

# Interfacing Space Communications and Navigation Network Simulation with Distributed System Integration Laboratories (DSIL)

Esther H. Jennings<sup>\*</sup>, Sam p. Nguyen<sup>†</sup>, Shin-Ywan Wang<sup>‡</sup>, and Simon S. Woo<sup>§</sup>  
*Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109*

NASA's planned Lunar missions will involve multiple NASA centers where each participating center has a specific role and specialization. In this vision, the Constellation program (CxP)'s Distributed System Integration Laboratories (DSIL) architecture consist of multiple System Integration Labs (SILs), with simulators, emulators, testlabs and control centers interacting with each other over a broadband network to perform test and verification for mission scenarios. To support the end-to-end simulation and emulation effort of NASA' exploration initiatives, different NASA centers are interconnected to participate in distributed simulations. Currently, DSIL has interconnections among the following NASA centers: Johnson Space Center (JSC), Kennedy Space Center (KSC), Marshall Space Flight Center (MSFC) and Jet Propulsion Laboratory (JPL). Through interconnections and interactions among different NASA centers, critical resources and data can be shared, while independent simulations can be performed simultaneously at different NASA locations, to effectively utilize the simulation and emulation capabilities at each center. Furthermore, the development of DSIL can maximally leverage the existing project simulation and testing plans. In this work, we describe the specific role and development activities at JPL for Space Communications and Navigation Network (SCaN) simulator using the Multi-mission Advanced Communications Hybrid Environment for Test and Evaluation (MACHETE) tool to simulate communications effects among mission assets. Using MACHETE, different space network configurations among spacecrafts and ground systems of various parameter sets can be simulated. Data that is necessary for tracking, navigation, and guidance of spacecrafts such as Crew Exploration Vehicle (CEV), Crew Launch Vehicle (CLV), and Lunar Relay Satellite (LRS) and orbit calculation data are disseminated to different NASA centers and updated periodically using the High Level Architecture (HLA). In addition, the performance of DSIL under different traffic loads with different mix of data and priorities are evaluated.

## I. Introduction

Space exploration has been one of NASA's visions, and NASA's Exploration Systems Mission Directorate (ESMD) is responsible for realizing this vision. Within ESMD, the Constellation program (CxP)[1] is providing the next generation of human transportation spacecraft, Orion (or Crew Exploration Vehicle, CEV). The Ares I and Ares V crew launch vehicles (CLVs) will provide the thrust. The Distributed System Integration Laboratories (DSIL) project fits under the CxP. The goal of DSIL is to develop capabilities to support systems integration, testing, and monitoring of an end-to-end distributed system architecture. DSIL involves multiple System

---

<sup>\*</sup>Member of the technical staff, Communication Architecture and Research Section of the Jet Propulsion Laboratory, 4800 Oak Grove Drive M/S 238-420, Pasadena, CA 91109.

<sup>†</sup>Member of the technical staff, Communication Architecture and Research Section of the Jet Propulsion Laboratory, 4800 Oak Grove Drive M/S 238-420, Pasadena, CA 91109.

<sup>‡</sup>Member of the technical staff, Communication Architecture and Research Section of the Jet Propulsion Laboratory, 4800 Oak Grove Drive M/S 238-420, Pasadena, CA 91109.

<sup>§</sup>Member of the technical staff, Communication Architecture and Research Section of the Jet Propulsion Laboratory, 4800 Oak Grove Drive M/S 238-420, Pasadena, CA 91109.

Integration Laboratories (SILs), simulators, emulators, testlabs, and control centers interacting with each other over a broadband network to provide virtual test systems for multiple test scenarios.

Currently, DSIL architecture contains the following simulated components: Orion crew exploration vehicle (CEV), Ares I crew launch vehicle (CLV), Space Communications and Navigation Network (SCaN), Ground Systems (LCS), International Space Station (ISS), and mission control center (MCC), where these are interconnected through the High Level Architecture (HLA). In the future, it is envisioned to extend DSIL architecture to include extra-vehicular activity (EVA), cargo launch vehicle (CaLV), and Lunar Surface Access Module (LSAM). In the early stage of DSIL work, the above components are simulated at different NASA centers and the requirements for reliable and accurate operations of each space vehicle and networks of connected vehicles are being assessed. Preliminary traffic measurement for operations required for DSIL as well as the Command, Control, Communications and Information (C3I) telemetry data streams are collected and analyzed. From these preliminary traffic measurements, we conclude that the HLA traffic is small compared to the available bandwidth from NASA Integrated Services Network (NISN) [2]. As the capabilities are being developed incrementally, DSIL will progress towards integrating higher fidelity simulations, emulations and including hardware-in-the-loop tests.

## II. Software & Network Infrastructure for DSIL

DSIL consists of multiple testlabs that are geographically located at different parts of the United States. These testlabs are connected through NISN [2]. The NISN backbone and the externally accessible IP addresses through the facility network forms the NASA Distributed Simulation network (DSNet) which is part of NISN. Currently, the DSIL testlabs that are connected through DSNet are located at Jet Propulsion Laboratory (JPL), Marshall Space Flight Center (MSFC), Johnson Space Center (JSC), Goddard Space Flight Center (GSFC) and Kennedy Space Center (KSC).

To enable communications and interoperability among different NASA centers within DSNet, HLA is chosen as the framework for distributed simulation. HLA [3] is a layered, event-driven software architecture that defines a set of common interfaces, models, and rules for distributed simulation. HLA was originally developed by the U.S. Department of Defense (DoD) Defense Modeling and Simulation Office (DMSO) to achieve interoperability across the large numbers of different types of simulations developed and maintained by the DoD [3]. Later, the core of HLA distributed simulation technology is adopted for the IEEE 1516 HLA standard. HLA is platform independent and it enables the combination of independent simulations into a single, comprehensive simulation. There are different implementations of this standard. DSIL is using the HLA implementation by Pitch [4]. The software implementing HLA specification is called Pitch Run-Time Infrastructure (RTI). An instance of a distributed simulation is called a federation where each simulation within the distributed simulation is a federate. A federation Object Model (FOM) defines the objects and interaction classes in the federation. Information exchange among federates are defined by FOM.

