IWSCFF-2017 17-25

A MASSIVE-SCALE SATELLITE CONSTELLATION HARDWARE-IN-THE-LOOP SIMULATOR AND ITS APPLICATIONS

Christopher F. DeGraw[*], Marcus J. Holzinger[†]

Given the growing interest in satellite constellations, there is a surprising lack of rigorously tested operations and control methods for constellations larger than 30 to 50 spacecraft. The purpose of this paper is to discuss the development of a robust, modular and scalable system able to provide software-in-the-loop (SWIL) and hardware-in-the-loop (HWIL) simulation capabilities for the advancement of formation and constellation Flight Software Technology Readiness Levels (FSW TRL). The system being developed is called COSMoS (COnstellation Simulation on a Massive Scale). The goal of this system is to a) enable massive-scale (100+ satellite) realistic testing and characterization of formation control and operations techniques, b) examine the scaling of centralized and decentralized formation guidance, navigation, and control algorithms, and c) improve Technology Readiness Levels in realistic conditions. Examples of developing capabilities include the implementation of control schemes using a Minimum Lyapunov Error approach as well as semi- and fully-autonomous decision making systems.[1]

INTRODUCTION

Smallsats have become an economically viable route for commercial space development on a massive scale thanks to the recent miniaturization and standardization of satellite components.[2] The reduced cost and risk associated with smallsats has led entrepreneurs to find more uses for such constellations and enabled them to find the financial backing to implement them. Additionally, public sector entities continue to operate and design large constellations for a variety of purposes both civilian and military. These constellations vary from large constellations of large satellites, such as the GPS constellation, to large constellations of small satellites like the Planet Labs Earth observation constellation.[3,4] There are many other constellation paradigms including mixtures of cheap smallsats in LEO coordinating with large, expensive assets in GEO like the Sirius XM constellation.[5] The number of proposed constellations increases the chances that orbital debris and malfunctioning or dead satellites will pose a risk to operational systems. With plans by SpaceX, OneWeb and others to launch constellations containing thousands of satellites, the level of risk involved in these systems must be considered.[6,7]

One theoretical paradigm for satellite constellations is the development of satellite "swarms". There appears to be no globally agreed upon definition separating a "swarm" from a constellation, but a useful definition used by Verhoeven et. al. specifies that the individual elements of a swarm be functionally identical.[8] Such swarms provide reliability and performance through the interactions of

[*] Master's Candidate, The Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Georgia Institute of Technology North Ave NW, Atlanta, GA 30332. AAS and AIAA Student Member

[†] Assistant Professor, The Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Georgia Institute of Technology North Ave NW, Atlanta, GA 30332. AIAA Senior Member

their members without a requirement for the survival or availability of any particular member of the swarm. They also rely on the emergent behaviors created by the interaction of swarm members.[8–10] The reliance on emergent behaviors and distributed control systems can make analytical evaluation of the swarm behavior difficult thus requiring a robust simulation environment to evaluate swarm behavior prior to operational testing. When this paper refers to "constellations" the definition will include current paradigms and proposed swarm architectures.

Given the challenges involved in placing experimental satellites on orbit, high-fidelity simulations are necessary to retire risk associated with constellation deployment and operation while avoiding purely in-flight testing missions. Currently, there is no multi-purpose, large-scale constellation simulator capable of testing the interactions between members of a constellation, interactions between constellations, constellation interactions with ground operations, or the response of a constellation to accidental or intentional interference. One proposed simulator suggests the use of cloud computing to scale single satellite simulations up to a constellation simulation.[11] Unfortunately, the latency associated with cloud computing does not permit the capability for the integration of flight hardware into a real-time simulation.

Hardware-in-the-loop (HWIL) simulation has become a common industry practice for retiring risk in experimental and production technologies. The testing of flight and flight-like hardware with high technology readiness level (TRL) in experimentally validated simulations increases confidence in its performance on planned missions, but also allows for increasing the TRL of untested technologies by incorporating them in well established simulations. Marko Bacic quantifies the quality of an HWIL simulation in terms of "transparency" and robustness of prediction. "Transparency" is a measure of how well the interaction between hardware and the simulation matches the expected interactions between the hardware and its operating environment. Robustness of prediction is the measure of how well simulation results match experimental and operational results.[12]

The purpose of this research is to build a scalable, HWIL and software-in-the-loop (SWIL) simulator for constellations that is robust, transparent, and modular. This capability will allow for the improvement of TRLs through high-fidelity HWIL simulation. The enabling of massive-scale, realistic testing of constellations will help to develop, test, and characterize new techniques of formation control and operations. The development of this system will also help to understand how centralized and decentralized formation guidance, navigation, and control methods will scale as the size of constellations grows by one to two orders of magnitude from current levels.

The project is called Constellation Simulation on a Massive Scale, or COSMoS. The initial operational goal of COSMoS is to test different control algorithms on a cohesive 94-satellite constellation and determine the efficacy and efficiency of those methods. This paper will discuss 1) the conceptual design behind COSMoS and 2) its implementation and plans for its future use. Because COSMoS is designed with the integration of external hardware in mind, mention will also be made of hardware available for use by Georgia Tech's Space Systems Design Laboratory (SSDL).

POTENTIAL APPLICATIONS

The initial research goal for this project is to test the efficacy of newer, theoretical constellation control techniques, such as a Minimum Lyapunov Error approach, against currently implemented algorithms such as the Walker and Flower approaches.[1] Follow-up tasks include testing the robustness of such control systems in Blue-Grey-Red scenarios where constellation members are exhibiting anomalous behavior due to faults (Grey) or malicious actors are deliberately interfering with constellation operations (Red).

