Can SCA 4.1 Replace STRS in Space Applications?

Ran Cheng, Li Zhou, Qi Tang, Dongtang Ma, Haitao Zhao, Shan Wang and Jibo Wei

National University of Defense Technology
May 17, 2017
1. Introduction
2. Core Framework Design
3. Testbed Implementation
4. Experiment Results and Analysis
5. Conclusion
Expectations
• Reduce the development cycle-time
• Reduce the development cost
• Increase the communication flexibility among space radios and ground stations

Challenges
• Strict SWaP constrained requirement
• Low capability of chips for space use
• Reuse of hardware and software
### Software Execution Model Comparison

#### SCA 2.2.2

- **Waveform Applications and High Level Services**
  - CORBA APIs
  - CF APIs
  - CORBA
  - Core Framework (CF)
  - AEP
  - HAL API
  - BSP
  - Drivers
  - GPM
  - Specialized HW

#### STRS

- **Waveform Applications and High Level Services**
  - POSIX APIs
  - STRS APIs
  - AEP
  - Transfer Mechanism and Services
  - STRS Infrastructure
  - HAL API
  - BSP
  - Drivers
  - GPM
  - Specialized HW

#### SCA 4.1

- **Waveform Applications and High Level Services**
  - POSIX APIs
  - CF APIs
  - AEP
  - Transfer Mechanism and Services
  - CF
  - HAL API
  - BSP
  - Drivers
  - GPM
  - Specialized HW
SCA Evolution from 2.2.2 to 4.1

- Adopt “push model” behavior
- Remove dependence on CORBA
- Add static registration behavior
- Provide Units of Functionality (UOF) and SCA Profiles

The year of announcement of several SCA versions

SCA Development and Evolution
Layered Framework Comparison

SCA 4.1 layered framework

Core Module
- File Manager
  - File Operation/Management
- Domain Manager
  - Waveform Install/Delete
  - Waveform Application Manager
  - Waveform Control Manager
  - Domain Configuration Manager
- Device Service Manager
  - Device Login/Logout
  - Device Management Control
  - Service Login/Logout
  - File System Management

Basic Module
- Domain Description Parser Module
- Component Connection Management module
- Component Deployment Management Module
- Waveform Reconstitution Module
- ORB Encapsulation Module
- Waveform Development Public Library

STRS layered framework

Core Module
- File Manager
  - File Maintenance Management
- Domain Manager
  - Waveform Install/Delete
  - Waveform Application Manager
- Device Service Manager
  - Device Login/Logout
  - Device Management Control
  - Service Login/Logout
- File Operation Interface

Basic Module
- Configuration Parser Module
- Component Connection Management module
- Component Deployment Module
- Command Control Management Module
- Debug Information Log Module
- Message-oriented Middleware
- Waveform Development Public Library
SCA 4.1 core framework written in C++ has complex inheriting relationship

Infrastructure API

- Basic Application Interfaces
  - ComponentIdentifier
  - LifeCycle, etc.
- Basic Device Interfaces
  - AggregateDevice
  - CapacityManagement, etc.
- Framework Control Interfaces
  - ApplicationManager
  - DeploymentAttributes, etc.
- Framework Service Interfaces
  - ComponentFactory
  - FileManager, etc.
STRS core framework written in C simplifies the implementation

STRS core framework interfaces

**Time Control**
- STRS_GetNanoseconds()
- STRS_GetSeconds()
- STRS_GetTime()
- STRS_GetTimeWarp()
- STRS_SetTime()
- STRS_Synch()

**File Control**
- STRS_FileClose()
- STRS_FileGetFreeSpace()
- STRS_FileGetSize()
- STRS_FileGetStreamPointer()
- STRS_FileOpen()
- STRS_FileRemove()
- STRS_FileRename()

**Application Control**
- STRS.Configure()
- STRS.GetMultiple()
- STRS.Initialize()
- STRS_ReleaseObject()
- STRS_ReturnTest()
- STRS_Synch()
- STRS_Stop()

**Messaging**
- STRS_QueueCreate()
- STRS_QueueDelete()
- STRS_Register()
- STRS_Unregister()

**Application Setup**
- STRS.AbortApp()
- STRS.GetErrorQueue()
- STRS.HandleRequest()
- STRSInstantiateApp()
- STRS.IsOK()
- STRS_Log()

**Device Control**
- STRS_DeviceClose()
- STRS_DeviceFlush()
- STRS_DeviceLoad()
- STRS_DeviceOpen()
- STRS_DeviceReset()
- STRS_DeviceStart()
- STRS_DeviceStop()
- STRS_DeviceUnLoad()
- STRS_SetISR()

**Data Sink**
- STRS_Write()

**Data Source**
- STRS_Read()
Comparison Aspects and Variables

Comparison aspects:
- the static memory occupation
- inter-components communication delay
- waveform deployment delay
- waveform switching delay

Variables:
- the packet size
- the total amount of packets
- the number of components
Agency Mechanism

- The stub and skeleton combine to form the RMI frame protocol.
- The remote reference layer is adopted to find the communication object.
- The transport layer provides the interconnection of client and server based on the TCP/IP protocol.

