Integrating Collaborative Distributed Simulations for Space Exploration Missions

Esther H. Jennings [1]
Jet Propulsion Laboratory/Caltech, 4800 Oak Grove Drive, Pasadena, CA 91109

Michael G. Blum [2]
Science Applications International Corporation, Moffett Field, CA 94035

Juan M. Busto [3]
John F. Kennedy Space Center, Kennedy Space Center, FL 32899

Victoria Chung [4]
NASA Langley Research Center, Hampton, VA 23681

Edwin Z. Crues [5], Dan Dexter [6]
NASA Johnson Space Center, Houston, TX 77058

David Hasan [7]
L-3 Communications/ Titan Group, Houston, TX 77058, USA

and

Joe Hawkins [8]
Teledyne Brown Engineering, Huntsville, AL 35805

This paper describes the architecture and implementation of a distributed simulation of NASA's Orion spacecraft and Ares launch vehicle in a mission to the International Space Station. The simulation is part of the Integrated Mission Simulation (IMSim) project which aims at research and development collaboration among NASA centers. As mission systems are complex, a distributed simulation allows participating centers to build specific simulation models utilizing specialized technologies at each center. Interactions among the simulation models are defined according to mission operation concepts. Traditionally, simulation of space vehicles has been performed as disjoint collections of simulations. However, the traditional approach does not address integration and interaction among the separate simulations. To simulate a system of systems, distributed simulation technologies provide promising solutions. Evaluations from a previous middleware trade study showed that the IEEE 1516 High Level Architecture (HLA) standard is a suitable candidate for IMSim. In this paper, we describe the use of HLA for the Orion to ISS mission. As of July 2008, there are up to eight federates participating in a distributed simulation. They are: Mobile Launcher (ML), Launch Control System (LCS), Launch Abort System (LAS), Orion

[1] Technical Staff, Communications Networks Group, 4800 Oak Grove Drive M/S 238-343, Pasadena, CA 91109
[2] Principal Engineer, Space Simulation Development, Science Applications International Corporation, Mail Stop 243-6, AIAA Member.
[3] Project Engineer, IT-C2.
[4] Assistant Head, Simulation Development and Analysis Branch, Mail Stop 125B, AIAA Senior Member.
[5] Architecture Lead, Modeling and Simulation Labs, 2101 NASA Parkway, ER7, NASA Johnson Space Center, Houston, Texas 77058, AIAA Senior Member.
[6] Aerospace Technologist, Simulation and Graphics Branch, 2101 NASA Parkway, ER7, NASA Johnson Space Center, Houston, Texas 77058.
[7] Engineer, L3 Communications, Titan Group, 1002 Gemini, Suite 200, Houston, TX 77058.
[8] Senior Systems Analyst, Teledyne Brown Engineering, 300 Sparkman Drive, M/S 134, AL 35807.

(Crew Exploration Vehicle), Ares (Crew Launch Vehicle), International Space Station (ISS), the Tracking and Data Relay Satellite System (TDRSS), and Space Communications and Navigation Network (SCaN). Milestone simulation runs confirmed the expected behavior of the system. Issues, concerns and future plans are also discussed.

I. Introduction

One of the Vision for Space Exploration's prime objectives is to continue the exploration of space and specifically return humans to the moon and continue on to Mars. This Vision is being realized through NASA's Exploration Initiative and this is the responsibility of NASA's Exploration Systems Mission Directorate (ESMD). This involves the technical expertise and active participation of scientists and engineers from all ten of NASA's research and flight facilities. As actual mission systems are expensive to build, it is essential that NASA develop a low cost method to validate and verify mission designs and operations. This can be accomplished through Simulation.

Simulation can aid design decisions where alternative solutions are being considered, support trade-studies and enable fast study of what-if scenarios. It can be used to identify risks, verify system performance against requirements, and as an initial test environment as one moves towards emulation and actual hardware implementation of the systems. In a system of systems, while each system can be simulated in isolation, the integrated simulation enables added fidelity by also modeling the interactions and inter-dependencies between the systems. Since simulation is an integral part of fulfilling the Vision, and development activities are geographically distributed across the United States, it was judged worthwhile to attempt integration of the separated and distinct simulation components available at different NASA centers, into one overall simulation without relocating the components; In other words, form a 'distributed simulation'. The Integrated Mission Simulation (IMSim) project, a multi-center product research and development project, has been formed to address challenges of developing effective and efficient simulations in this distributed environment.

Like many NASA projects, IMSim lives within an organizational hierarchy. Specifically, the IMSim project is run from the Modeling and Simulation Labs (M&SL) out of the Software Avionics Integration Office (SAVIO). SAVIO in turn supports the Systems Engineering and Integration (SE&I) group for NASA's Constellation Program (CxP). The Constellation Program is the organization responsible for implementing the vision of the Exploration Systems Mission Directorate (ESMD), which leads NASA's new Vision for Space Exploration initiative. The Constellation Program is responsible for building and operating NASA's new space vehicles that will return humans to the moon and then eventually enable further exploration to Mars.

The IMSim project goal is to research, develop and deploy technologies, processes and simulations which support the collaborative, interoperable and distributed simulation of complex space systems in support of NASA's Exploration Initiative. The objective is to provide NASA with the ability to simulate systems of systems that are modular, scalable, re-configurable, and extensible. This will be met through research and development. Specifically, the IMSim team researches long lead time simulation technologies that can be applied to Constellation program simulation projects. The team also develops simulation infrastructure to support the development and deployment of distributed simulation solutions throughout the Constellation program. This provides the Constellation program with support for integration of high fidelity, time based simulations of all Constellation program vehicle elements.

