A Gaming Environment for Resilient Network Design and Adversarial Co-Evolution Modeling

Marco Carvalho, Adrian Granados, James McLane, and Evan Stoner
Intelligent Communication and Information Systems Laboratory (ICIS)
Department of Computer Sciences, Florida Institute of Technology, Melbourne, Florida, USA
Email: mcarvalho@fit.edu,{agranados, jmclane, estoner2010}@my.fit.edu

Abstract—**The design and evaluation of resilient communication and computation infrastructures still relies heaily on empirical studies, and are often based on advanced simulation or emulation techniques that include external attacks or specific failure modes. While generally useful to provide an initial baseline, such static approaches fail to take into account the adaptive nature of the attacker. Recognizing the co-evolutionary nature of the attacker, we argue that, especially in the case of adversarial environments, an intelligent and adaptive attacker model must be considered for the test and evaluation of defense infrastructures. This paper describes the implementation of an adversarial gaming environment for the development, test, and evaluation of Resilience Network and Electronic Warfare procedures. The motivation for the gaming approach is based on the concept of attack co-evolution, and red teaming. In this work, we propose that serious games can be used to engage humans in high-fidelity adversarial simulations that will allow us to capture, and possibly model the co-evolutionary behavior of adversaries, and will provide the basis for the design of online electronic defense mechanisms that will take such effects into account.**

Index Terms—**electronic warfare emulation, adversary co-evolution, serious games, EMANE, VIA**

I. Introduction

It is generally accepted today that resilient mission-crucial systems and applications are likely to rely on some kind of automated way to identify, respond and possibly recover from faults or attacks.

In the last several years, a number of research efforts have focused on the design of adaptive, or autonomic defense mechanisms implementing a combination of properties for system resilience and defense. In wireless and tactical communication environments, for example, the research community has made significant advances in building robust, and self-regulating algorithms and protocols for network management and application support. More recently, self-management capabilities have also made their entrance in the electronic warfare (EW) arena, where automated detection, defense and adaptation mechanism started to appear as effective and timely tools for attack mitigation.

The common practice in distributed protocol design and evaluation still relies heavily on simulation [1]-[5] (or emulation [6], [7]) tests and experiments. This is largely due to the complexity and the distributed nature of the problem, as well as the many inter-dependencies associated with the timing and strata transitions of distributed protocols. Experimental tests allow for a reasonable reconstruction of normal operational conditions of large distributed systems, and the introduction of controlled and localized failures.

Numerous high-fidelity emulation (and simulation) infrastructures have been built or proposed for the development, test and validation of distributed systems, protocols and coordination algorithms; they generally constitute some of the most important tools in the arsenal of scientists and technology developers in this domain. It is often through extensive tests and experimentations, normally including simulations, emulations and field tests, that the majority of self-adaptive defense mechanisms are evaluated and characterized.

One of the limitations of this approach is that the emulation/simulation environments used for experimentation generally rely on the accuracy of the simulation, as well as the pre-establishment of background activity, failures, and malicious activities [8], [9]. Most of the emulation/simulation-based experiments today focus on providing a realistic communication and computation infrastructure that is consistent with a" ground truth", and to evaluate the effectiveness of adaptive algorithms against such scenarios.

In practice, however, the evolution of individual players in the system is more complicated. The background activities, as well as attacks, are themselves a function of the response mechanisms developed for the protection of the system.

Moreover, one of the main difficulties associated with the realistic emulation of EW environments is the consideration of the dynamics of the adversaries monitoring and controlling the electronic resources. The effects of attacker co-evolution, that is the correlated changes in attacker strategy based on the engagement of defense mechanisms, has just recently started to be considered in adversarial settings.

Most approaches, so far, have relied on game-theoretic models of adversaries to anticipate their actions and strategies based on the evolution of the game [10]. These approaches are largely bias to the assumptions of rationality and cost functions attributed to adversaries, which has limited the applicability of these kinds of

Manuscript received June29, 2013; revised August 26, 2013.

formations for the selection of electronic protection strategies. In this work, we propose that serious games can be used to engage human in high-fidelity adversarial simulations that will allow us to capture, and possibly model the co-evolution of behavior, and will provide the basis for the design on online electronic defense mechanisms that will take such effects into account.