Specifically, in our simulation, the Crew Exploration Vehicle (CEV) federate uses the Advanced NASA Technology Architecture for Exploration Studies (ANTARES) software [5]. This simulation supports Orion Guidance Navigation and Control flight software evaluation and test, where the simulation is run from JSC. Mission Control Center (MCC) federate is also operated at JSC. The Launch Control Center (LCC) federate provides pre-launch capabilities and it is simulated at KSC. The Crew Launch Vehicle (Ares) simulation is called Ares Real-Time Environment for Modeling, Integration and Simulation (ARTEMIS) and it is run from MSFC. GSFC provides DSIL Interface Unit (DSILIU) test with JSC to measure DSNet performance. Space Communications and Navigation Network (SCaN) simulation is run from JPL. The purpose of SCaN simulation at the JPL is to simulate the end-to-end communications and networks systems to support CxP missions while interoperating with other SILs. Specifically, this is the communication network that transports data between Orion (or Ares) and Mission Control Center (MCC). SCaN consists of assets from Space Network (SN), Ground Network (GN), and potentially the Deep Space Network (DSN). Although NASA's Integrated Services Network (NISN) is not part of SCaN, it is on the data path between Orion and MCCs thus it is included in the simulation.

Spacecraft status such as position, velocity, and acceleration are exchanged among simulation federates through HLA. In addition, Orion and Ares also send telemetry data to MCC and (or) LCC. The C3I telemetry data are passed through User Datagram Protocol (UDP) sockets by using the Managed Automation Environment for Simulation, Test, and Real-time Operations (MAESTRO) developed at MSFC. MAESTRO provides a

configuration and control system for test conductor to setup and run a test. It also provides data distribution and data exchange to distribute data generated by models/hardware residing at different facilities.

To measure the end-to-end throughput and delay (of HLA and C3I data streams among different NASA centers) during simulation runs, Wireshark [6] was used. Wireshark is a freeware that measures the end-to-end protocol performance. In addition, to assess the bandwidths between network end-points, IXIA [7] was employed.

### III. SCaN Simulator and Interface Design

#### A. Network Simulation Tool (SCaN/MACHETE overview)

The SCaN Simulation tool[8] which is based on MACHETE [9] has been developed for the purpose of determining the performance of existing and emerging communications protocols and services in the context of space exploration. In addition, the SCaN simulator has a capability to interface with external inputs that are necessary for DSIL operations. Emphasis has been given to capturing the unique effects imposed by Space environment where the environment consists of the following general systems:

1. Orbital and planetary motion kinematics modeling: the simulation tool imports spacecraft positions for Orion and Ares and it uses assumptions to predict Tracking and Data Relay Satellite (TDRS) positions. From the given positions, the tool computes the range and propagation delay accordingly. Link dynamics (visibility) are either imported or computed by the network simulation tool. This is further described in a later section.
2. Traffic generation, protocol state machine modeling and execution: we have built models for the complete Consultative Committee for Space Data Systems (CCSDS) [10] protocol stack, including Proximity-1, Packet Telemetry, Advanced Orbiting Systems (AOS), and CCSDS File Data Protocol (CFDP), using Scalable Network Technologies' QualNet [11] simulation environment. On-board spacecraft data management system models have also been developed to capture the interaction among communications payloads, flight computer, persistent storage (Solid State Recorder, SSR), spacecraft busses, and traffic generated by on-board science instruments. In DSIL evaluations, the scenario consists of Orion, Ares, MCC, and TDRS bent-pipe. We are currently using an in-house link budget library to simulate the communications effects and the tool is capable of delaying data to simulate propagation delay effects. Figure 1 shows the current SCaN simulation scenario used for DSIL, where the dotted region shows a TDRS bent-pipe link.

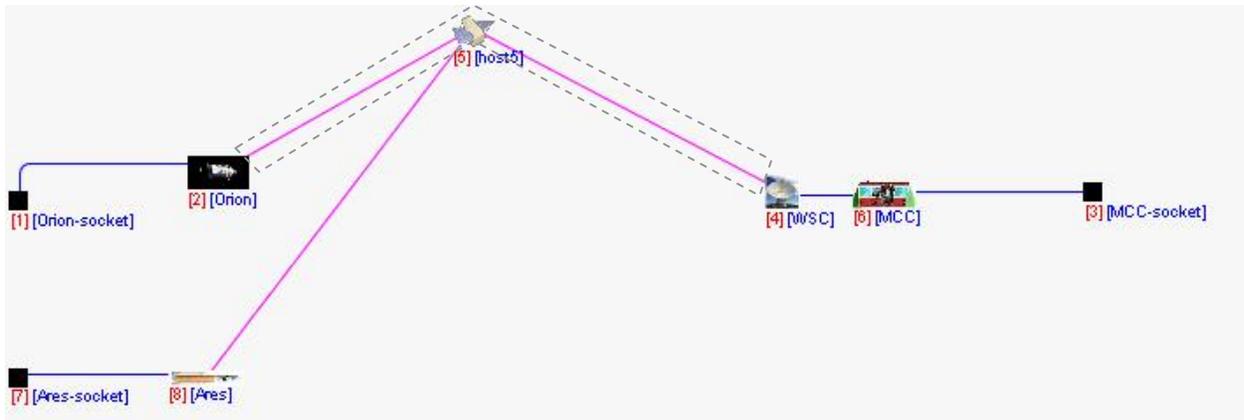


Figure 1. SCaN simulation Scenario

3. Real-time application interface: this feature enables the connection of MACHETE with external applications (e.g. middleware, protocol emulators) to perform real-time hybrid simulation/emulation. Currently, MACHETE has a HLA interface which is used to synchronize different federates and exchange global information such as spacecraft positions, velocities, positions etc. There is also an IP Network Emulator (IPNE) interface [11] to

accommodate for the transport of mission payload (C3I) data. Figure 2 shows the SCA-N-HLA interface and the SCA-N-MAESTRO (IPNE) interface. Through the SCA-N-HLA interface, the SCA-N simulator interacts with JSC, KSC, MSFC and LaRC. The SCA-N-MAESTRO interface enables the transport of 3CI telemetry between CEV (and / or CLV) and MCC. Nodes simulated by external federates are mapped to virtual simulation nodes in the SCA-N simulator to evaluate communications performance.

