3. *DomainManager* requests *Device* info from DeviceManager  
4. *DomainManager* requests from *Device* Software Profile (SPD/PRF) to extract advertised capabilities
1. Device registers with its DeviceManager
2. DeviceManager registers Device with DomainManager
# Feature: Registration

## Registration time (ms) vs. Number of Components

![Graph showing registration time vs. number of components for SCAv2.2.2 and SCAv4.1.](image)

<table>
<thead>
<tr>
<th># Components</th>
<th>Time (ms) SCAv2.2.2</th>
<th>Time (ms) SCAv4.1</th>
<th>Improvement %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>55.705</td>
<td>53.263</td>
<td>4.38%</td>
</tr>
<tr>
<td>3</td>
<td>121.93</td>
<td>62.075</td>
<td>49.09%</td>
</tr>
<tr>
<td>5</td>
<td>182.928</td>
<td>65.246</td>
<td>64.33%</td>
</tr>
<tr>
<td>10</td>
<td>333.542</td>
<td>126.867</td>
<td>61.96%</td>
</tr>
</tbody>
</table>
Outline

Node Boot Up: Device Registration

Assembly Deployment: Mass Connections

Application Deployment: Application Creation
Feature: Mass Connections

- SCAv4.1 also supports the push approach to establish connections between components
  - Components can register all their ports during registration
  - Connections can be established in bulk
  - The result is a shorter connection sequence

- Mass connections can provide substantial improvements for secure radios
  - Secure radios need to keep red and black information separated
  - The Multiple Independents Levels of Security (MILS) is often used
  - MILS relies on RTOS with separate partitions (like ARINC 653)
  - ...
Feature: Mass Connections

- Mass connections can provide tremendous improvements for secure radios (…)
  - Using SCAv2.2.2, making a connection requires 2 calls to getPort() and 1 call to connect()
  - CORBA is fast, on normal systems, many connections can be performed in milliseconds
  - However, establishing connections between components hosted in different partitions can require up to 4 rounds of the secure scheduler
    - Can be very slow for several connections
    - Typical systems has minimum of 20 connections, some have over 100
Feature: Mass Connections

- Mass connections can provide tremendous improvements for secure radios (...)

Round #1
- DomainManager calls `getPort()` on first component
- First component responds to `getPort()`

Round #2
- DomainManager receives first response from `getPort()`
- DomainManager calls `getPort()` on second component
- Second component responds to `getPort()`

Round #3
- DomainManager receives second response from `getPort()`
- DomainManager calls `connect()` on second component
- Second component responds to `connect()`

Round #4
- DomainManager receives response from `connect()`
Feature: Mass Connections

- **SCAv2.2.2**

```
portA1 = component_A.getPort("Port1")
portB1 = component_B.getPort("Port1")
portA1.connectPort(portB1, "toB1")
portA2 = component_A.getPort("Port2")
portB2 = component_B.getPort("Port2")
portA2.connectPort(portB2, "toB2")
```
Feature: Mass Connections

- SCAv4.1

Connection

All provides port references are obtained at registration

component_A.connectUsesPorts("Port1", "toB1", "Port2", "toB2"…)
## Feature: Mass Connections

### Average Time to Make Connections (ms)

![Graph showing average time to make connections for SCAv.2.2.2 and SCAv.4.1](image)

<table>
<thead>
<tr>
<th># of connections</th>
<th>SCAv.2.2.2</th>
<th>Average time per connection</th>
<th>SCAv.4.1</th>
<th>Average time per connection</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 connection</td>
<td>1</td>
<td>3.852</td>
<td>1.118</td>
<td>2.145</td>
<td>0.133</td>
</tr>
<tr>
<td>3 connections</td>
<td>3</td>
<td>6.465</td>
<td>1.307</td>
<td>2.288</td>
<td>0.071</td>
</tr>
<tr>
<td>5 connections</td>
<td>5</td>
<td>8.153</td>
<td>1.075</td>
<td>2.381</td>
<td>0.118</td>
</tr>
<tr>
<td>10 connections</td>
<td>10</td>
<td>12.614</td>
<td>0.974</td>
<td>2.563</td>
<td>0.209</td>
</tr>
<tr>
<td>Base Cost</td>
<td></td>
<td>2.734</td>
<td></td>
<td>2.012</td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Node Boot Up: Device Registration
- Assembly Deployment: Mass Connections
- Application Deployment: Application Creation
Feature: Application Creation

- Application Deployment Time is Paramount
  - SCAv4.1 utilizes less component interactions to launch SCA Applications
  - SCAv4.1 reduces footprint while preserving equivalent functionality
  - The result is a faster application creation
Feature: Application Creation