Full SCA 4.1

RMI: Remote Method Invocation
Testbed Introduction

**Testbed:** ZLSDR-1000

**Main chip:** ZYNQ 7030
- a dual-core of ARM Cortex-A9 (clock speed 667MHz)
- a FPGA of Kintex-7 (logic cells 125K, DSP Slices 400)

**DDR memory size:** 1GB

**Operating system:** Linux 3.17
The inter-components communication time of a single link $T = T_2 - T_1$ is zero. The amount time of all links in inter-components communication $T = T_N - T_1$. 

Inter-components communication delay:

- Component processing delay
- Inter-components communication delay

$T_1$ to $T_N$
Waveform deployment delay and waveform switching delay

Waveform deployment delay:
1. Waveform Upload
2. Waveform Initialize
3. Waveform Launch

Waveform switching delay:
1. Waveform stop
2. Waveform unload
3. Waveform Upload
4. Waveform Initialize
5. Waveform Launch
### Experiment parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of components</td>
<td>3, 4, ..., 11</td>
</tr>
<tr>
<td>Packet size (bytes)</td>
<td>128, 256, 512, 1024, 2048, 4096</td>
</tr>
<tr>
<td>Amount of packets (10^6)</td>
<td>1, 2, ..., 10</td>
</tr>
</tbody>
</table>
## Static memory occupation comparison

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Static memory occupation (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRS</td>
<td>4.77</td>
</tr>
<tr>
<td>Lightweight SCA 4.1</td>
<td>27.82</td>
</tr>
<tr>
<td>Full SCA 4.1</td>
<td>77.20</td>
</tr>
</tbody>
</table>
## Experiment Results

### Inter-components communication delay comparison with different packet size

<table>
<thead>
<tr>
<th>Package size (bytes)</th>
<th>STRS (us)</th>
<th>Lightweight SCA (us)</th>
<th>Full SCA 4.1 (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>17.30</td>
<td>17.20</td>
<td>189.00</td>
</tr>
<tr>
<td>256</td>
<td>17.30</td>
<td>18.40</td>
<td>189.00</td>
</tr>
<tr>
<td>512</td>
<td>17.30</td>
<td>19.00</td>
<td>196.00</td>
</tr>
<tr>
<td>1024</td>
<td>17.30</td>
<td>20.80</td>
<td>200.00</td>
</tr>
<tr>
<td>2048</td>
<td>17.30</td>
<td>21.00</td>
<td>216.00</td>
</tr>
<tr>
<td>4096</td>
<td>17.50</td>
<td>21.00</td>
<td>240.00</td>
</tr>
</tbody>
</table>
## Experiment Results

### Total time consumption comparison with different amount of packets

<table>
<thead>
<tr>
<th>Amount of packets</th>
<th>STRS (s)</th>
<th>Lightweight SCA 4.1 (s)</th>
<th>Full SCA 4.1 (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,000</td>
<td>17.50</td>
<td>21.00</td>
<td>240.00</td>
</tr>
<tr>
<td>200,000</td>
<td>40.10</td>
<td>42.20</td>
<td>476.00</td>
</tr>
<tr>
<td>300,000</td>
<td>66.70</td>
<td>63.60</td>
<td>715.00</td>
</tr>
<tr>
<td>400,000</td>
<td>81.60</td>
<td>85.00</td>
<td>946.00</td>
</tr>
<tr>
<td>500,000</td>
<td>101.90</td>
<td>105.40</td>
<td>1190.00</td>
</tr>
<tr>
<td>600,000</td>
<td>119.20</td>
<td>125.40</td>
<td>1441.00</td>
</tr>
<tr>
<td>700,000</td>
<td>143.10</td>
<td>147.80</td>
<td>1669.00</td>
</tr>
<tr>
<td>800,000</td>
<td>164.90</td>
<td>162.20</td>
<td>1905.00</td>
</tr>
<tr>
<td>900,000</td>
<td>184.10</td>
<td>191.00</td>
<td>2142.00</td>
</tr>
<tr>
<td>1,000,000</td>
<td>201.70</td>
<td>207.40</td>
<td>2399.00</td>
</tr>
</tbody>
</table>
### Experiment Results