II. IMSim History

Like most projects, the IMSim project did not emerge fully formed in either scope or participation. A number of NASA centers have been investigating distributed simulation technologies. One precursor to the IMSim project was the DIstributed Simulation (DIS) project. This project's purpose is to build a distributed simulation for use in flight procedures development and training of the HII-A Transfer Vehicle (HTV) and its operations in proximity of the International Space Station (ISS). This simulation is referred to as the HTV Flight Controller Trainer (FCT) simulation. This is a collaborative effort between the Japanese Aerospace eXploration Agency (JAXA) and the NASA Johnson Space Center (JSC) Mission Operations Directorate. In this simulation, the HTV components run at the JAXA facilities in Tsukuba, Japan and the ISS components run at JSC in Houston, Texas. A graphic scene generated from this simulation is shown below in Figure 1. Much of the knowledge and technologies used in the HTV FCT have been directly applied to the Distributed Space Exploration Simulation (DSES) project. The DSES project is the direct precursor to IMSim.

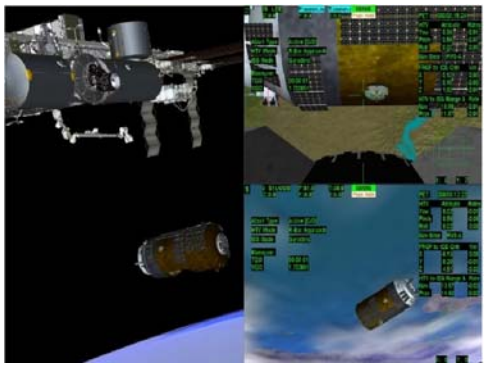

Figure 1. The HTV FCT Simulation

Initially, the DSES project was referred to as a "coalition of the willing" in the sense that small groups of NASA scientists and engineers with similar interests and little funding met on a regular basis to exchange information and ideas on new approaches to simulation. These groups located at NASA Johnson Space Center (JSC), NASA Ames Research Center (ARC) and NASA Langley Research Center (LaRC) configured a small informal network of computers and began testing out some simple distributed simulations. As the DSES work progressed, interest began to spread to other centers, including NASA's Marshall Space Flight Center (MSFC), Kennedy Space Center (KSC), Glenn Research Center (GRC), Goddard Space Flight Center (GSFC) and the Jet Propulsion Laboratory (JPL). In October of 2007, the DSES project was renamed the Integrated Mission Simulation (IMSim) project.

II. Development Areas

To date, the IMSim project has focused on three principal research and development areas: Network Infrastructure, Software Infrastructure, and Simulation Development. The following sections provide a high level description of the three principal work areas for IMSim. The details of these areas are covered in separate papers. [1, 2]

A. Network Infrastructure

The IMSim network infrastructure is currently based on an interconnection of facility local area networks through the NASA Integrated Services Network (NISN). The NISN backbone along with a combination of externally accessible IP addresses and access allocations through the facility network firewalls forms the NASA Distributed Simulation Network (DSNet). While the current network has limited levels of service guarantees, the determination of the form and necessity of network guarantees is an area of IMSim investigation. Figure 2 shows the eight currently active IMSim NASA centers connected through the DSNet: ARC, GSFC, GRC, JPL, JSC, KSC, LaRC and MSFC. There are plans to connect the remaining two NASA center as the project continues.

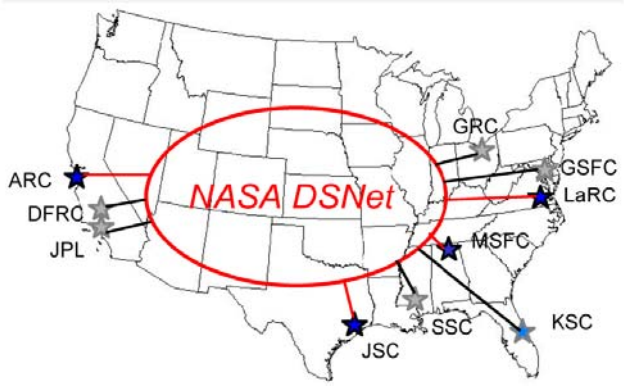

Figure 2. Distributed Simulation Network

While the DSNet does provide the necessary connectivity for distributed simulation, pure connectivity is not sufficient; it is also important to understand both the status and performance of the network. To gain insight into

both the current status and performance of the network connections, the IMSim project is creating a collection of DSNet status and performance tools.

B. Software Infrastructure

The IMSim software infrastructure consists of identifiable collections of software assets located at each of the participating NASA facilities. These software assets include a common distributed simulation software framework. For the initial phases of the IMSim project, this framework is the High Level Architecture (HLA). However, other distributed simulation software infrastructures are being investigated. Each NASA facility provides simulation components for the distributed simulation using in-house simulation software and systems but rely on the common distribution framework for external interfaces, timing and execution control.

In a distributed simulation setting, an association of possibly distributed processes cooperating using HLA is called a federation where each process participating in the federation is a federate. A federation is usually created by the first process (federate) accomplishing the dynamic registration process (ie. 'joining') the federation and destroyed by the last process leaving the federation. Federates exchange information through a publish/subscribe mechanism to update object attributes. Interactions among federates are realized through sending and receiving interaction events. A federation has a specific federation object model (FOM) that defines the structure of the types of objects and interactions that may be exchanged by participating federates. A runtime infrastructure coordinates the processes so that federates in a federation can be time-synchronized.

IMSim project simulations are currently HLA based; they depend on a common description of interoperable data types that are defined in the IMSim FOM. The FOM currently consists of a simple set of objects that are used to exchange space vehicle state information like position, velocity and acceleration. The IMSim FOM also defines a collection of interactions or message definitions that are used to exchange vehicle command and status information.