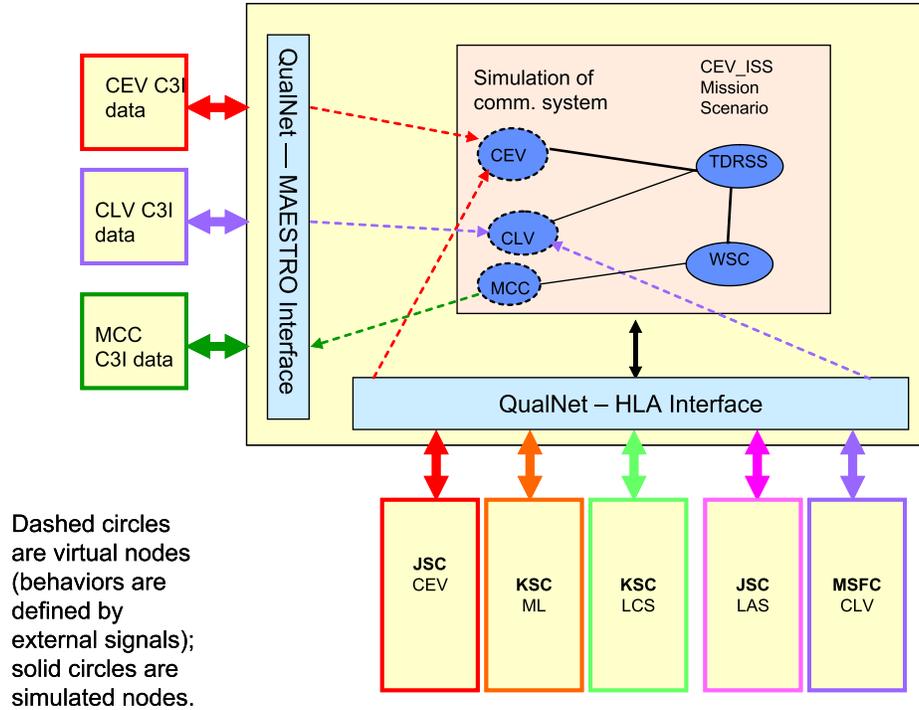


Figure 2. SCA-N HLA interface

Using this tool, technology researchers and mission designers can (1) determine system resource requirements such as bandwidth, buffer size, schedule allocations, etc for distributed simulation environment, (2) characterize performance benefits of new or alternative protocols, services, and operations, (3) validate new technologies for mission infusion and (4) enhance mission planning and operations.

### B. High Level Architecture Interface (Pitch HLA-RTI)

HLA has three components: 1) Interface specification, 2) Object Model Template (OMT), and 3) HLA Rules. Interface specification defines how simulator interacts with the Run-Time Infrastructure (RTI). The OMT provides a common framework for the communication between HLA simulations. The HLA rules defines the rules that simulations must obey. The Federation Object Model (FOM) is one of the templates under OMT to describe the relationships among data that the federate exchanges in a federation execution. In the current DSIL simulation, we have 6 federates as described earlier and they are defined as *Space Vehicles* in the FOM. For each federate or *Space Vehicle*, a total of 16 state information and attributes are defined that are necessary for accurate spacecraft operations. More state information and attributes can be added as the mission and its requirements become more complex.

In this work, the CEV JSC federate is acting as the master federate to initiate the HLA simulation. However, the Master federate role can be moved to other federates at other centers. One of the critical aspects about HLA simulation synchronization is achieved through an interactive time management (TM) scheme. In HLA, there is a time management (TM) scheme to manage, control, synchronize, and update time-interactions between different federate objects. Sending and receiving of vehicles' state information and attributes can have a RTI timestamp,

called Time Stamp Order (TSO) events. Depending on whether it is desirable to enable TSO event in HLA simulation, the TM scheme can be a combination of *time-regulating* and /or *time-constrained*. *Time-regulating* means a federate can send TSO events. By doing so, RTI can prevent other federates from advancing time until time-regulated federates have sent all the data that they are going to send before time advancement request. In *time-constrained* mode, a receiving federate can process TSO events in time stamped order. Hence, RTI prohibits time-constrained federates from advancing time until it has received all the data that may be sent by other federates up to time advancement request. In DSIL simulation, both *time-regulating* and *time-constrained* modes are used to synchronize time between federates and to analyze and trace simulation interactions.

### C. Managed Automation Environment for Simulation, Test and Real-time Operations (MAESTRO)

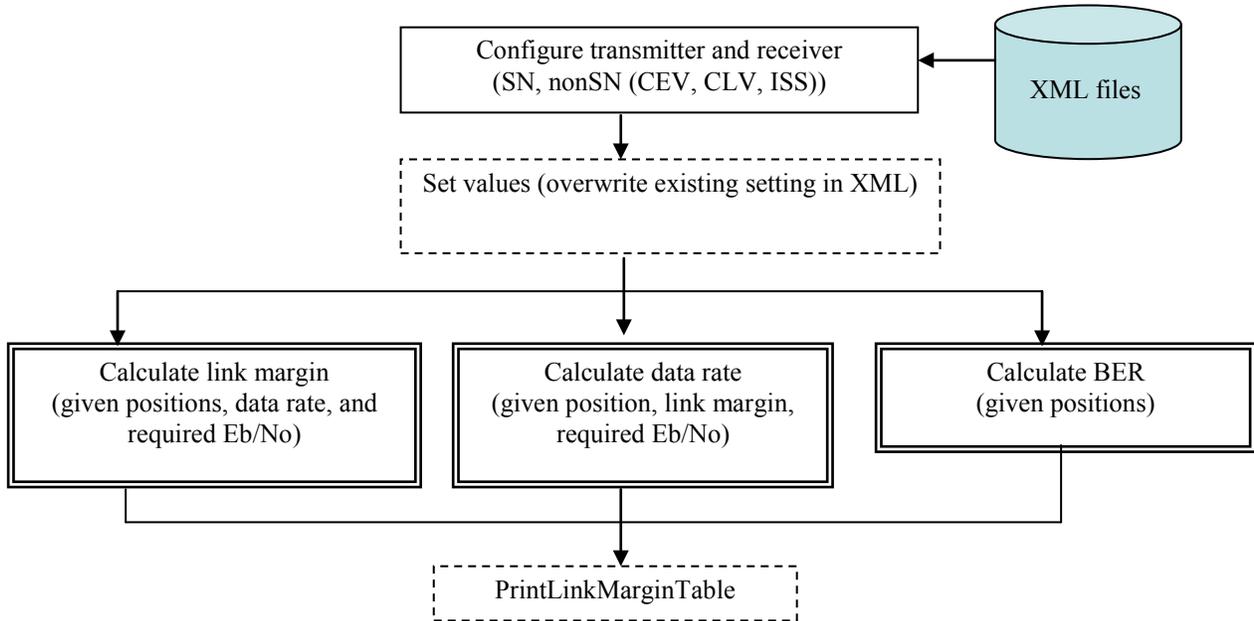
MAESTRO was developed at Marshal Space Flight Center. It is a software package that uses User Datagram Protocol (UDP) on a specified port to transport Command, Control, Communication, and Information (C3I) at a rate of 1 Hertz. The Data Exchange Message (DEM) is used as a base format for MAESTRO. In addition to the C3I data, DEM also contains information on time, fragmentation, and security in the functional context.