Another goal is to evaluate how current methods for ground station control of satellite constellations function with a massive-scale constellation or swarm. Individual element control is antithetical to swarm designs, and so new paradigms will need to be developed for the interaction of human controllers with autonomous or semi-autonomous constellations. This simulator will allow for the testing of control algorithms and interactions between human operators and the constellation in nominal and off-nominal (including crisis) scenarios. The development of such autonomous approaches will require a study of optimal satellite-to-satellite communication networks, distributed decision making algorithms and constant verification and validation of the simulation against theoretical models for neural networks and other topics in AI research. However, testing the algorithms developed by subject matter experts will only require the integration of these algorithms into the COSMoS simulation environment.

COSMoS will also be designed to interact with FlatSats undergoing testing and the ground support equipment (GSE) in Georgia Tech's Mission Operations Center. This will provide an opportunity for Georgia Tech's student ground controllers to interact with their individual satellites in a realistic environment prior to launch. It will also provide an opportunity to develop and test practices for the interaction between ground controllers and massive-scale satellite constellations.

A final concept for the use of the simulator is its application to systems other than satellite constellations. The simulator is designed for a high degree of modularity, with the intent that the framework could be used to simulate the behavior of any set of independent actors (agents) within a swarm. It is the belief of the authors that, from a mathematical perspective, there might be very little difference between a constellation of satellites and a swarm of self-driving cars. As such, the simulator hardware and application program interface (API) is named the Multi-Agent Network Distributed Simulator (MADNS) while the particular application being discussed here is COSMoS. As COSMoS is the first, and currently only, application for MADNS this paper will refer to the system as COSMoS.

CONCEPTUAL DESIGN

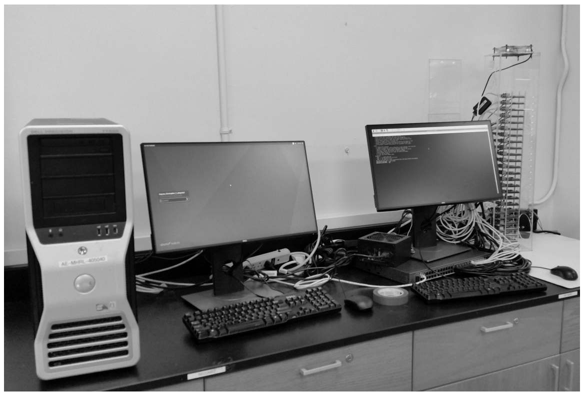

Figure 1. Current COSMoS System.

HWIL simulation quality is the guiding design principle behind COSMoS. To this end, there is an emphasis on simulation fidelity, modularity, and the strict separation of noumenological and phenomenological processes. The terms noumenological and phenomenological are borrowed from

Kantian philosophy where a noumenon is a "thing as it is in itself", in this case simulation truth, while phenomenons are the measurements of events such as the navigation solution.[13] To have the highest simulation fidelity possible, the architecture will not have its own simulation code but will use plug-in modules developed by subject-matter experts. The heart of COSMoS will therefore be an API designed to provide maximum transparency between the simulation environment and the hardware and/or software being tested. Robustness of prediction will come from the fidelity of the physical models and the high level of transparency.

Modularity is accomplished through both hardware and software design. The current design includes a host computer, two managed gigabit network switches, and 94 Raspberry Pi 3B (RPi) micro-computers. Each RPi will host a single simulated satellite and will referred to as agents. The COSMoS environment will include a wrapper designed to provide maximum transparency for the flight software (FSW) running on the agents, while the host computer runs the simulated environment and processes all communications between agents.

Because COSMoS is fundamentally an HWIL system, there must exist strict separation between the noumenological processes involved in developing simulation truth and the phenomenological processes which provide inputs for FSW and flight hardware. A design diagram showing a dual-loop FSW routine operating under this regime can be seen in Figure 2. The result is a closed-loop control system which presents flight hardware and software with an environment that is as realistic, from the perspective of the flight system, as possible.

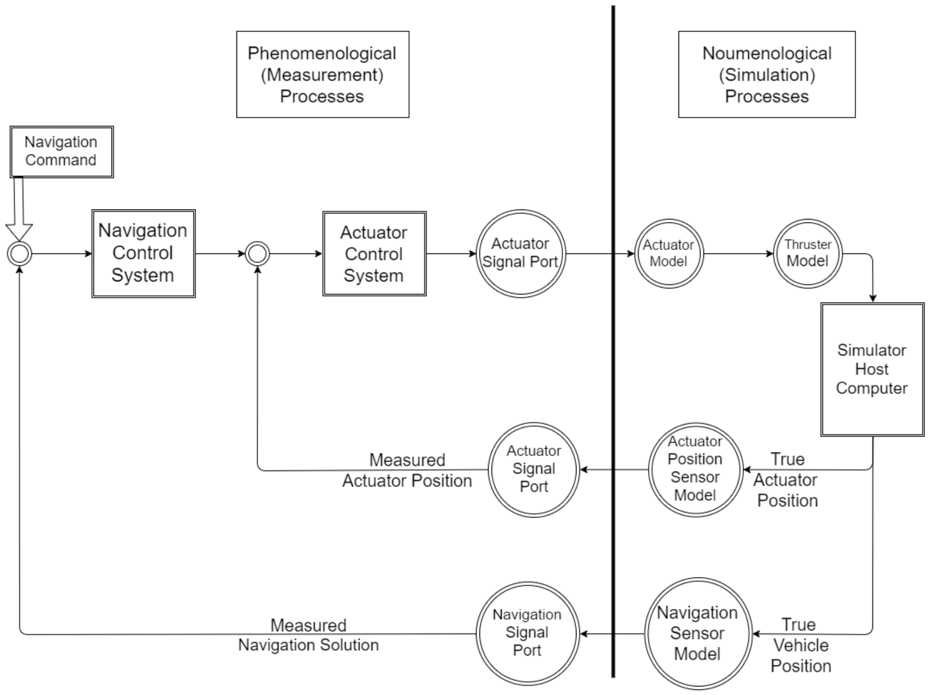

Figure 2. Control scheme with strict noumenonological and phenomenonological separation.