Since many of the simulations being developed for the Constellation Program are built in a common simulation development environment called Trick, the IMSim project is also creating a generalized set of HLA classes. These Trick HLA classes take advantage of some of Trick's generalized I/O and variable access capabilities to provide a model that can be included into almost any Trick based simulation to allow it to join into almost any IMSim Federation. The mapping of FOM objects, attributes, interactions and parameters are defined in data at run time. In this way, many of the details and intricacies of HLA development will be hidden from the developers.

C. Simulation Development

The essential elements for the successful development and deployment of interoperable CxP mission segment simulations include a combination of IMSim network infrastructure and software solutions. The IMSim project is using an evolutionary development process. In this process the various models and systems required for a working distributed simulation of a space exploration vehicle are developed in phases. The early phases have concentrated on technology demonstrations and simulation architecture development. The next phases will concentrate on modeling capabilities, data exchange, model exchange and coordinated execution. The final phases will concentrate on integrated system execution and analysis products. The Orion and Ares I Launch and Ascent simulation is the first simulation product from this IMSim simulation development activity. As of April 2008, we have further developed our simulation to include launch ascent, first and second stage separations of Ares, orbit insertion of Orion, rendezvous and docking with the International Space Station (ISS).

III. Demonstration system for Constellation's Orion to ISS Mission

In the following, we describe network monitoring tools and system component simulation software for the distributed simulation of Constellation's Orion to ISS mission.

A. Network Connectivity and Automation

This Section describes the distributed simulation effort undertaken to establish and verify a basic and reliable set of the specific connections needed between applications. Key facets are the internet, IP permissions, connections and settings, and automation of distributed communications services at a high level, excluding the HLA middleware, described in Section B.

1) Early Infrastructure Tools

Testing between the initial two and three NASA centers on DSNet revealed that although a special ARC-JSC link worked well, the performance over the Internet was insufficient to enable a reliable simulation at frame rates below 1 Hz. There were packet loss and jitter issues which were discovered by ARC through the use of *iperf*, a common open source network diagnostics tool readily available on the Linux Operating System suggested and adopted for distributed NASA simulations.

In order to provide timely feedback, for DSNet users and infrastructure support staff, concerning needed corrective action, dozens of person hours were scheduled each week until the ARC *Autoperf* became available and replaced that practice in 2006. This tool included browser-based tools written by ARC and MSFC to automate the collection and presentation (including charts and plots) of *Autoperf*'s jitter and packet loss information on all six links between the four NASA centers involved.

Autoperf gathers network latency, bandwidth and packet loss data and passes it to a central server where results are parsed, averaged, and archived for viewing through a web browser. This has saved hundreds of hours of manual tests, and was necessary to facilitate discovery of problems and to enable network improvement to current levels of performance.

Continued additions to the DSNet, and major interest in efficiently checking network and node performance led to the development of another toolkit, which would have a smaller footprint in terms of both size and network bandwidth use. As of March 2008, there are 65 nodes with over 2000 links to test. It became apparent, as the numbers of nodes and links grew exponentially, that something was needed which does not require a re-plan of its test suite when additional nodes are added, and which can also recover faster from socket connectivity problems.

2) Tool Specification Matures

The automation phase resulted in more mature requirements in the form of additional elements IMSim required, beyond just the distributed simulation models and specifically oriented data transfer executives, in order to achieve flexible and efficient distributed simulations through a scalable network.

The first requirement was the creation of one common program which reads a single common input file edited in seconds to reflect additional nodes. This is simpler, and less labor-intensive than manual or individualized planning and initialization scripts, and hence less prone to error than is the Autoperf suite. A plain text configuration file makes documentation easier, which paves the way for automatic and centralized configuration management for each planned simulation. This also allows simulation users to collect their results easily since that capability and others related to it, are afforded by the same features needed to run network diagnostics effectively, making simulation assistance services available as free 'piggyback' features.

The second requirement stems from HLA benchmark work. We were in the process of porting the four HLA Benchmarks to gcc Linux in order to compare 1516 HLA sim executives with other types, so it was desired to collect packet loss and latency information from a small footprint executive, to provide comparison data with a communications executive type with minimal (near zero) overhead.

Lastly, for the third requirement, we wanted a Real Time watchdog mode, so that different types of simulation programmers and support people can monitor activities on their machines, their IT department, and remote ones.

3) SODA Tool

Clearly, the requirements were calling for something original and unique. As a result, one of the ARC crew federate interface components was expanded for experimentation. The PC platform, Linux OS, and gnu *toolchain* started performing well enough in early development and testing, to build and run the key software agent, referred to as the Self-Organizing Distributed simulation Assistant (SODA). On the surface, *SODA* is a direct transparent command line interface between nodes or applications. A user, application, or script can easily submit a message for automatic delivery to the receiving application. For easy automation, a command line allows automation and encapsulation *ad infinitum*, permitting others to easily customize agent use for progressively larger automated endeavors.

The most commonly used feature to date is the internal radar and transponder package which allows each agent to relate to all other agents every few seconds if desired. Through encrypted file transfers, semi-automatic updates, message passing, consoles, a graphical viewer which allows users to rapidly test different frame rates, and the ability to call up or schedule remote simulation components where permissions are given, applications are specified, and a full SODA agent is installed, a minimal amount of human time is now required to set up, verify, and run the simulation. This leaves people free to participate in whatever manual or human roles there are to play.