<table>
<thead>
<tr>
<th>Number of Resources</th>
<th>SCAv2.2.2</th>
<th>SCAv4.1</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>136.661</td>
<td>104.147</td>
<td>23.79%</td>
</tr>
<tr>
<td>3</td>
<td>408.392</td>
<td>304.178</td>
<td>25.52%</td>
</tr>
<tr>
<td>5</td>
<td>685.381</td>
<td>495.727</td>
<td>27.67%</td>
</tr>
<tr>
<td>10</td>
<td>1392.28</td>
<td>989.534</td>
<td>28.93%</td>
</tr>
</tbody>
</table>
Outline

About NordiaSoft

SCA v4.1 vs. SCA v2.2.2 metrics

New SCA v4.1 features
Outline

- Core Assignment
- Component Factories
- Process Allocation
- Backwards Compatibility
- Optional Composition
Core Assignment

- SCAv2.2.2 does not support Core Assignment
- SCAv4.1 leverages full capabilities of multicore platforms
  - Deterministic control over component deployment
  - Override core allocation control from the OS

- Component Core Assignment
  - Optimize throughput (core collocation)
  - Maximize parallel processing (core separation)

- Overall gains
  - Enhance performance by distributing execution across multiple cores
  - Reduce costs by fully utilizing existing computing infrastructure
  - Save power by requiring fewer processors
Component Factories

- **SCAv2.2.2 only supported factories for application components**
  - Allows several components to be co-located in a same program space
  - Provides foot print savings
  - Provides better throughput

- **SCAv4.1 generalizes the Factory concepts to both node and application components**
  - Node components can now be co-located together as well
Component Factories

- The feature can also drastically accelerate communications between components with good real time ORBs

<table>
<thead>
<tr>
<th>Average Round Trip Time in usec for PPC405GPr (400MHz) running INTEGRITY RTOS and ORBexpress</th>
<th>Double Sequence</th>
<th>Octet Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1024</td>
<td>2048</td>
</tr>
<tr>
<td>using TCP/IP</td>
<td>3334</td>
<td>1428</td>
</tr>
<tr>
<td>using INTCOHN</td>
<td>2215</td>
<td>1042</td>
</tr>
<tr>
<td>using direct method invocation thanks to a ResourceFactory that yielded 40% smaller footprint</td>
<td>244</td>
<td>155</td>
</tr>
</tbody>
</table>

May 2017
All Rights Reserved. © 2017 NordiaSoft.
Process Allocation

- Allows developers to define within which process a component must be launched
  - Create any number of program spaces to co-locate any number of components
  - Creation of program processes is completely dynamic

- Allows application components to be co-located with node components
  - Allows a high-data rate node component to feed an application within a same program space

- Overall: Offers substantial potential for performance optimization
Backwards compatibility

- Launch SCAv2.2.2 applications on a SCAv4.1 platform
  - Backwards compatibility with SCAv222
Backwards compatibility

- Launch SCAv2.2.2 application components as part of SCAv4.1 applications
  - NordiaSoft extended feature to allow customers to make a progressive transition from SCAv222 to SCAv4
  - Avoid the cliff jump approach
Backwards compatibility

- Launch SCAv2.2.2 application components as part of SCAv4.1 applications (...)
  - Why force the port of all the application components before they can be tested? The eCo Core Framework supports application made of a mixture of SCAv2.2.2 and SCAv4.1 components

![Diagram of backwards compatibility]

May 2017
All Rights Reserved. © 2017 NordiaSoft.
Backwards compatibility

- The eCo Core Framework even supports a mixture of SCAv2.2.2 node and application components
  - You can reuse cards/boards from SCAv2.2.2 radios to transition towards a SCAv4.1 compliant radio
Backwards compatibility

Many Functional Points

SCA V4.0
SCA V4.1

V4 CF and Platform

V4 Application Support

Full V4 System
Optional Composition

- **SCAv4.1 supports varying levels of granularity for components**
- **Components can implement only the standard interfaces that are required**
  - Ex: a component with no properties doesn’t have to implement the PropertySet interface

- **This can also help address some Information Assurance (IA) requirements**
  - No dead/stubbed code for unsupported APIs
  - No interface to provide information that should not be provided
Conclusion: why use SCAv2.2.2?

- **SCAv4.1 provides more features**
  - Adds support for multi-core processors that powers every recent consumer electronics devices
  - Offers more control over information assurance

- **Offers better performances at smaller footprints**
  - Faster startup times thanks to bulk connections and push registration
  - Smaller footprints and better performances thanks to component factories and process space allocation

- **Supports backwards compatibility**
  - SCAv2.2.2 applications can run on SCAv4.1 platforms
The End