For the COSMoS project, each RPi board will run NASA's Core Flight Executive (cFE) within a wrapper designed to interface with its specific FSW.[14] This allows for the testing of mission-specific software and hardware in an environment with high-fidelity physics and operations. A conceptual schematic of data-flow within the system can be seen in Figure 3.

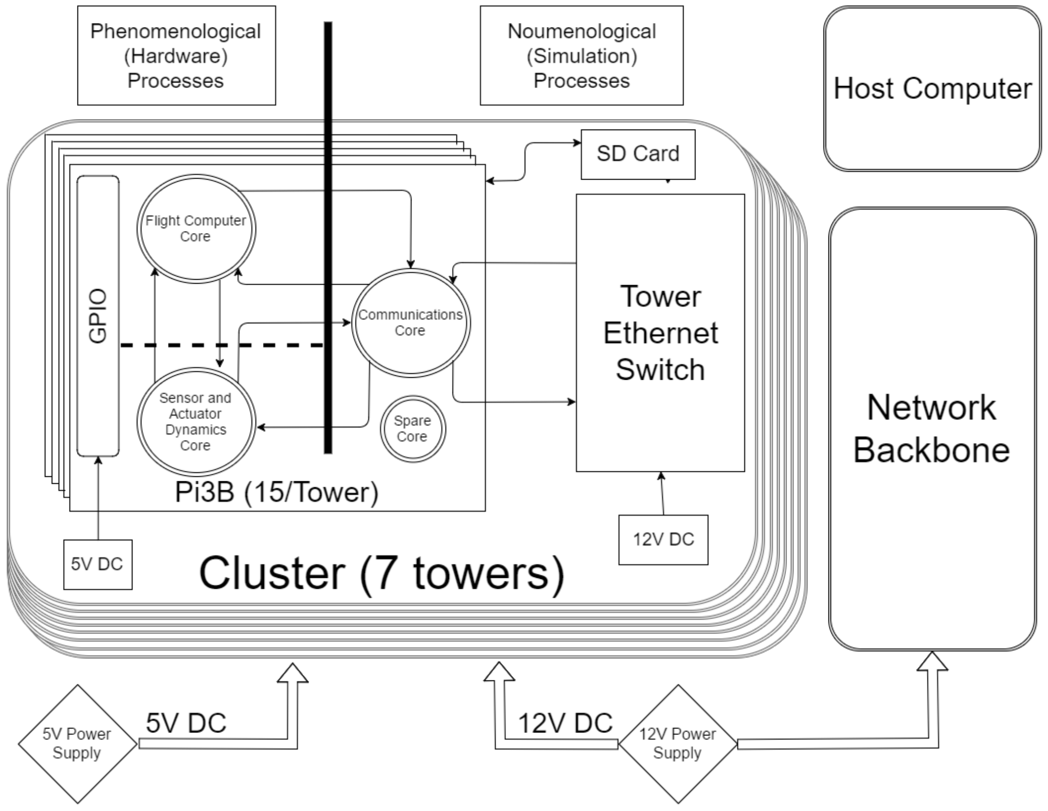

Figure 3. High-level system design and data-flow.

Software and Hardware

Software. The software is designed to be as modular as possible while still remaining HWIL-ready. The host computer for the constellation runs on Ubuntu 14.4 and not on a true real-time operating system (RTOS). This generates a requirement that the simulation environment software monitor computational sub-processes for completion before their results are needed by a real-time function and handle real-time exceptions gracefully. Without a true RTOS there is no guarantee of time accuracy, so it will be necessary to monitor time deltas that develop in the system and determine if they are within acceptable limits for the simulation.

Following the modularity guiding principle, all computations not involving data transfer to and from the simulated satellites will be handled through module calls. The current plan is to use SGP4 as the primary simulation module, but this could be replaced by models with higher fidelity or different outputs for particular applications. While the initial system may not be able to handle high-resolution simulations, a future goal is to allow the porting of the main simulation environment to a more capable computer or even an HPC system.

The modularity principle will also enforce the noumenological/phenomenological division by requiring a logical separation of truth functions from any flight elements as shown in Figure 2. The cFE will run within a wrapper designed to pass information to and from the FSW as if it were installed on a real flight computer. This wrapper will only be exposed to phenomenological values generated by the flight code or measurements which have passed through a sensor model. The processes responsible for converting actuator commands into actuator positions will be logically separated from the wrapper as will the sensor models responsible for “fuzzing” the noumenons. The host computer will only process noumenons during a simulation. Phenomenological values will be stored on the agents and collected at the end of a simulation run. The only exposure of the host computer to phenomenological values will be incoming satellite telemetry, which it will route directly to display and storage devices.