During IMSim simulations in July 2008, SODA determined that some periods of markedly slow simulation time with respect to real time were due not to network performance issues, but rather to host load issues. A “real-time-ometer” is part of the ARC crew federate, which is logged along with other simulation artifacts pertaining to vehicle trajectory and crew input. Since *SODA* runs without adding significant network and host loading, there is always a

realtime watchdog available to capture and identify potential problems, in the hosts or in the network that might degrade the simulation results.

B. TrickHLA

Over the years, JSC has developed and used a number of simulation tools. Recently, divisions in the Engineering Directorate at JSC have been adopting a common simulation development tool: "The Trick Simulation Development Environment." [3-5] Trick automates many of the tedious processes in constructing simulations and provides a number of generalized simulation capabilities. In addition, JSC has been assembling a collection of common models. These models, while hosted in the Trick environment, are not necessarily Trick dependent. The goal is to develop a suite of space systems models that can be shared across projects and facilities.

IMSim project has also developed a generalized TrickHLA model which sits between Trick and HLA; it consists of simulation objects (data) and jobs (functions). The TrickHLA model provides an abstraction of HLA in the Trick simulation environment allowing a developer to focus on simulation development without needing extensive knowledge of HLA. Additional information on TrickHLA can be found in the TrickHLA User Guide.[14]

C. Orion Crew Exploration Vehicle Federate

JSC IMSim simulations are built using Trick and its collection of common models. One of these is the Advanced NASA Technology Architecture for Exploration Studies (ANTARES), a multi-purpose simulation supporting the Orion Guidance Navigation and Control (GN&C) team. ANTARES supports a wide variety of analyses, including requirements assessments, design trades, GN&C flight software evaluation and test, and pilot-in-the-loop interactions. ANTARES also supports computational and parametric studies as well as hardware and crew-in-the-loop facilities. ANTARES is based on common model library architecture, leveraging off simulation development across several organizations at JSC. It forms the core of the JSC contribution to IMSim Orion simulations, such as the Launch and Ascent simulations and the Orion/ISS rendezvous and docking simulation.

D. Crew Launch Vehicle (CLV) Federate

The Crew Launch Vehicle (CLV) federate, the Ares I element, is represented in IMSim by the MSFC developed simulation the Ares Real-Time Environment for Modeling, Integration, and Simulation (ARTEMIS).

ARTEMIS is being developed by MSFC for use in the CLV System Integration Lab (SIL) hardware-in-the-loop test bed to test out Ares I avionics. It is designed to model the Ares I from pre-launch through orbit insertion. There are multiple configurations of ARTEMIS – a non real-time, single platform version and a real-time, distributed version that eventually will interface with avionics hardware components. In the current IMSim federation configuration, ARTEMIS is executed in the single platform configuration. The addition of ARTEMIS to the IMSim configuration provides a ready pathway to allow the IMSim configuration to evolve with higher fidelity Ares I configuration representations as the Ares SIL matures.

ARTEMIS was modified for IMSim by adding TrickHLA and objects to represent the HLA version of objects (space vehicles, simulation control, and launch control system). The code was modified to accept the mission start time from the launch control system. Changes were also made to support an abort command.

E. Mobile Launcher (ML) Federate

The simulation modeling language currently in use at KSC is known as the Shuttle Ground Operations Simulation (SGOS). SGOS was developed in-house and has been successfully used throughout the life of the Shuttle program for simulating the Ground Support Equipment (GSE) to vehicle interfaces. It provides the capability to perform launch team training as well as application software verification and checkout.

For the CxP Launch Control System (LCS) project KSC will be developing new GSE models utilizing Simulink, a Commercial Off-The-Shelf (COTS) development tool. Simulink, a product of MathWorks, provides a graphical development environment for creating models. The simulation execution engine selected to run these models is the Trick Simulation Development Environment. Once the models have been created under Simulink the output generated from the Simulink tool will be a Trick based source which can then be compiled and executed under the Trick execution engine.

The Mobile Launcher (ML) simulation federate, although a Trick based model, has not followed this simulation model design process. It was written entirely in Trick and TrickHLA and provides positional information to the vehicle while it sits on the pad prior to T-0. It essentially acts as a launch vehicle "Stage 0". When the count reaches T-0 the ML simulation notifies the other federates that the hold-down bolts have been released. Although this simulation currently provides basic functionality it has served as a pathfinder for not only utilizing Trick and TrickHLA but in understanding the infrastructure required for supporting distributed simulations.

F. Launch Control System (LCS) Federate

The Launch Control System (LCS) simulation federate was also not developed utilizing the new LCS Simulation development design process. It was also written entirely in Trick and TrickHLA for the IMSim project. This simulation provides the capability to run various launch countdown scenarios which include a nominal countdown or an extended/contingency hold. A GUI, written in TCL, provides up-to-date countdown status. The display includes a universal time clock, a countdown timer, a launch window close time, a projected launch time, and a countdown status flag.

G. Launch Abort System (LAS) Federate

The Launch Abort System (LAS) simulation of the IMSim project in October 2007 was developed by the Simulation Development and Analysis Branch (SDAB) at the NASA Langley Research Center. The SDAB established the Langley Standard Real-Time Simulation in C++ (LaSRS++) framework in 1995. LaSRS++ is an object-oriented application framework for constructing continuous cyclic simulations. The framework was built from scratch using modern object-oriented programming and design techniques. LaSRS++ consists of a suite of software libraries that can be shared among developers, projects, and facilities. The framework provides all of the critical services required to perform simulations, allowing developers to focus only on vehicle model development. The benefits of using the common framework for simulation development include the increase in productivity and ease of maintenance due to a high degree of framework code reuse, cohesive software practices, and common coding standards for project software development. The framework supports multiple, heterogeneous models in a simulation. While originally developed to support aircraft simulation, models are not restricted to aircraft and can represent any interactive item. Vehicle models are portable across simulators with little to no modifications to its "hardware interface" code. LaSRS++ was designed to be platform non-specific, and has been used on IRIX, Linux, Solaris, and Windows operating systems. Descriptions of the LaSRS++ framework and application software are detailed in references. [9-13]

