Porting AMS to RTEMS

Utilizing the Operating System Abstraction Layer (OSAL)

David Edell
Overview

- What is AMS?
- VxWorks vs RTEMS
- What is OSAL?
- APL AMS Implementation (history)
- Porting to OSAL in VxWorks
- Porting to OSAL in RTEMS
- Conclusions
Asynchronous Messaging Service (AMS)

- AMS is a software bus architecture facilitating communication between applications.
  - Applications are known as ‘modules’
  - Modules may be local to a single processor or distributed across multiple CPUs.
- Registration Servers (RS) monitor module health
- A configuration server (CS) monitors registrar health and directs modules to the appropriate RS
- An extension of AMS, called RAMS, provides the ability to communicate over larger distances, for example using a Delay Tolerant Network (DTN).

- The AMS protocol is an official CCSDS Standard.
  - See CCSDS 735.1-B-1 from www.ccsds.org
AMS Overview
Real-Time Operating Systems

- WindRiver VxWorks is a RTOS used on several past APL missions.
  - Partial POSIX Support

- RTEMS is an open-source RTOS of increasing popularity within the space community.
  - Full POSIX Support
An APL implementation of AMS was started in 2008.

- The goal was to construct a software bus for usage in VxWorks between memory management unit (MMU) protected processes, known as Real Time Processes (RTPs).
- POSIX message queues were used as the primary transport service to facilitate future transition to other systems.
- Support for alternative transport services was planned but never implemented.
- Implementation was later extended to support both kernel (non-MMU) and user (MMU) modes using compiler directives.
VxWorks to RTEMS Porting Process

- Existing support for MMU and non-MMU operation resulted in messy compiler directives.
  - This was exacerbated by issues with VxWorks POSIX support in kernel mode.
  - RTEMS support would add additional cases.

- An abstraction layer can simplify the code base while supporting this transition
  - No components of the AMS Library are inherently OS-specific.
  - An abstraction layer allows a single code base to be used across platforms.
  - GSFC’s Operating System Abstraction Layer (OSAL) is an open source library
What is the OSAL?

- The Operating System Abstraction Layer (OSAL) is an open-source library created by Goddard Space Flight Center (GSFC)
  - [http://osal.sf.net](http://osal.sf.net)
- OSAL provides support for VxWorks, RTEMS and Linux. It can also run under Windows/Cygwin.
- OSAL is a key component in GSFC’s Core Flight Executive (cFE).
  - cFE is a software bus system with supporting services being used on several upcoming APL and GSFC missions.
- OSAL provides wrappers for standard OS functions, including I/O operations, queues, and semaphores. Build settings select the platform-specific version to be loaded.
Porting APL-AMS to use the OSAL

- We began porting APL-AMS to the OSAL under VxWorks
  - All native functions were replaced with OSAL equivalents.
  - A new transport service was created utilizing OSAL message queues (osmq).
    - OSAL Queues are wrappers for native message queues. Native calls typically have less overhead than POSIX systems.
    - The original POSIX message queues can be used as an alternative transport service on supporting platforms.
  - VxWorks specific debug functions were removed or replaced.
  - VxWorks task management functions replaced with OSAL equivalents.
RTEMS Transition

- Using the OSAL, changing OS requires only a change to the build configuration
  - At least in theory

Initial Compilation and testing under RTEMS revealed several issues.

- VxWorks-specific debug function (logMsg) was not removed.
- VxWorks errnoGet() must be changed to C Standard errno from errno.h, also available in VxWorks
- A few minor bugs in the AMS Library.
  - These issues were present in the VxWorks version, but did not present issues in that environment.
Application Porting

- Representative Flight Applications were ported to AMS
  - Original applications were written for cFE, nominally running vxWorks
  - The Core Flight Executive (cFE) is GSFC’s flight software architecture, including a software bus and application services.
    - cFE utilizes the OSAL, combined with a platform support package, to support multiple platforms.
The Scheduler (SCH), Command Ingest (CI), and Telemetry Output (TO) applications were ported.

- OSAL Calls were unchanged
- cFE Services were removed or replaced.
- Application framework and initialization converted to the AMS API.
- cFE Software bus calls were replaced with AMS Equivalents
  - APL-AMS supports a ‘synchronous’ and an ‘asynchronous’ style API.
  - Synchronous style can be directly mapped to cFE SB framework.
Questions?
OSAL Conversion Details

- Differences in MQ syntax

Next Slide:
- RTEMS Conversion
  - Revealed additional platform-dependencies to be changed, such as difference between errno.h and errnoLib.h (vxWorks-specific)
  - Conversion to ROS build system
  - Initial testing revealed software bugs. Further investigation showed that these were implementation bugs that did not manifest themselves under VxWorks, though they were equally applicable.

Next Slide: Application Porting

Conclusions