Hardware. The system hardware design also follows the principles of modularity, simulation fidelity and noumenon/phenomenon separation. The 94 RPi micro-computers will be divided into four towers, each holding 23 or 24 RPi boards encased in an acrylic enclosure. Each tower is a separate unit with access ports for power, Ethernet, and other mission-specific communication pathways. Two 48-port managed network switches will each support 47 agents with one port left free to connect to the host computer or the other switch. A CAD model of a tower is in Figure 4 and the first tower prototype can be seen in Figure 5.

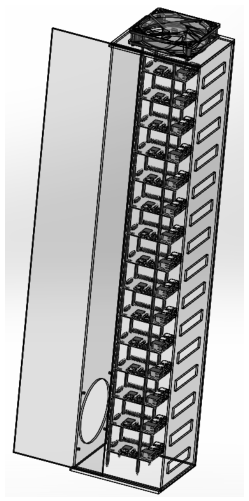
Figure 4. Tower CAD model.

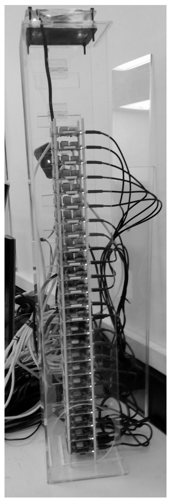
Figure 5. Tower prototype photograph.

A hardware connection diagram for the entire system can be seen in Figure 6

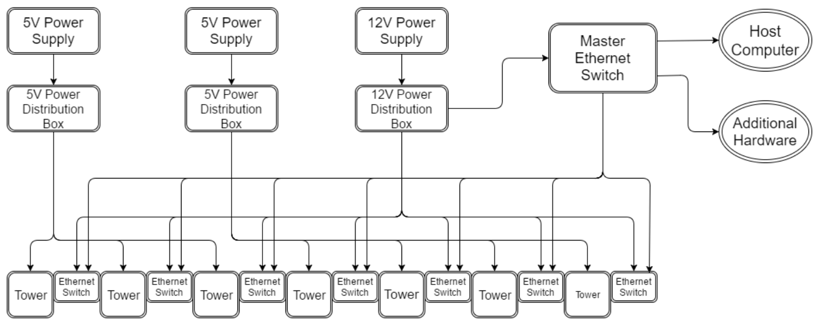

Figure 6. Hardware layout for the system.

Host Computer The host computer will be responsible for maintaining the system truth values such as the 6-DoF position and attitude of each satellite and their time derivatives. The current host computer contains an nVIDIA Tesla CUDA GPU which will be used for simulation processing, although alternate libraries could be developed for other architectures should they prove more effective. Additionally, the host computer will create environmental information for each satellite such as radiation, magnetic field and sun exposure as needed for specific mission requirements. Finally, the host computer will be responsible for storing and routing incoming telemetry data from each member of the constellation.

An eventual goal of this system is to determine if constellations will be able to act autonomously. Therefore, the host computer will also run a scalable matrix matching algorithm developed by the authors to determine which satellites have line-of-sight communication pathways and the time delay involved in such a communication. This will facilitate the future implementation of satellite to satellite (S-S) communications in support of constellation automation.

The host computer will also inject faults into the system to test the robustness of constellation control methods. These faults can range from something as benign as a temporary loss-of-signal to something much more serious like a stuck ACS actuator. Faults could also be include intentional interference in constellation operations by malicious actors. These "Blue-Gray-Red" scenarios are of use in developing both military and commercial space assets.

Towers. The primary unit of organization for the system is a set of 4 towers each containing 23-24 RPi boards as shown in Figs. 4 and 5. Each tower measures 170- by 150- by 800-millimeters. The final tower designs will include LEDs and text displays providing state of health (SoH) information about the operating state of the cluster and the simulated constellation.

Current tests have shown that the the RPi computers quickly reach their maximum recommended operating temperature and stay there for the duration of runs. A fan has been included to provide cooling for each tower, although further analysis and design will be required given the current results.

Power Distribution. The system will use 16 6-port, 60 W USB chargers to power the RPi computers. Tests have shown that while the RPi 3B can use up to 2.5 A of current, they draw a maximum of 1.2 A without any attached peripherals. A second 200 W computer power supply is used to power the CPU fans currently planned for cooling the towers.

Raspberry Pi Agents. The Raspberry Pi computers function as the satellite simulators. Each RPi has a 4-core ARM Cortex-A35 1.2 GHz processor, 1 GB of 900 MHz RAM, a 10/100 Mbps Ethernet connection, and access to a 32 GB microSD card for storage. The processor is roughly equivalent to some current mission-grade flight computers such as the Tyvak Intrepid and its descendants and the Innoflight CFC-300.[15] Access to processing resources, RAM, and SD card storage can be tailored to better approximate an actual satellite's capabilities on a mission-by-mission basis by the software wrapper.

In the proposed primary operating mode, one core of each RPi will run a version of the cFE or some other mission-specific code. The code will run inside a wrapper designed to provide the cFE with inputs as if it were in flight. The wrapper restricts the cFE's access to RAM and disk storage to what would exist on the satellite it is simulating. Any non-mission code can be logically separated from the cFE and run on a separate core within the RPi.

A second core is responsible for converting the noumenological data from the simulator into phenomenological data which is transmitted to the cFE. It is also responsible for emulating any onboard inertial measurement units to remove that load from the host computer. The IMU models will include the expected biases of the device, but to protect against unexpected drift in the simulation, its data will be saved to the RPi's SD card and transmitted to the host computer for analysis after the simulation.