The LaSRS++ framework includes various world models such as Earth, Moon, and Mars for use in simulations. Several atmospheric models are also included in the framework such as the Global Reference Atmosphere Model (GRAM), Mars Global Reference Atmosphere Model (Mars-GRAM), and Marshall Engineering Thermosphere (MET). The framework supports real-time human-in-the-loop, hardware-in-the-loop, and parametric simulation and analysis. It also supports multiple simulation aspects such as GNC, vehicle dynamics and behavior, trajectory analysis, vehicle performance assessment, crew displays, and flight deck avionics and layout for part task or end-to-end full mission simulation. The LaRC's HLA interface software is being designed and developed to be independent of LaSRS++, making it available for potential reuse within other simulation architectures.

The LaRC Launch Abort System (LAS) simulation for the IMSim project was derived from the generic spacecraft model of the LaSRS++ framework. The generic spacecraft model added enhancements to the LaSRS++ framework including staging and orbital mechanics calculation. Generic spacecraft models were composed of interconnected sub-models such as mass, propulsion, GNC, and aerodynamics. The LAS simulation included force and mass models that provided simulation of the escape, pitch, and jettison motors.

H. TDRSS

Any model of space-to-space and space-to-ground communications links must know the positions of the endpoints in order to calculate light-time delays and simulate link errors, among other things. Early versions of the communications and tracking simulation used tabulated TDRSS satellite positions from an input file.

To get around the limitation of pre-computed tabulated data, IMSim developed a Trick-based simulation of a five-satellite TDRSS constellation (although Constellation Program might only use two of these). The simulation is based on the Johnson Space Center Engineering Orbital Dynamics (JEOD) model. It is a six degree of freedom simulation, modeling the orbit state as well at the attitude of each of five TDRSS vehicles. The positions and velocities of the vehicles are published via HLA (where they are available to the subscribing communications and tracking federate). In the future, the attitude data in this model might be used to calculate antenna pointing parameters.

I. International Space Station

The IMSim International Space Station (ISS) simulations is also a Trick based simulation built with the same basic common simulation elements as the CEV and TDRSS simulations. However, this simulation was not built for the Constellation Program. It was built to support the ISS program. Specifically, this simulation was built to support ISS robotics analysis. This particular variation has been built to support the rendezvous and proximity operations of the Japanese HII transfer Vehicle (HTV) with the ISS. It contains a high fidelity model of the ISS attitude control system and Space Station Robotic Manipulator Systems (SSRMS). It is used in training the distributed flight controller teams of NASA and the Japanese Aerospace eXploration Agency (JAXA). With some

simple modifications, it is now being used in IMSim to support Orion operations with the ISS. This highlights the reuse advantages that come from the flexible simulation interoperability framework employed in IMSim.

J. Space Communications and Navigation (SCaN) Network Federate

Space Exploration missions will involve numerous human-capable and robotic elements, requiring reliable and high-bandwidth communications with cost-effective networking. Traditionally, telecommunications design and technology development processes have used analytical methods to ensure mission success by providing significant resource margin to mitigate the impact of design error. However, future space exploration missions will experience exponential increase in complexity due to the large number of system elements. Traditional performance prediction and evaluation methods must be supplemented with detailed simulation to capture the high-order dynamics of complex space systems and networks. Accurate characterization of performance and system behavior is required to make sound technology development and engineering decisions. Therefore, JPL and GRC are developing a SCaN Network simulator tool under System Concepts Integration and Planning (SCIP) Project in order to assist in analysis of current and proposed network architectures. At the core of the SCaN simulator is the QualNet network simulator, a commercial product by Scalable Network Technologies (SNT). In addition to the QualNet discrete-event simulation engine, the SCaN simulator includes packages and add-on modules developed at JPL and GRC to model specific SCaN entities or protocols, not supported by the commercial package from SNT. SCaN may also utilize the models developed for the Multi-mission Advanced Communications Hybrid Environment for Test and Evaluation (MACHETE) tool suite.[6]

The SCaN Network architecture[7] for Orion mission will include the Space Network (SN), the Ground Network (GN), and will use the services provided by NASA Integrated Services Network (NISN). The space segment of the SN element consists of multiple operational Tracking Data Relay Satellites (TDRSS) in geosynchronous orbit at allocated longitudes for relaying forward and return service signals to and from customers for data transfer and tracking. The GN sites primarily support S-band communication links. Some GN stations can provide radiometric range and Doppler measurements of space vehicles from the S-band RF links. Some GN ground stations also provide antenna angle tracking data when S-band autotrack is used. The White Sands Complex (WSC) provides the communications equipment necessary for transmitting and receiving data and tracking information relayed via each TDRS. WSC includes three ground terminals which are: the White Sands Ground Terminal (WSGT), the Second TDRS Ground Terminal (STGT), and the Guam Remote Ground Terminal (GRGT). WSC controls the GRGT remotely because of its location. The NASA Integrated Services Network (NISN) provides wide area network (WAN) telecommunications services for the transmission of terrestrial data, voice and video between all SCaN Network ground elements and Constellation/user ground elements.

