SPACE SYSTEMS SYMPOSIUM (D1)
Interactive Session on Space Systems (I)

Author: Mr. Alan Mick
The John Hopkins University Applied Physics Laboratory, United States, Alan.Mick@jhuapl.edu

Mr. Edward Birrane
The John Hopkins University Applied Physics Laboratory, United States, Edward.Birrane@jhuapl.edu
Mr. Christopher Krupiarz
The John Hopkins University Applied Physics Laboratory, United States,
Christopher.Krupiarz@jhuapl.edu

DISTRIBUTED FLIGHT SOFTWARE COMPUTING INFRASTRUCTURE USING THE CCSDS ASYNCHRONOUS MESSAGE SERVICE STANDARD

Abstract

The use of distributed computing in flight software applications is well established, but is generally implemented using static point-to-point connections between process managing resources and those that utilize those resources to accomplish some specific task. This approach does not allow for easy, efficient reassignment of resources that may not be serving their primary purpose for long periods of time. During those periods that the resource may be underutilized, the point-to-point connections can not be easily reconfigured to make those resources more generally available.

One approach to distributed computing that aids in decoupling tasks and allows for spontaneous reconfiguration to better utilize available resources is the "publish and subscribe" or "software bus" approach to task interaction. In this approach a middle-ware layer allows information sources to publish messages of various types and information consumers to subscribe to specific messages without having to create explicit connections. This allows more flexible use of resources as loading increases and decreases in a system. The CCSDS has recently advanced a standard protocol for publish and subscribe interactions for flight software, the Asynchronous Message Service (AMS).

The JPL reference implementation of the AMS protocol was used to implement a generalized, customizable framework for distributed computing that allows run time discovery and utilization of under-utilized resources under a real-time version of the Linux operating system on the MCP-750 single board computer. The framework defines three generalized task types or roles – the "shell", which accepts and manages a high-level command stream, the "exec" which is responsible for carrying out specific commands, and the "worker" which manages particular resources and provides services that are used by the "exec." It makes use of several of the AMS protocol's features to provide an approach to resource discovery and utilization that can be easily reused for a variety of tasks in a variety of computational environments.

The general features of the AMS protocol, its strengths in support of resource discovery and utilization and the design and implementation of the framework are explained and discussed.