This core will also convert actuator commands from the flight computer into noumenological actuator position, force, and moment values for the host computer and phenomenological measured actuator position values for the satellite's control systems. The noumenons are transmitted to the host computer for use in integrating the equations of motion while the phenomenons return to the flight computer to close the control loops. As this core is under a potentially large strain, its processes will have access to the fourth core should it be available.

The third core will be responsible for facilitating communication with the host computer and transmitting data to and from the sensor and actuator processes. This is the master simulation control for each RPi "node" and will enforce real-time operations. It will be responsible for caching messages that arrive too early from the host computer and monitoring time stamps to be sure they are transmitted to the cFE at an appropriate time. It will also send SoH messages to the master process on the host computer to determine if the cluster has developed an abnormal state and if its simulation results can be trusted.

The fourth core on the RPi will be held in reserve against potential future known and unknown needs. Known needs include potentially taking load from the sensor/actuator core and the inclusion of mission-specific hardware. For example, the extra core could serve as a separate ADCS processor if such exists on a satellite. It could serve a similar purpose if there were a desire to test a scientific instrument or optical sensor as part of the simulated satellite in constellation flight. The RPi's multiple GPIO connections increase its utility for this sort of application.

Alternate operational modes take the RPi computers and logically split them into two 2-core computers or four single core systems. This functionality allows the expansion of the simulator to accommodate a massive constellation of 188 or 376 satellites, although additional safeguards will be required for the separation of truth data from hardware as well as for handling network traffic.

Network. The expected load on the network includes the transmission of satellite telemetry packets ($<$1MB, $>$1Hz), truth and state updates for each satellite ($<$1MB, $\sim$1Hz), ground commands ($<$1 MB, $>$1Hz) and potential satellite-to-satellite (S-S) communications. Based on this estimate, commercial, off-the-shelf Gigabit switches are more than enough to handle the network traffic even if an increased update rate is required for monitoring an active maneuver. Additionally, timing algorithms are implemented so that the towers will send and expect their network traffic on a rolling basis, thus reducing the load on the network.

EXTERNAL HARDWARE INTERFACE

The Georgia Tech Space Systems Design Laboratory (SSDL) has access to several facilities and pieces of equipment which are targets for integration into the constellation simulator. The SSDL Mission Operations Center allows students, faculty and sponsors to monitor and control active missions directly from the Georgia Tech campus. The ORACLE facility is capable of testing optical instruments, such as sun sensors, but could also be used in the development of constellation visualization techniques. As the scale of satellite constellations increases, the amount of data will outgrow the ability of humans to track the information on traditional telemetry screens. Novel visualization and ground control techniques will need to be developed to assist human operators. Additionally, the SSDL has constructed a Helmholtz cage and owns a 2-D air bearing, both of which present interesting opportunities for HWIL integration.

Mission Operations Center

The SSDL Mission Operations Center is a 540 ft^2 facility located in the Engineering Science and Mechanics building at the Guggenheim School of Aerospace Engineering. The facility features desktop computer consoles for monitoring and controlling mission subsystems as well as a mission director workstation. The center includes a wireless voice network for ground team operations and access to data servers in an adjacent building. The Mission Operations Center can assist in the development of cognitive engineering solutions to the problems associated with managing a massive satellite constellation by presenting COSMoS data as if it were an active mission.

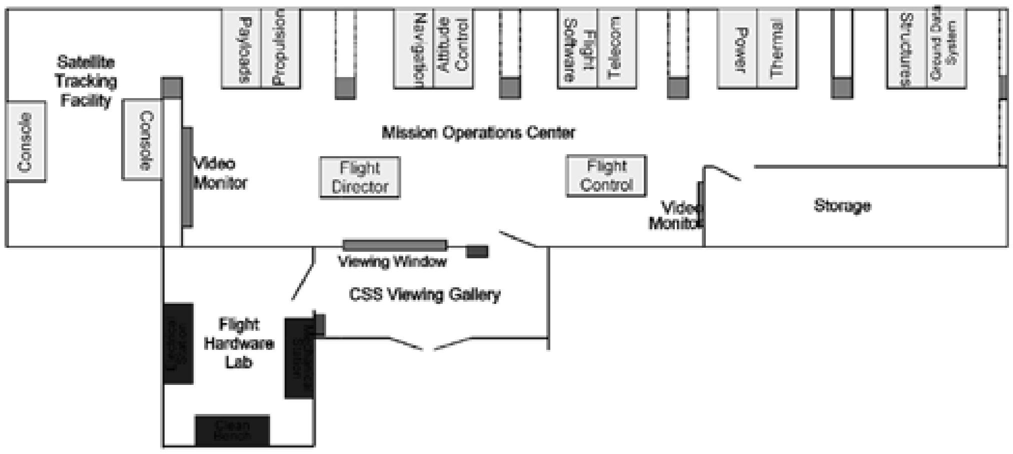

Figure 7. SSDL Mission Operations Center.

ORACLE

The Optical Real-time Algorithm Calibration and hardware in the Loop Experimentation (OR-ACLE) system was designed for the testing of optical instruments attached to the RECONSO and GT-SORT Space Situational Awareness (SSA) projects. The system is comprised of a 4K LCD projector with a 16- by 9-foot screen. The projector is capable of projecting 4096x2160 images at 30 Hz with full 16-bit color. It can project a similarly sized image at 60 Hz using interpolated color or monochrome. The laboratory windows are fitted with fully baffled blackout shutters to protect against any external interference with optical sensor tests.