The main function of the SCaN federate is to simulate communications effects of the SCaN network. This federate obtains spacecraft status and position information through HLA infrastructure. SCaN receives TDRS positions through HLA with respect to time. It also calculates the distances between spacecraft and TDRS, and between TDRS to ground stations with respect to time. Using spacecraft positions and specifications of the link type (e.g. S-band, data rate, bandwidth), the tool computes bit error rates (BER). A link budget library was built to simulate the communications effects and the tool is capable of delaying data to simulate propagation delay effects. Actual data from a spacecraft can be piped to the SCaN simulator through a socket; the simulator will then relay the data to the intended destination. If actual data is not available, the tool can generate data traffic according to specific traffic profiles to model internal traffic flows. Data will be dropped according to the computed BER; latency is also imposed according to the computed propagation delay. A representative topology of the SCaN network for the Orion mission is shown in Figure 3; it is an actual screen capture during simulation run. In Figure 3, CEV-socket, CLV-socket, and MCC-socket denote the sockets connecting to external data sources (e.g. spacecraft, mission control center). For each external data source (or destination), we need to define a corresponding virtual node in the simulator (e.g. CEV, CLV, MCC). The SCaN network consists of one TDRS and WSC ground station. In the future, we will add multiple TDRSs and simulate handover among the satellites.

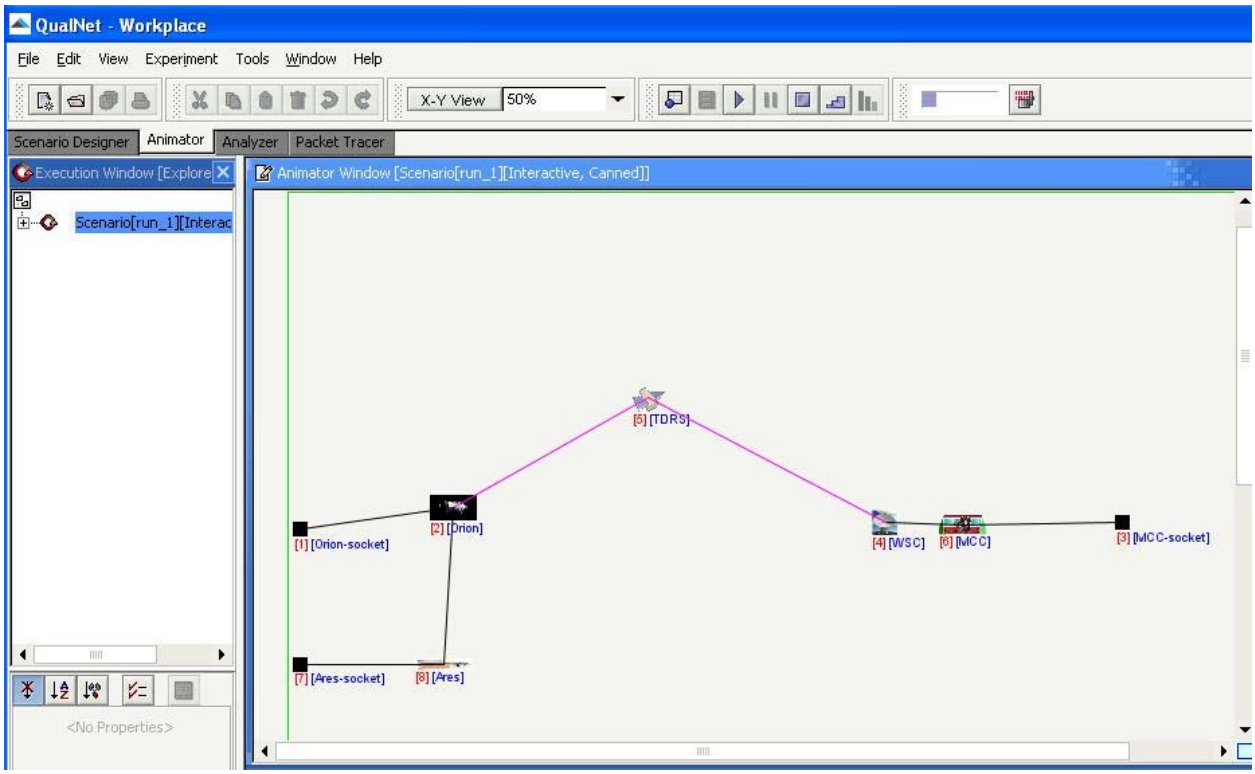

Figure 3. Orion mission network topology

As an initial experiment of IP-based space networking, we have set up our scenario to examine data transmission via an architecture using satellite bent-pipe (Orion through TDRS to WSC ground station). Figure 4 includes a path in the scenario and a generic representation of the protocol stack used at each node (system element). Physical layer effects (e.g. RF signal strength, bit error, path loss) are simulated through calls to subroutines in a link budget library which was built according to Constellation's Master Link Book. The data link layer (DL) consists of CCSDS Encapsulation service and CCSDS Advanced Orbiting Systems (AOS) Data Link protocol[8] over Low-density Parity-check (LDPC) model. IP and UDP were the protocols used at the network and transport layers respectively. As an example, Orion may send telemetry data to Earth (MCC) using Data Exchange (DE) protocol (Application layer). This is then transported using UDP (transport layer) and IP (network layer). The data units would then be encapsulated to flow over the CCSDS AOS protocol[8] between Orion and WSC via TDRS; link budget library is called to compute bit error rate and propagation delay. From WSC, the data are transferred to the appropriate mission control center via NISN.