#### Inter-components communication time comparison with different numbers of components

<table>
<thead>
<tr>
<th>Number of components</th>
<th>STRS (us)</th>
<th>Lightweight SCA 4.1 (us)</th>
<th>Full SCA 4.1 (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>15.50</td>
<td>23.00</td>
<td>175.00</td>
</tr>
<tr>
<td>4</td>
<td>16.00</td>
<td>19.67</td>
<td>186.67</td>
</tr>
<tr>
<td>5</td>
<td>16.75</td>
<td>20.50</td>
<td>197.50</td>
</tr>
<tr>
<td>6</td>
<td>16.00</td>
<td>19.80</td>
<td>208.00</td>
</tr>
<tr>
<td>7</td>
<td>16.50</td>
<td>21.33</td>
<td>211.67</td>
</tr>
<tr>
<td>8</td>
<td>16.43</td>
<td>20.14</td>
<td>218.57</td>
</tr>
<tr>
<td>9</td>
<td>17.00</td>
<td>20.75</td>
<td>227.50</td>
</tr>
<tr>
<td>10</td>
<td>17.22</td>
<td>20.22</td>
<td>233.33</td>
</tr>
<tr>
<td>11</td>
<td>17.50</td>
<td>21.00</td>
<td>240.00</td>
</tr>
</tbody>
</table>
Waveform deployment delay comparison with different numbers of components

<table>
<thead>
<tr>
<th>Number of components</th>
<th>STRS (ms)</th>
<th>Lightweight SCA 4.1 (ms)</th>
<th>Full SCA 4.1 (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>46.7</td>
<td>843.3</td>
</tr>
<tr>
<td>4</td>
<td>13.3</td>
<td>51</td>
<td>1078</td>
</tr>
<tr>
<td>5</td>
<td>14.7</td>
<td>55</td>
<td>1318</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>58.7</td>
<td>1588</td>
</tr>
<tr>
<td>7</td>
<td>17.7</td>
<td>62</td>
<td>1841.8</td>
</tr>
<tr>
<td>8</td>
<td>18.7</td>
<td>65.3</td>
<td>2130</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>69.7</td>
<td>2385</td>
</tr>
<tr>
<td>10</td>
<td>20.5</td>
<td>73</td>
<td>2641.6</td>
</tr>
<tr>
<td>11</td>
<td>21</td>
<td>79.6</td>
<td>2903.7</td>
</tr>
</tbody>
</table>
## Waveform switching delay comparison with different numbers of components

<table>
<thead>
<tr>
<th>Number of components</th>
<th>STRS (ms)</th>
<th>Lightweight SCA 4.1 (ms)</th>
<th>Full SCA 4.1 (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>13</td>
<td>51.6</td>
<td>880.6</td>
</tr>
<tr>
<td>4</td>
<td>14.6</td>
<td>56</td>
<td>1124</td>
</tr>
<tr>
<td>5</td>
<td>15.7</td>
<td>60</td>
<td>1376.3</td>
</tr>
<tr>
<td>6</td>
<td>17.7</td>
<td>64.4</td>
<td>1661.6</td>
</tr>
<tr>
<td>7</td>
<td>19.4</td>
<td>67.7</td>
<td>1930.4</td>
</tr>
<tr>
<td>8</td>
<td>20.4</td>
<td>71.3</td>
<td>2234</td>
</tr>
<tr>
<td>9</td>
<td>21.6</td>
<td>76.7</td>
<td>2502</td>
</tr>
<tr>
<td>10</td>
<td>22.2</td>
<td>79</td>
<td>2769.6</td>
</tr>
<tr>
<td>11</td>
<td>23</td>
<td>85.9</td>
<td>3035.1</td>
</tr>
</tbody>
</table>
Lightweight SCA 4.1 and STRS have very close transfer efficiency.

Their transfer delay is almost 1/10 of full SCA 4.1.
Experiment Analysis

Reasons of low efficiency in full SCA 4.1:

• the creation and communication between processes

• the adoption of object request broker (ORB) mode
Experiment Analysis

**STRS Advantages:**
- high efficiency
- the ability to suit the resource-limited lightweight platform

**STRS Disadvantages:**
- low portability and interoperability
- difficulty for the application developer

**SCA 4.1 Advantages:**
- high flexibility
- providing different SCA Profiles
- reducing the expenditure with thread communication

**SCA 4.1 Disadvantages:**
- high waveform deployment delay
- high waveform switching delay
- large static memory occupation
Recommendation:

• Lightweight SCA 4.1 is worth considering for space radios owing to its high efficiency.
• Users should pay attention to waveform deployment, switching delay and static memory occupation.

Future work:

• Carrying out experiments in platform with limited computing capability and memory size.
• Evaluate the power consumption of different architectures.
Thank you!

Q&A