The Extensible Markup Language (XML) is used to define the Metadata for all the packets used in MAESTRO. The Metadata describes all the packet data structure, name, descriptions, data format type, and engineering units used in C3I data and it is sent once at the beginning of simulation to reduce overhead.

### D. TDRSS Link model and Link Budget Calculation Library

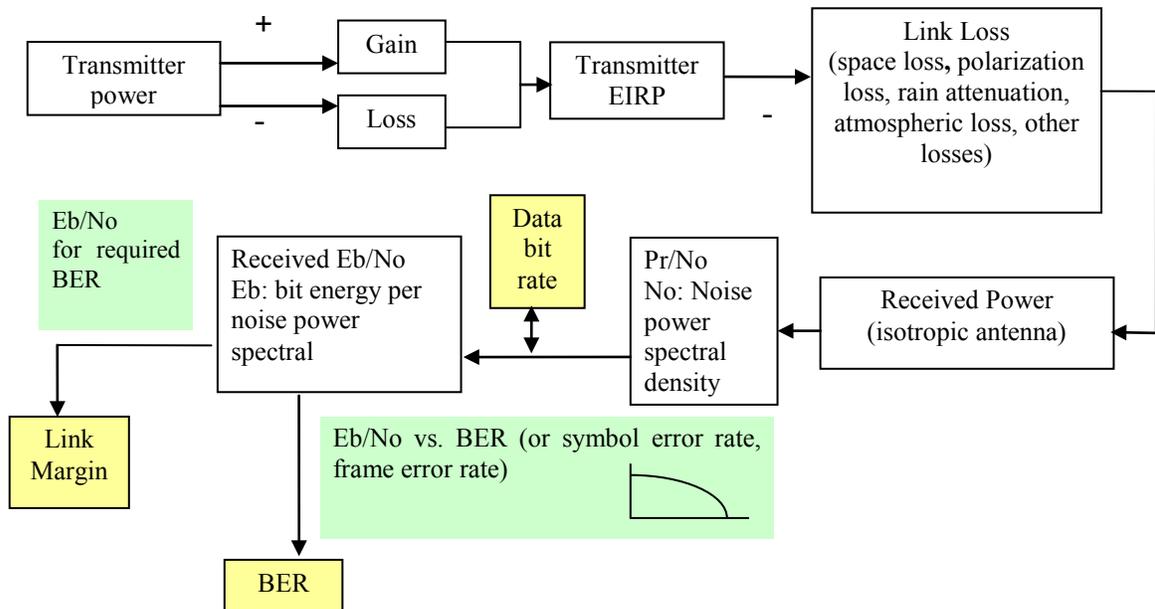
In order to evaluate and simulate the communications performance of CxP scenario described in Fig 1, we implemented the link budget calculation library and delay function in the SCaN Simulator based on the operational link budgets depicted in [18]. Hence, the SCaN simulator is capable of simulating TDRS link and it can simulate various Bit Error Rates (BERs) causing data loss according to various power, frequency and coding schemes. Further, we developed the internal Satellite bent-pipe model to accurately capture the bent-pipe behavior of TDRS in the SCaN simulator by using the link budget library to accurately model the physical layer. That is, the internal Satellite bent-pipe model calls the Link budge library to determine whether to drop the specific frames or not.

With this link budget calculation library, the CEV-SN and CLV-SN link budgets at various supporting frequency bands of different mission phases are calculated. The primary purpose of the link budget calculation is to evaluate the system operating point and to determine whether the error probability associated with that point meets the system requirements. The link budget calculation library is written in C++ and can be run on either Linux or Windows environments. All static communication parameters listed in [18] for link budget calculations are dictated in the XML format. Each DSIL node such as CEV and SN in the SCaN Simulator has one XML file composed of different frequency bands for various phases and they are hierarchically categorized. The proper XML files and relevant data are loaded based on the configuration of transmitter and receiver for the execution of link budget calculation. Moreover, some dynamic parameters needed for link budget calculation, such as the positions of the receiver and transmitter, are updated externally from HLA in real time. Some static data, such as the data rate, can be set externally after observing the performance variation as the setting changes. The entire link budget calculation library is constructed as shown in the following figure.



**Figure 3. Link budget calculation library**

In the first initialization step, both the transmitter and the receiver are configured and the proper static communication data are loaded from the XML files. Furthermore, some static data are optionally specified to overwrite the existing setting. There are three link budget relevant functions provided in the link budget calculation library. Each function calculates the link margin, data rate, and the BER based on user's request, respectively. The following picture illustrates in general how the link margin is calculated. The power received at the receiver is calculated by adding the gains and subtracting the losses to and from the transmitter power. At the receiver, the received  $E_b/N_0$  is calculated according to the data rate and it is further compared with the  $E_b/N_0$  of the required BER. This difference is the link margin.



**Figure 4. Detailed Link Budget calculation**

By default, the transmission is claimed to be at the safe side if the link margin exceeds 3 dB. The calculation of data rate is the reverse process of the link margin calculation. The user has to specify the expected link margin, then the achievable data rate is calculated according to the expected link margin and the power received at the receiver node. The calculation of the BER or Symbol Error Rate (SER) is to calculate the data error rate according to the required and corresponding  $E_b/N_0$  values. All calculations are based on the link budget calculation formula provided [18]. This library also provides a printing function which tabulates a balance list of calculated losses and gains.

#### IV. Network Traffic Analysis Methodology for C3I data

The DSIL simulation is performed under Distributed Simulation Network (DSNet), which is a collection of firewall facility local area networks that connected different NASA centers through NASA Integrated Services Network (NISN). The NISN provides the data transfer backbone for space missions. There are two separate networks for NISN: a mission network (Figure 5), controlled out of NASA Goddard Space Flight Center (GSFC), and an institutional network, controlled out of NASA Marshall Space Flight Center (MSFC). The NISN maintains staffed Gateway facilities at each NASA Center and most NASA facilities. Staffed sites include Ames Research Center (ARC), Dryden Flight Research Center (DFRC), Cluster Controller (CC), Jet Propulsion Laboratory (JPL), Johnson Space Center (JSC), Kennedy Space Center (KSC), Langley Research Center (LaRC), Michoud Assembly Center (MAF), MSFC, Stennis Space Center (SSC), Vandenberg Air Force Base (VAFB), and White Sands Test Facility (WSTF)/White Sands Complex (WSC). However, NISN does not provide or manage the local on-site networks at the various NASA centers.