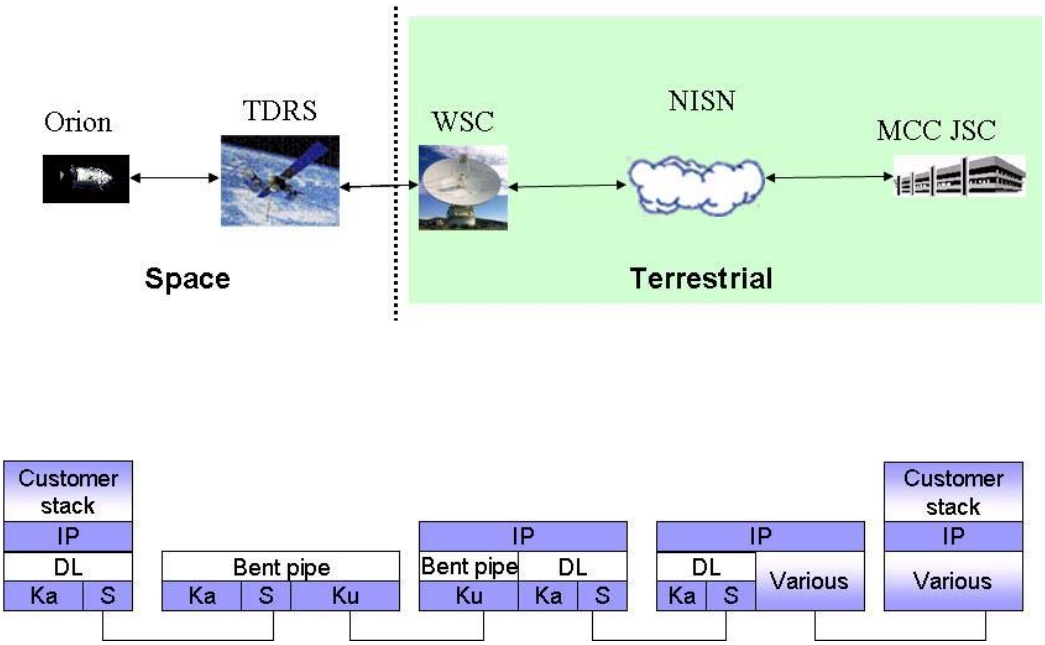

Figure 4. Orion mission protocol stack

IV. Simulation Scenarios and Outcomes

IMSim focuses on incremental development of infrastructure, expertise and technical capabilities. Early IMSim work concentrated on validation of the distributed simulation concept and the deployment of an initial Orion/Ares simulation. Later IMSim milestones concentrated on more complex Constellation mission scenarios that involve more federates.

A. Early work

IMSim work in 2005 consisted mainly of developing the distributed simulation concept and building the necessary network and software infrastructure.

It was not until 2006 that actual simulations were developed. In the spring of 2006, a simple Constellation stack simulation was built that included a MAVERIC-based CLV, an IMSim-developed CEV, a crew federate (for crew initiated abort commands) and a Launch Abort System. These were integrated into an ascent simulation in the summer. In the winter, a separate on-orbit CEV/ISS rendezvous/docking capability was developed using an ANTARES-based CEV federate.

By the end of 2006, IMSim had a general capability to design and deploy HLA-based distributed simulations using expertise and infrastructure that had been developed during 2005 and 2006. Although much model development remained to be done, the basic distributed simulation machinery was in place.

K. Milestones in 2007

In mid-2007, IMSim deployed two simulations: (a) an Orion/Ares simulation from pre-launch to orbit insertion and (b) an on-orbit CEV/ISS rendezvous and docking simulation. In the **ascent simulation**, ARTEMIS CLV models were developed, and the ML and LCS federates were added to the federation, allowing users to simulate a simple countdown. In the **on-orbit simulation**, an ANTARES-based CEV replaced the IMSim-developed model used in the ascent simulation.

In late 2007, IMSim deployed three simulations: (a) Orion/Ares launch to orbit insertion, including an ANTARES-based CEV, (b) on-orbit CEV/ISS rendezvous and docking and (c) an Orion crew module entry, descent and landing based on ANTARES CEV models. The **ascent simulation** consisted of the ML, LCS, CLV (ARTEMIS-based), CEV (ANTARES-based), LAS and a crew monitoring federate. The **on-orbit simulation** consisted of CEV (ANTARES-based) and an IMSim-developed ISS. The **entry simulation** consisted only of the ANTARES-based CEV.

By the end of 2007, IMSim had deployed a number of additional federates and had three different simulations that covered mission scenarios from pre-launch through ISS docking and CEV landing. Although these three

scenarios were simulated separately, the models necessary to support an end-to-end mission scenario were mostly in place and the remaining integration challenges were identified.

L. Milestones in 2008

The most recent IMSim milestone (April 2008) involved an integrated Orion/Ares ascent to orbit, including some pre-launch events, Ares stage one and two separation, orbit insertion, and finally rendezvous and docking with the International Space Station. It consisted of ML, LCS, CLV, CEV, ISS, SCaN and TDRSS federates.

This mission scenario did not involve an abort, so the crew monitoring federate was not strictly necessary, although it did participate in the federation. In addition, the LAS element of this simulation was embedded into the CEV federate rather than implemented as a stand-alone federation. (This was done both for technical and programmatic reasons: the CEV/LAS separation dynamics can be quite complex, involving extended force and moment interactions that make distributed simulation challenging, and funding for the LAS team was reduced by NASA.) The non-critical TDRSS and SCaN federates were integrated into the simulation as part of the IMSim philosophy of incremental deployment.

By the middle of 2008, IMSim had made progress towards an end-to-end (pre-launch-to-landing) scenario, with the ascent and on-orbit phases successfully merged into a single simulation. In addition, the SCaN and TDRSS federates were integrated into the federation in anticipation of the role they will plan in upcoming 2008 milestones in which simulated vehicle telemetry will be relayed through the SCaN federation that will model space-to-space data links and errors using positions generated by the TDRSS federate.