This capability allows for the testing of optical sensors, such as Sun, Earth, and star trackers within the context of an operational mission. The COSMoS host computer could communicate with the ORACLE system to project the expected field of view of a sensor based on the sensor's position and orientation. This would include other satellites passing through a sensor's field of view to assist in the development of SSA technologies and algorithms. Given a high enough fidelity model, the simulation could also predict scattering, reflections and thruster exhaust effects on a sensor's field of view. A picture of the ORACLE lab can be seen in Figure 8.

Figure 8. ORACLE Laboratory.

Helmholtz Cage and Air Bearing

One of the newer additions to the SSDL toolkit is a large Helmholtz cage capable of housing a 6U cubesat. The cage's test section measures 0.5 m on each side. The coils are independently powered and each has a diameter of 1.5 m. Because the coils are independently powered, the cage can generate a dynamic field within the test section. The addition to an active controller to the Helmholtz cage would allow the simulator to actively set magnetic field conditions within the test section to represent the magnetic environment of a satellite on orbit. When combined with the 2-D

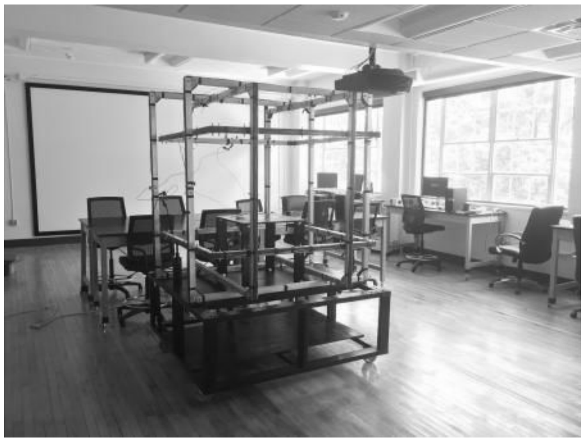

Figure 9. Georgia Tech Helmholtz Cage.

air bearing, this would allow for the testing of magnetic attitude control systems within a simulated operational environment. Rather than running through test patterns on an air bearing within a fixed magnetic field, a satellite mock-up connected to the simulator could experience the varying magnetic fields expected during a mission and execute station keeping or pointing maneuvers. This capability could add to the FSW TRL of a mission without flight testing.

CURRENT RESULTS

Current results from tests performed on the COSMoS system are promising although not without some developmental issues.

The initial system test was designed to stress the system beyond what is expected to occur in simulations. The test included a component designed to stress the RPi agents and a parallelized network communication task designed to stress the network and the host computer. RPi performance was tested by running sysbench prime number calculation threads on each RPi to determine cooling requirements and find potential upper limits for system performance. The sysbench tasks also functioned as a stand in for the computational load created by running flight software and actuator models on the agents.

The network communication test involved a simple counting task where the host computer generates a set of 24 packets with a data payload of the value “1”. Each agent which received a specifically addressed packet, incremented the payload, and transmitted the incremented payload back to the host computer. This task would continued until the payload reached a specified maximum value. It was considered to be a worst-case test for the network traffic because the counting task did not have any time delay requirements and required very little computation time.

Analysis of the results from four of these runs are presented below. The test parameters were the number of sysbench threads, maximum prime number computed by sysbench, total runtime for sysbench, minimum state of health update increment, and maximum counter value. The parameters for the four runs are shown in Table 1

Table 1. COSMoS System Test Parameters

	# Threads	Max Prime	Total runtime (s)	Update Increment (s)	Max Counter Value
Run 1	3	20000	2400	60	90000
Run 2	2	20000	2400	60	30000
Run 3	2	5000	2400	60	90000
Run 4	2	5000	2400	15	90000

Each of these tests was allowed to run until the sysbench task was terminated by exceeding its maximum time and the counter task was completed by reaching its maximum value on each agent. The data collected included both the agent state of health and the time at which a particular packet was incremented. Values for the time required between each increment step (Δt) were calculated from the increment timestamps. Plots showing the results of this run set can be found in the appendix.

Similar results were seen across all four runs, suggesting that the sysbench computational load did not substantially impact the system’s ability to process incoming and outgoing data packets. Across all runs, the range of maximum to minimum Δt values was generally around 1 second with a mean value of 110 ms and standard deviation of ≈ 15 ms. However, while no packets were lost during the test, some did experience Δt values greater than 1 second and as high as 6 seconds in one case. In a real-time environment operating at 1 Hz, these events would have caused real-time faults and some method for graceful handling will be required.

Additionally, some of the Δt values were recorded as negative. This was determined to be the result of poor time synchronization between the clocks on the agents and the host computer. While the time bias can be estimated using a batch or Kalman filter, other methods to address this problem are being considered such as using the host computer as a network time server for the agents.

Finally, the state of health packets showed that all agents quickly reached their maximum operating temperature of $\approx 82^oC$. Further experiments will be needed to determine how the RPis respond to reaching their maximum operating temperature. It is currently unclear if the computers throttle their performance to remain within operating specifications. Additionally, it is possible that FSW simulations will not create the same thermal loads as sysbench, and so further evaluation will be done to determine if the cooling system should be designed for such a high-use case.

FUTURE WORK

The results of the diagnostic tests have suggested further additions to the data logging and state of health monitoring subsystems of the simulation framework. Specifically, better time coordination between the host and agents will be necessary along with improved data collection to determine if processing bottlenecks are occurring at the network level or during computation. Further analysis of the thermal loads present on the system will also be necessary to determine how robust the cooling systems will need to be for long-term simulation use. The message passing interface will also be further optimized based on the results collected from the improved data logging to minimize transmission delays during real-time operations and protect against real-time faults.