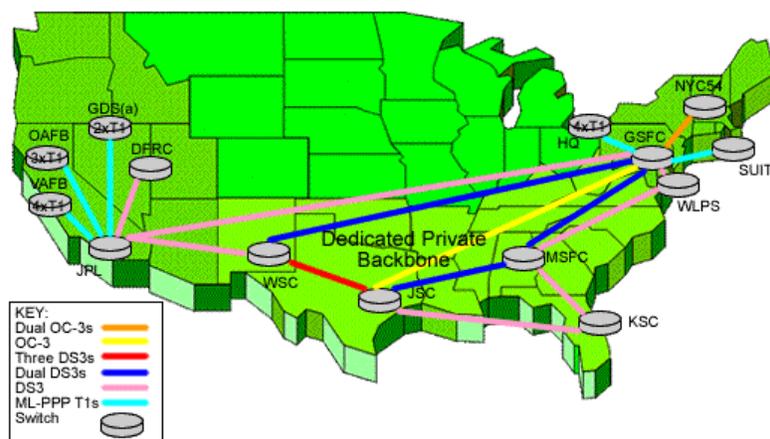


Figure 5. NISN Mission network [19]

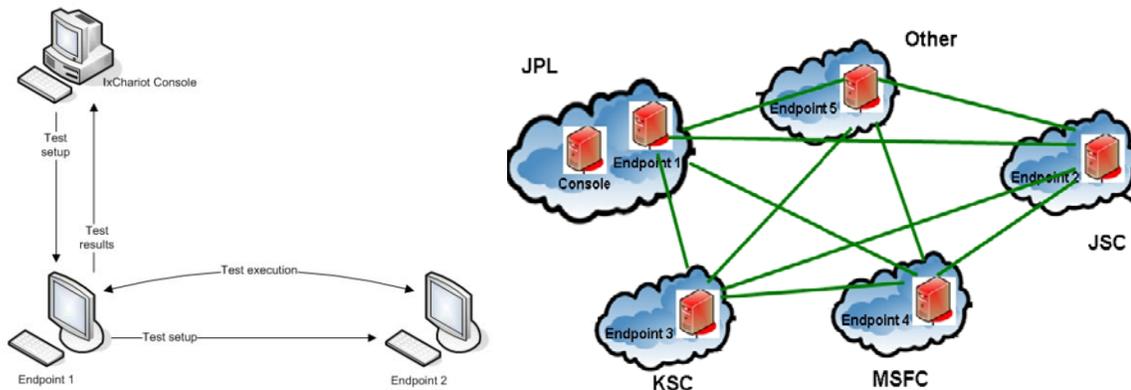
There are two ways data can flow through NISN network: 1) using the Internet Protocol Operational Network (IPONet) or 2) the High Data Rate System (HDRS). The IPONet uses the Transmission Control Protocol (TCP)/Internet Protocol (IP) or User Datagram Protocol (UDP)/IP, which is the standard way to transfer data on the Internet. The HDRS transports data rates from 2 Mbit/s to 48 Mbit/s for specialized missions requiring a high rate of data transfer and does not require the infrastructure of routers, switches and gateways to send its data forward like IPONet. The DSIL uses IPONet to transfer data between all the SILs. The SILs use TCP to simulate the truth data and UDP to transfer the C3I telemetry data, communicate with each other through the Internet Voice Distribution System (IVODS), and conduct video teleconferencing through VSee[17]. All these features consume certain amount of bandwidth, but NISN is only guaranteeing 15 Mbps limited bandwidth for DSIL from time to time. As a result, it is necessary for DSIL to characterize the NISN network to understand its capability and ensure certain standard of Quality of Services (QoS). Furthermore, it is important to monitor the traffic during the test to make sure that all the messages are being sent and received intact between different SILs with acceptable levels of

available bandwidths. Because DSIL is working with a constrained network, any waste in network bandwidth should be minimized and there is a strong need to quantify these values. Currently, we use IxChariot [7] to characterize the NISN network and Wireshark [6] to monitor the incoming and outgoing traffic at the JPL SIL. However, it is desirable to implement netflow [15] in the future to monitor traffic in real-time during the test to monitor bandwidth usages. The DSNet currently guarantees 15Mbps throughput for reliable DSIL operations. There are TCP traffic generated for HLA operations as well as separate C3I data telemetry streams which are sent as UDP packet. In this section, we quantitatively measure traffic coming to the SCaN SIL. The preliminarily empirical data traffic measurement show that the required HLA traffic is very small and does not interfere with the C3I UDP data stream.

### A. Traffic Monitoring and Throughput Benchmarking

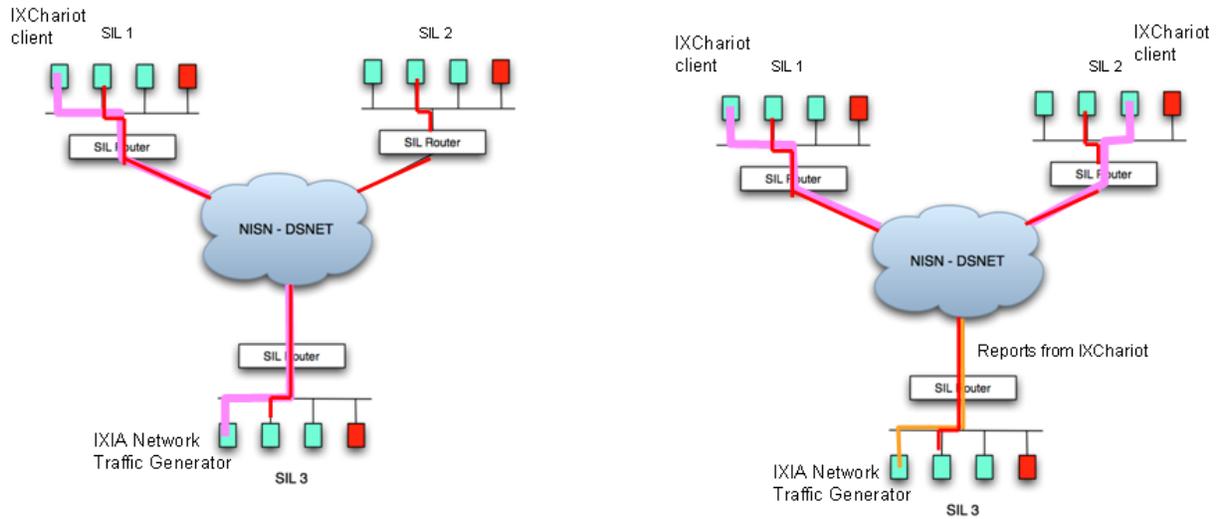
IxChariot is a commercial software tool developed by IXIA[7], it is considered to be a leading test tool for simulating real-world applications for predicting device and system performance under realistic load conditions. Furthermore, it provides the ability to assess the performance characteristic of any application running on wired and wireless networks. Some of IXIA’s capabilities include: 1) over 150 application scripts for simulation Enterprise, Tripple-Play, and Internet traffic 2) real-world application behavior at the transport layer, 3) Separate control and data-plane activity creation using application groups, 4) tailored scripting capabilities, 5) throughput, jitter, packet lost, and end-to-end delay measurement capabilities, 6) embedded custom payloads to test specific data content across the network, etc. Currently, we measure the throughput, jitter, packet loss, and end-to-end delay in the NSIN using IxChariot.

