Software

Asynchronous Message Service Reference Implementation

This software provides a library of middleware functions with a simple application programming interface, enabling implementation of distributed applications in conformance with the CCSDS AMS (Consultative Committee for Space Data Systems Asynchronous Message Service) specification.

The AMS service, and its protocols, implement an architectural concept under which the modules of mission systems may be designed as if they were to operate in isolation, each one producing and consuming mission information without explicit awareness of which other modules are currently operating. Communication relationships among such modules are self-configuring; this tends to minimize complexity in the development and operations of modular data systems.

A system built on this model is a "society" of generally autonomous, inter-operating modules that may fluctuate freely over time in response to changing mission objectives, modules' functional upgrades, and recovery from individual module failure. The purpose of AMS, then, is to reduce mission cost and risk by providing standard, reusable infrastructure for the exchange of information among data system modules in a manner that is simple to use, highly automated, flexible, robust, scalable, and efficient.

The implementation is designed to spawn multiple threads of AMS functionality under the control of an AMS application program. These threads enable all members of an AMS-based, distributed application to discover one another in real time, subscribe to messages on specific topics, and to publish messages on specific topics. The query/reply (client/server) communication model is also supported.

Message exchange is optionally subject to encryption (to support confidentiality) and authorization. Fault tolerance measures in the discovery protocol minimize the likelihood of overall application failure due to any single operational error anywhere in the system. The multi-threaded design simplifies processing while enabling application nodes to operate at high speeds; linked lists protected by mutex semaphores and condition variables are used for efficient, inter-thread communication. Applications may use a variety of transport protocols underlying AMS itself, including TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and message queues.

This work was done by Scott C. Burleigh of Santa Barbara Applied Research for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1). NPO-42814

This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-42814.

Zero-Copy Objects System

Zero-Copy Objects System software enables application data to be encapsulated in layers of communication protocol without being copied. Indirect referencing enables application source data, either in memory or in a file, to be encapsulated "in place" within an unlimited number of protocol headers and/or trailers.

Zero-copy objects (ZCOs) are abstract data access representations designed to minimize I/O (input/output) in the encapsulation of application source data within one or more layers of communication protocol structure. They are constructed within the heap space of a "Simple Data Recorder" (SDR) data store to which all participating layers of the stack must have access. Each ZCO contains general information enabling access to the core source data object (an item of application data), together with (a) a linked list of zero or more specific "extents" that reference portions of this source data object, and (b) linked lists of protocol header and trailer capsules. The concatenation of the headers (in ascending stack sequence), the source data object extents, and the trailers (in descending stack sequence) constitute the transmitted data object constructed from the ZCO.

This scheme enables a source data object to be encapsulated in a succession of protocol layers without ever having to be copied from a buffer at one layer of the protocol stack to an encapsulating buffer at a lower layer of the stack. For large source data objects, the savings in copy time and reduction in memory consumption may be considerable.

This work was done by Scott C. Burleigh of ACRO for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-41627.

Delay and Disruption Tolerant Networking MACHETE Model

To verify satisfaction of communication requirements imposed by unique missions, as early as 2000, the Communications Networking Group at the Jet Propulsion Laboratory (JPL) saw the need for an environment to support interplanetary communication protocol design, validation, and characterization. JPL's Multi-mission Advanced Communications Hybrid Environment for Test and Evaluation (MACHETE), described in "Simulator of Space Communication Networks" (NPO-41373) *NASA Tech Briefs*, Vol. 29, No. 8 (August 2005), p. 44, combines various commercial, non-commercial, and in-house custom tools for simulation and performance analysis of space networks. The MACHETE environment supports orbital analysis, link budget analysis, communications network simulations, and hardware-in-the-loop testing. As NASA is expanding its Space Communications and Navigation (SCaN) capabilities to support planned and future missions, building infrastructure to maintain services and developing enabling technologies, an important and broader role is seen for MACHETE in design-phase evaluation of future SCaN architectures.

To support evaluation of the developing Delay Tolerant Networking (DTN) field and its applicability for space networks, JPL developed MACHETE models for DTN – Bundle Protocol (BP) and Licklider/Long-haul Transmission Protocol (LTP). DTN is an Internet Research Task Force (IRTF) architecture providing communication in and/or through highly stressed networking environments such as space exploration and battlefield networks. Stressed networking environments include those