The current simulator is only 1/4 the size of the planned system. Once the thermal properties of the tower are better understood, the tower case will be redesigned and three more will be constructed to house the remaining RPi agents.

After the system has been expanded to the originally planned MADNS scale, it will have four major testing goals.

1. Testing the Minimum Lyapunov Error approach to constellation control using a simplified flight code created using the cFE.

2. Testing of the Georgia Tech RECONSO satellite being developed under Dr. Marcus Holzinger. The simulator will be updated to interface with RECONSO's flight hardware and the ORACLE projector to present RECONSO's optical sensors with a simulated star and satellite field based on maneuvers calculated by its GNC algorithms.

3. Connecting the simulator to Georgia Tech's Mission Operations Center to facilitate the training of ground operators for RECONSO and other upcoming Georgia Tech satellites.

4. Incorporating simulator with the Mission Operations Center to begin the development and evaluation of ground support policies and procedures for massive scale satellite constellations.

These goals will be the targets of research and development extending into the future and will hopefully provide a fruitful testbed for new technologies related to satellite swarm and constellation development.

REFERENCES

[1] E. Douglass, M. Holzinger, McMahon, J.W., and A. Jaunzemis, "Formation Control Problems for Decentralized Spacecraft Systems," *Astrodynamics 2013: Advances in the Astronautical Sciences*, Vol. 150, 2013, pp. 1337–1356.

[2] C. Dillow, "Here's why small satellites are so big right now," *Fortune.com [online]*, 8 2015. Retrieved on 10/4/2016.

[3] M. Safyan, "Overview of the Planet Labs Constellation of Earth Imaging Satellites," ITU Symposium and Workshop on small satellite regulation and communication systems, Prague, Czech Republic, 2-4 March 2015, 2015.

[4] NASA, "NanoRacks-Planet Labs-Dove (NanoRacks-Planet Labs-Dove) - 07.20.16," 2016.

[5] Wikipedia, "Sirius Satellite Radio — Wikipedia, The Free Encyclopedia," 2016. [Online; accessed 5-October-2016].

[6] C. Kang and C. Davenport, "SpaceX founder files with government to provide Internet service from space," *The Washington Post [online]*, 7 2015. Retrieved on 9/28/2016.

[7] J. Foust, "The return of the satellite constellations," *The Space Review [online]*, 3 2015. Retrieved on 9/28/2016.

[8] S. Engelen, S. Eberhard, and C. Verhoeven, "Systems Engineering Challenges for Satellite Swarms," *IEEE Aerospace Conference*, 2011.

[9] V. Trianni, R. Gross, T. Labella, E. Sahin, and M. Dorigo, "Evolving Aggregation Behaviors in a Swarm of Robots," *Advances in Artificial Life. Ecal 2003. Lecture Notes in Computer Science.*, Vol. 2801, 2003.

[10] T. Schetter, m. Campbell, and D. Surka, "Multiple agent-based autonomy for satellite constellations," *Artificial Intelligence*, Vol. 145, 4 2003, pp. 147–180.

[11] K. Sobh, K. El-Ayat, F. Morcos, and A. El-Kadi, "Scalable Cloud-Based LEO Satellite Constellation Simulator," *World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering*, Vol. 9, No. 6, 2015, pp. 1467–1478.

[12] M. Bacic, "On hardware-in-the-loop simulation," *44th IEEE Conference on Decision and Control and the European Control Conference 2005*, 12 2005.

[13] I. Kant, *The Critique of Pure Reason*. 1781. Retrieved from Project Gutenberg.

[14] G. S. F. C. I. P. P. Office, "Core Flight Executive (cFE)," 2016.

[15] I. Inc., "Compact Flight Computer," 2016.

APPENDIX: NETWORK AND TEMPERATURE TEST RESULTS

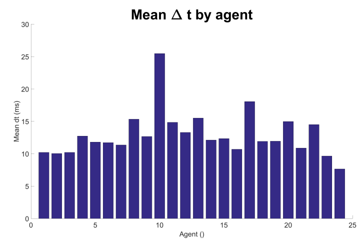

Figure 10. Run 1 Mean Δt.

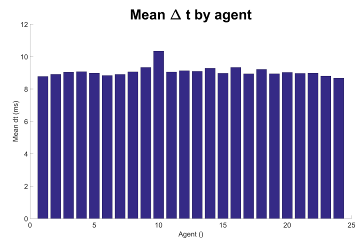

Figure 11. Run 2 Mean Δt.

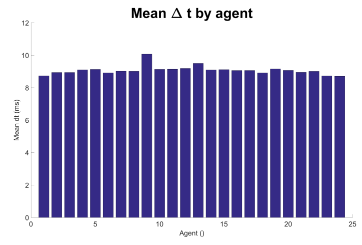

Figure 12. Run 3 Mean Δt.

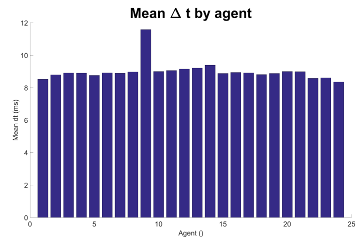

Figure 13. Run 4 Mean Δt.

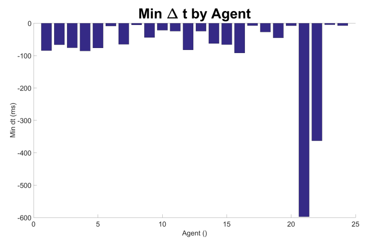

Figure 14. Run 1 Min Δt.