Installing and using IxChariot is required a single console and a pair of clients (endpoints). Figure 6 shows an example of how to connect and communicate between the console and endpoints pairs using IxChariot. At the beginning, the test conductor creates a test scenario in the IxChariot console and executes the test. At this point, the console establishes communications with end-points and sends test setup information to endpoint 1. Then, endpoint 1 establishes connections and sends test setup information to endpoint 2. When endpoint 2 has acknowledged the connection, endpoint 1 replies to the console. When all endpoint pairs are ready, the console directs them to start. After that, the two endpoints execute the test. Endpoint 1 collects the test results and sends results to the console. Currently, MSFC SILs is the only SIL that has installed IxChariot client. However, we anticipate that other SILs will participate in the near future.



**Figure 6. IxChariot connections between different SILs**

By injecting additional traffic during the test, we can benchmark DSIL traffic because we are able to figure out how much additional traffic DSIL can handle. This can be achieved by using IxChariot to inject additional traffic during the test. For example, the red flow represents normal DSIL test traffic, the pink flow is the additional injected traffic. Thus, we can collect statistics at all SILs and use the statistics to calculate the drop rate to benchmark DSIL's effective bandwidth.



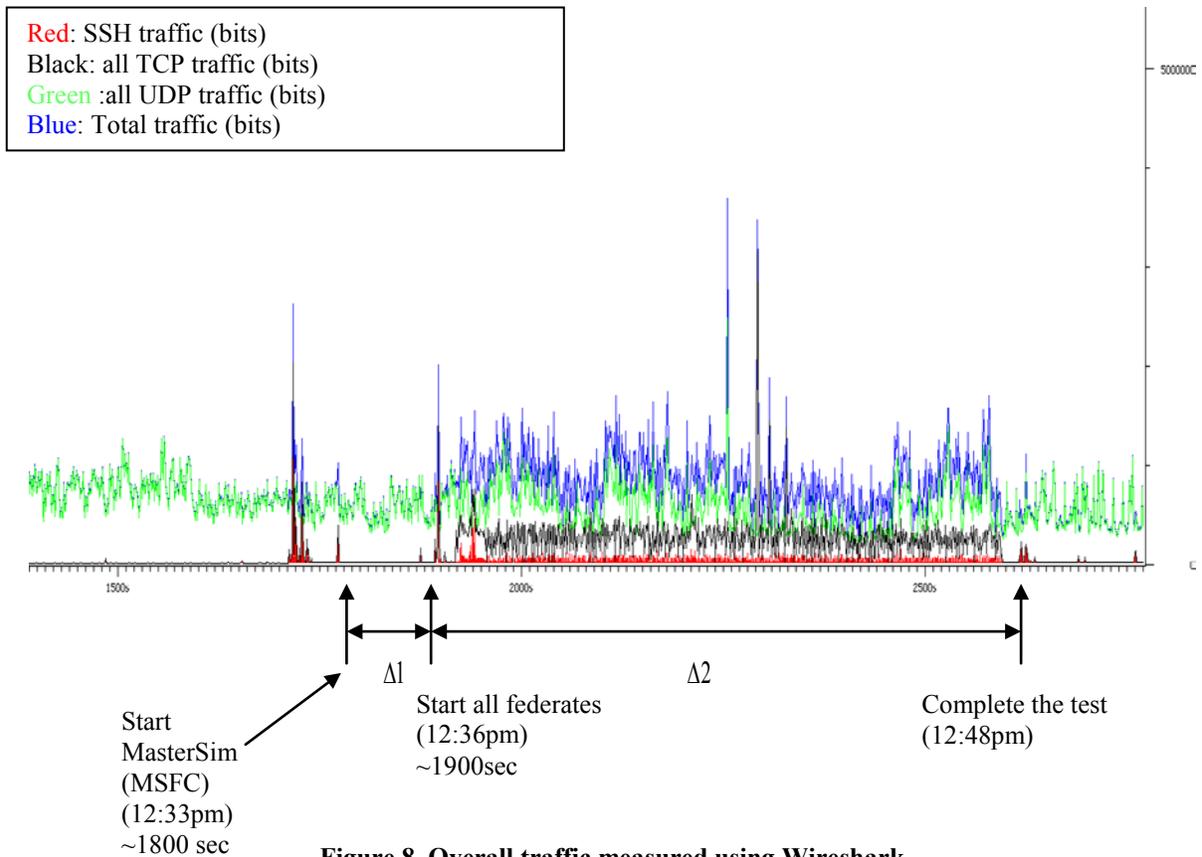
**Figure 7. IxChariot traffic generation**

## **B. End-to-end Traffic measurement**

To measure the incoming and outgoing traffic at JPL, Wireshark[6] (freeware, also known as Ethereal) is used. Wireshark is a “packet sniffer”. This tool can be used to determine the amount of data loss on DSNet between centers and variable delays on the DSNet.

## **Empirical Traffic**

During a test run of a 11-minute Launch/Ascent scenario, we measured traffic coming to JPL from different NASA centers and classified traffic by source and by protocols. All TCP, UDP, and SSH traffic coming are captured in the following figure and shown in the different colors.



**Figure 8. Overall traffic measured using Wireshark**

In Figure 8,  $\Delta 1$  depicts the time interval for test initialization and setup;  $\Delta 2$  marks the actual scenario run.

In Figure 8, UDP data include C3I telemetry data, as well as VSEE (video of different labs running the test); TCP include HLA data and internal network operation traffic. SSH is using TCP but we separated it out in the figure. From the TCP traffic, we can identify the duration of the scenario by looking for patterns that match the HLA traffic. We observed that C3I telemetry data from CEV and CLV are relatively small compared to VSEE where VSEE dominates the UDP traffic.