Findings during these simulations included bugs in the HLA software and peripherals drivers (eg. hand controllers), and pointers to additions or improvements to some simulation software management processes.

V. Conclusion

IMSim has demonstrated that a small dispersed team can integrate a large portion of a Constellation simulation across NASA centers, and that interfaces can then be tested at a sufficient fidelity to explore interoperability issues and other bugs at an early stage when the price to fix is minimal. Work will continue towards the goal of extending the IMSim capabilities to include a complete end-to-end mission scenario. This multi-center collaboration has led to establishment of agency wide capabilities and assets that include network and software. The integrated simulation demonstrates how simulation interoperability can be achieved across the Constellation program. The underlying payoff in development efficiency and cost effectiveness in integrated mission simulation is sought through leveraging NASA's dispersed simulation domain expertise and resources throughout the agency and cultivating a framework for simulation and model interoperability and reuse. The derived advantages provide for a more efficient and effective path to simulation verification, validation, and accreditation that ultimately increases confidence in simulation results and fidelity.

As the Constellation Program advances, new mission segments will be added. This includes missions to return humans to the Moon and our planetary neighbor Mars. As these and other mission segments evolve, integrated simulations can be extended to support the end-to-end test and verification of the Constellation Program mission profile.

Acknowledgements

The authors would like to extend our profuse appreciation to the other engineers and developers on the team who have contributed to the IMSim project: Paul Bielski, Dan Bowman, Mark Coffman, Chris Daum, Jim Gibson, Missy Hill, Sean Kenney, Michael Madden, Jason Neuhaus, Marty Pethtel, Al Ragsdale, Mark Ricci, Russ Sexton, Paul Sugden, and Jesse Trammell. Part of the work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. These engineers and developers were necessary to make the IMSim successful. Their contribution to the IMSim project showcased the proven fact of viable collaborative distributed simulations for space exploration missions

References

[1]Crues, E., Blum, M., Chung, V., and Bowman, J., "The Distributed Space Exploration Simulation (DSES)," Proceedings of the Spring 2007 Simulation Interoperability Workshop and Conference, Norfolk, Virginia, 25-30 March 2007.

[2]Phillips, R., Dexter, D., and Crues, E., "A Coordinated Initialization Process for the Distributed Space

Exploration Simulation (DSES)," Proceedings of the Spring 2007 Simulation Interoperability Workshop and Conference, Norfolk, Virginia, 25-30 March 2007.

[3]Vetter, K., and Hua, G., "The Trick User's Guide 2005.7.0 Release," NASA Technical Publication, NASA Johnson Space Center, June 2006.

[4]Vetter, K., "Trick Simulation Environment – User Training Material 2005.7.0 Release," NASA Technical Publication, NASA Johnson Space Center, June 2006.

[5]Paddock, E. J., Lin, A., Vetter, K., and Crues, E. Z., "Trick: A Simulation Development Toolkit," AIAA Modeling and Simulation Technologies Conference and Exhibit, Austin, TX, 2003, AIAA 2003-5809.

[6]Gao, J., Jennings, E., Clare, L., Segui, J., Kwong, W., "MACHETE: A Tool for Architectural Modeling, Performance Characterization, and Technology Infusion of Space-Based Networks", AIAA 2005 International Communications Satellite Systems Conference (ICSSC), Sept 2005.

[7]Bhasin, K., Hudiburg, J., Miller, R., "Space Communications and Navigation Constellation Integration Project – Architecture Design Document: ORION-ISS Mission Phase" (Draft), March 2007.

[8]CCSDS 732.0-B-1, "AOS Space Data Link Protocol", Blue Book, Issue 1, September 2003.

[9]Neuhaus, J. "Modeling Mass Properties in an Object-Oriented Simulation," AIAA Modeling and Simulation Technologies Conference, Providence, Rhode Island, August 16-19, 2004, AIAA 2004-5166.

[10]Madden, M. M. "Examining Reuse in LaSRS++-Based Projects", AIAA Modeling and Simulation Technologies Conference, Montreal, Canada, Aug. 6-9, 2001, AIAA-2001-4119.

[11]Chung, V. I., and Hutchinson, B. K.,"A Unique Software System For Simulation-To-Flight Research," AIAA Modeling and Simulation Technologies Conference & Exhibit, Montreal, Canada, August 6-9, 2001, AIAA 2001-4057.

[12]Madden, M. M.; Glaab, P. C.; Cunningham, K.; Leslie, R. A.; Kenney, P. S.; and Geyer, D. W.: "Constructing a Multiple-Vehicle, Multiple-CPU Simulation Using Object-Oriented," AIAA Modeling and Simulation Technologies Conference, Boston, Massachusetts, Aug. 9-11, 1998, AIAA-98-4530.

[13]Leslie, R. A.; Geyer, D. W.; Cunningham, K.; Glaab, P. C.; Kenney, P. S.; and Madden, M. M.: "LaSRS++ - An Object-Oriented Framework for Real-Time Simulation of Aircraft," AIAA Modeling and Simulation Technologies Conference, Boston, Massachusetts, Aug. 9-11, 1998, AIAA-98-4529.

[14]Hasan, David A.: "Trick High Level Architecture User Guide", revision 1.0, Simulation and Graphics Branch, Automation, Robotics and Simulation Division Engineering Directorate, NASA Johnson Space Center, Houston, Texas, August 2007.

[15]Consultative Committee for Space Data Systems (CCSDS), "Encapsulation Service", Recommended Standard, CCSDS 133.1-B-1, June 2006 (with Technical Corrigendum dated October 2006).