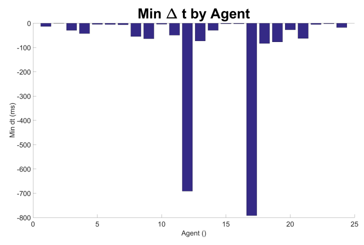

Figure 15. Run 2 Min Δt.

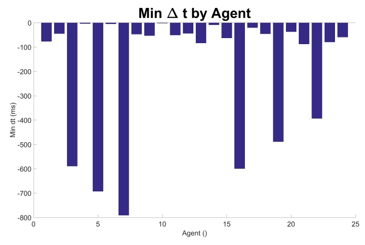

Figure 16. Run 3 Min Δt.

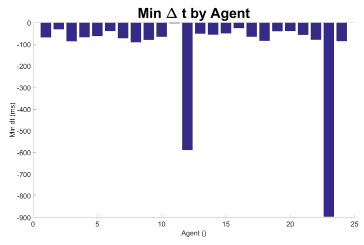

Figure 17. Run 4 Min Δt.

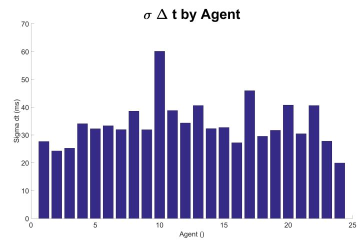

Figure 18. Run 1 σ of Δt.

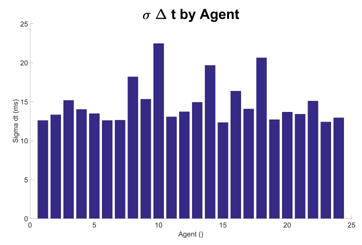

Figure 19. Run 2 σ of Δt.

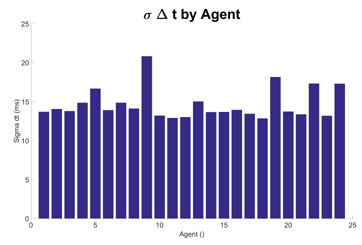

Figure 20. Run 3 σ of Δt.

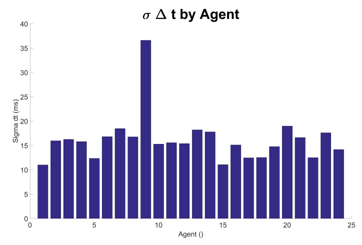

Figure 21. Run 4 σ of Δt.

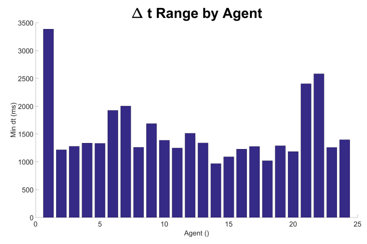

Figure 22. Run 1 Δt Range.

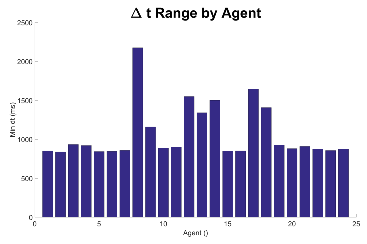

Figure 23. Run 2 Δt Range.

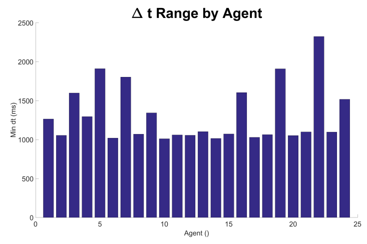

Figure 24. Run 3 Δt Range.

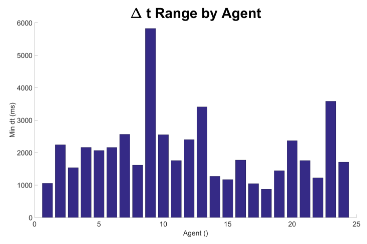

Figure 25. Run 4 Δt Range.

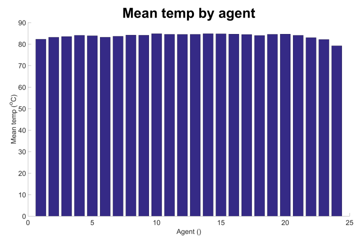

Figure 26. Run 1 Mean Temperature.

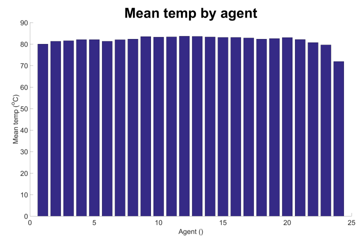

Figure 27. Run 2 Mean Temperature.

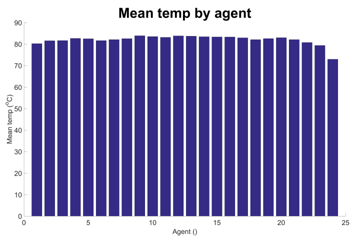

Figure 28. Run 3 Mean Temperature.

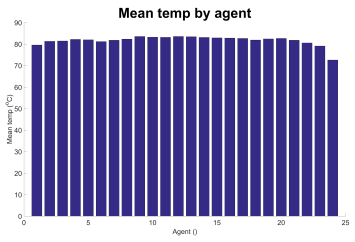

Figure 29. Run 4 Mean Temperature.