The total CEV data is being generated from CEV is 0.03 Mbps and 0.012 Mbps for CLV. This is consistent with what actually was sent from the CEV at JSC and CLV at MSFC. The minimum required bandwidth is 0.042 Mbps for the UDP C3I telemetry for this initial test. The rest of the UDP traffic consists of VSEE, the minimum UDP bandwidth required for DSIL operations are about 2.5 Mbps for this scenario. In the future, it is expected that CEV and CLV will send more C3I telemetry and we can analyze the performance in the similar way described in this work.

In Figure 8, the total TCP traffic is plotted with black color and the majority of TCP traffics are HLA traffic. The cumulative average of TCP traffic during the test is about 0.25 Mbps. In the following figure, all HLA traffic measured at JPL are separated according to the senders. The data points with pink color is all incoming HLA traffic at JPL. The plots with red, green, and blue indicates the HLA traffic received from JSC, MSFC, and KSC respectively.

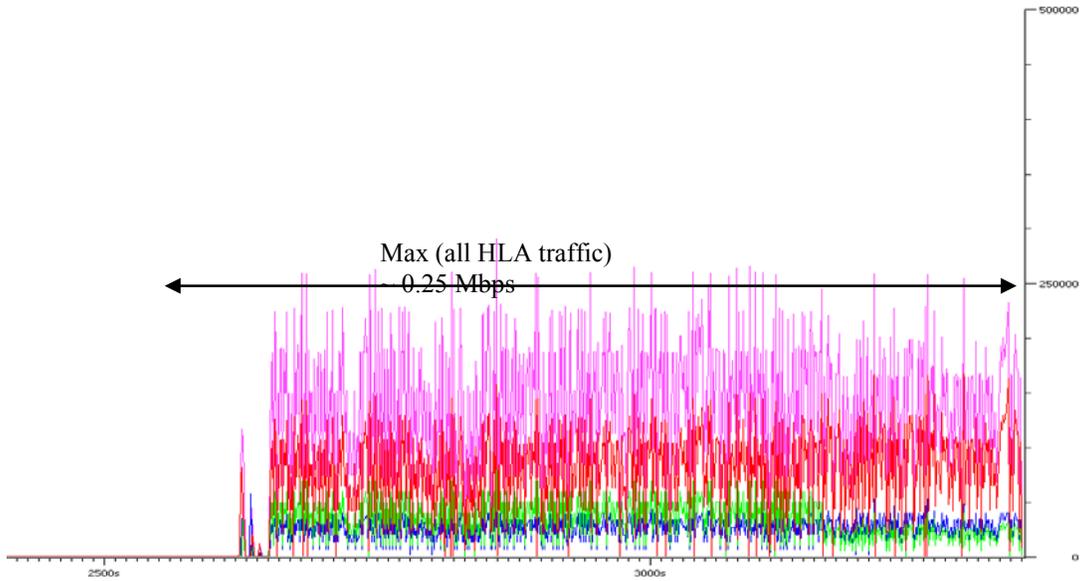


Figure 9. HLA traffic measured at JPL

We can see that the bandwidth required from HLA operations is 0.25Mbps. Hence, we can see that the HLA traffic is relative small compared to the available bandwidth given by the DSNet.

### C. NISN Characterization between JPL and MSFC

#### C.1 Latency

Currently, MSFC SIL is the only center that installed IxChariot client. We can only characterize NISN only the link between JPL and MSFC. First, we measure delay by sending 5000 UDP packets of sizes 654 (and 1698 bytes). The following figure shows the round-trip time delay from sending the packet to the receiving application sending back an acknowledgement. For example, the average round-trip delay is 95 ms for packet size of 654 bytes; the minimum is 94ms, and maximum is 97 ms. The delay is accurate with 95 percent confidence. For the packet with size of 1698 bytes, the average round-trip delay is 96 ms; the minimum is 96ms, and the maximum is 131ms with 95 percent confidence level.

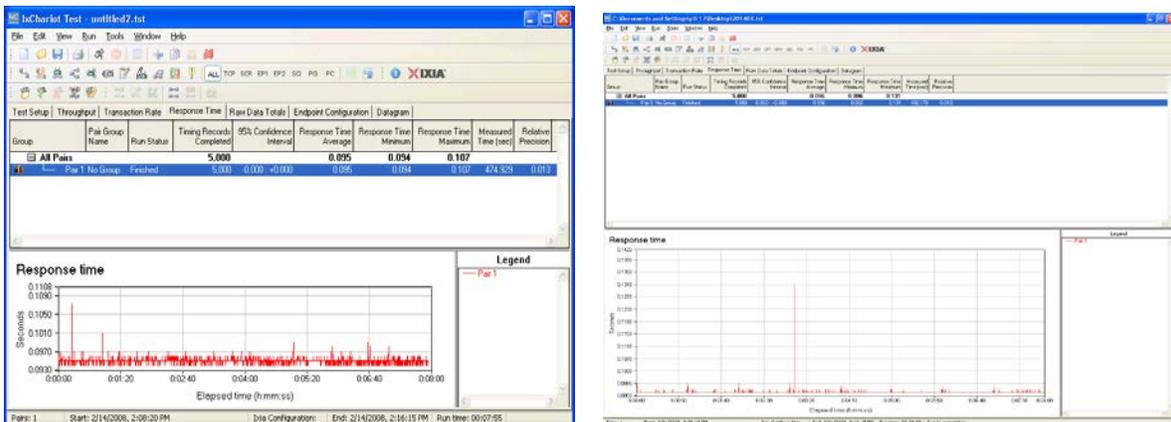


Figure 10. Latency measurement using IxChariot

## C.2 Bandwidth

### C.2.1 TCP Bandwidth Measurement

We also measured the TCP throughput, although it varies throughout the day, the throughput nevertheless is always less than 3 Mbps. Here is a snapshot of an instance of TCP throughput measurement between JPL and MSFC. The average is 2.32 Mbps; the minimum is 1.806 Mbps, and the maximum is 2.657 Mbps, with 95 percent confidence interval of [2.251, 2.38] Mbps.

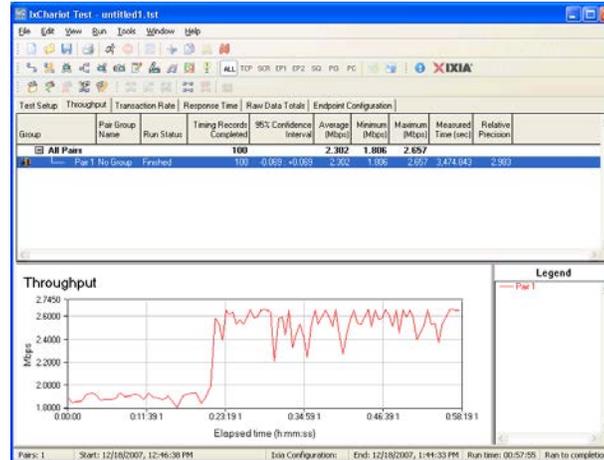


Figure 11. Throughput measurement using IxChariot.

### C.2.2 UDP Bandwidth Measurement

To measure the UDP throughput from one SIL to the other SIL over the DSNet, we send 365 MB of data, in packet sizes of 500 B, 1000 B, 1698 B, 10,000 B, 16,210 B, and 32,700 B with data rate between 10 Mbps to 140 Mbps. From the percentage of packet drop, we determine the optimal UDP packet sizes with respect to different data rates to minimize the packet drop over the DSNet. This is shown in the Figure 12, where X-axis is data rate in Mbps and the y-axis is the percentage of packet drop. Hence, packet size of 1698 B (and 5000 B) yield the lowest packet drop rate for data rates less than 50 Mbps (and 80 Mbps).

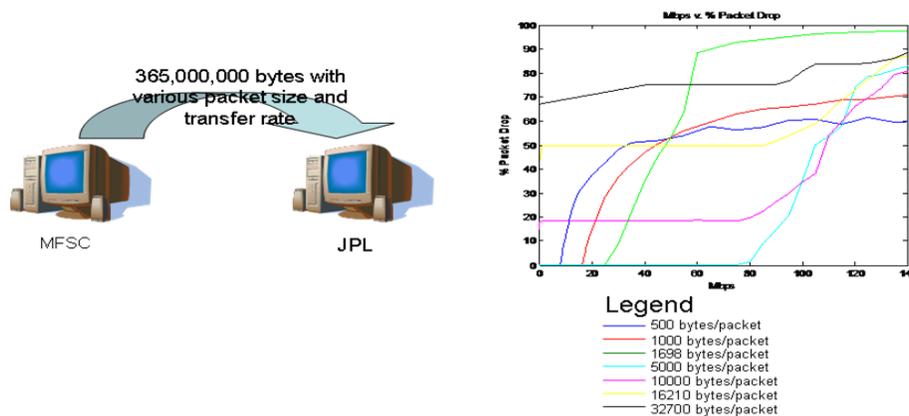


Figure 12. Packet drop rate as a function of transmission rates for different packet sizes

## V. Conclusions and Future work

We have successfully participated in distributed simulations of the Launch/Ascent scenario for the Distributed System Integration Laboratories (DSIL) involving different NASA centers and tested interoperability through HLA. In addition, we implemented SCaN simulation to be used within HLA infrastructure and framework to simulate the performance of CxP scenario. The collective behavior of different data traffic coming to JPL is captured and analyzed. The assessment of the minimum bandwidth required to achieve the successful DSIL operation is shown in this work. Current development work is underway to continuously monitor and characterize the network performance on different DSIL data traffic. We are also leveraging off the development of SCaN simulation tool and protocols in another project to achieve higher fidelity of the network models. As the CxP mission requirements become more complex, we envision that there will be a more need to perform the distributed simulation to analyze the mission requirements in various aspects while coherently achieving the real-time simulation.

### Acknowledgments

The authors would like to thank other fellow engineers and researchers involved in DSIL project from the Johnson Space center, Kennedy Space Center, Marshall Space Flight Center, and Langley Research Center. In particular, the authors like to thank Zack Crues, David Hasan, Dan Dexter, and Cindy Stemple for their invaluable supports on this project and appreciate Laura Hood, Polly Estabrook, and Gail Klein for their general leadership for this project. The research 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.

### References

- <sup>1</sup>NASA Constellation Program(CxP), [http://www.nasa.gov/mission\\_pages/constellation/main/index.html](http://www.nasa.gov/mission_pages/constellation/main/index.html)
- <sup>2</sup>NASA Integrated Services Network, <http://www.nisn.nasa.gov>
- <sup>3</sup>Simulation Interoperability Standards Committee (SISC) of the IEEE Computer Society, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) — Framework and Rules", The Institute of Electrical and Electronics Engineers, 3 Park Avenue, New York, NY 10016-5997, USA, 11 December 2000, ISBN 0-7381-2620-9 SS94882.
- <sup>4</sup>Pitch RTI, <http://www.pitch.se>
- <sup>5</sup>Chung, V., Crues E. Z., Blum M. B., Alofs C. A., and Busto, J. M., "An Orion/Ares I Launch and Ascent Simulation - One Segment of the Distributed Space Exploration Simulation (DSES)", AIAA Modeling and Simulation Technologies Conference and Exhibit, August 2007, South Carolina
- <sup>6</sup>Wireshark, <http://www.wireshark.org/>
- <sup>7</sup>IXIA, <http://www.ixia.com>
- <sup>8</sup>Jennings, E., and Heckman, D., "Performance Characterization of Space Communications and Navigation (SCaN) Network by Simulation", *IEEE Aerospace Conference*, March, 2008
- <sup>9</sup>Gao, J., Jennings E., Clare, L. SeGui, J., and Kwong, W., "MACHETE: A Tool for Architectural Modeling, Performance Characterization, and Technology Infusion of Space-Based Networks," *AIAA International Communications Satellite Systems Conference (ICSSC)*, Rome, Italy, September 2005.
- <sup>10</sup>The Consultative Committee for Space Data Systems, <http://public.ccsds.org/default.aspx>
- <sup>11</sup>QualNet, <http://www.qualnet.com>
- <sup>12</sup>Gibson, J., "Distributed Space Exploration Simulation", Internal Report
- <sup>13</sup>Kuhl, F., Weatherly, R., and Dahmann, J., "Creating Computer Simulation Systems: An Introduction to the High Level Architecture", Prentice Hall, 1999.
- <sup>14</sup>HLA course module, <http://www.ecst.csuchico.edu/~hla/>
- <sup>15</sup>Internet Engineering Task Force (IETF) Request for Comments (RFC) 3954 - Cisco Systems NetFlow Services Export Ver. 9
- <sup>16</sup>Tcpdump <http://www.tcpdump.org/>
- <sup>17</sup>Vseelab, <http://www.vseelab.com/>
- <sup>18</sup>Constellation Program Command, Control, Communication, and Information C(3I) Interoperability Standards book, Volume 3
- <sup>19</sup>NISN Mission network, <https://www.spacecomm.nasa.gov/spacecomm/programs/NISN/missionnetwork.cfm>