II. The Electronic Warfare Environment

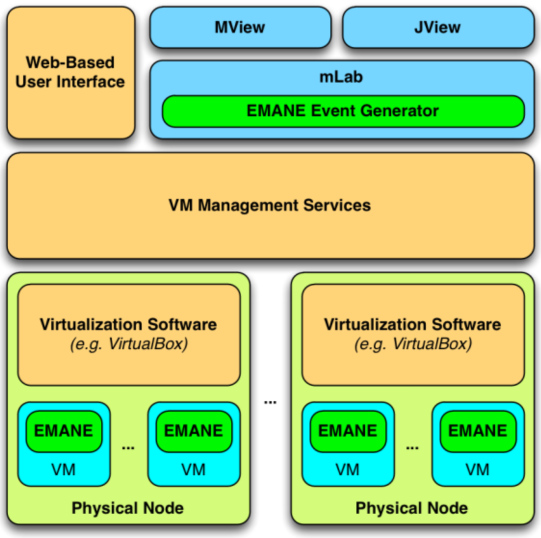

Figure 1. Virtualized infrastructure for network emulation (VINE)

We have modified and extended a multi-player game to allow for the simulation of EW battlefield scenarios. The game has been fully integrated with VINE (Fig. 1), a virtualized infrastructure for network emulation, and a cross-layer framework for tactical networks called VIA. This integration (Fig. 2) enables the simulation of wireless links, including path disruption and interference, based on the virtual position and radio settings (e.g. transmit power, frequency) of the game units and different propagation models commonly used in wireless network simulation environments, such as Free-Space, Two-Ray Ground, and Shadowing.

VINE is an infrastructure that allows for automatically creating and deploying virtual machines (VM) over a set of physical computers. VINE uses VirtualBox as the virtualization software, but it has been designed to easily adapt other virtualization packages, such as VMware. It is fully integrated with mLab [11], [12], a hybrid emulation environment for airborne networks that uses theoretical and statistical data-driven propagation models and the NRL's Extendable Mobile Ad-hoc Network Emulator (EMANE) [13] to enforce wireless link characterization at the physical layer. The mLab controller communicates with EMANE using the EMANE's event mechanism to update transmit power, position and path-loss properties. Each event elicits the appropriate response from the system to emulate network and link conditions based on EMANE's physical and propagation models.

The game is originally based on a popular open-source 3D Real-Time Strategy (RTS) engine called Spring (Fig. 3) Spring uses the Lua programming language for scripting game-specific code, making almost every aspect of the game customizable. The gaming engine provides fully 3D combat scenarios in land, sea, and air with realistic weapon trajectories and large, highly detailed maps with different geographical features, such as deformable terrain, forests, and bodies of water.

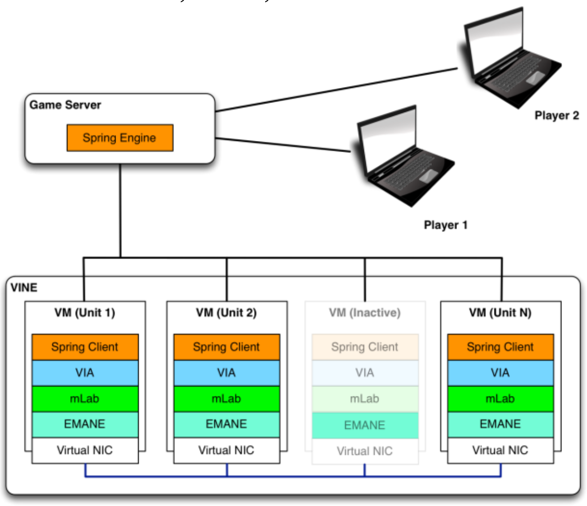

Figure 2. Integration of the EW emulation environment and VINE/VIA

As part of the integration, we have inserted the necessary hooks to intercept and transmit every message exchanged between game units over a simulated wireless communication infrastructure using VIA [14], [15]. Thus, game units within communication range can effectively exchange messages and coordinate mission specific tasks. On the other hand, game units that are located outside of the range of each other transmitters cannot directly communicate and must rely on ad-hoc routing protocols and multi-hop packet forwarding, as it is the case in real mobile network scenarios.

Game units include command centers, towers, vehicles equipped with jammers, UAVs, and ground troops. New units are automatically instantiated by the system as virtual machines (VM) in the VINE environment. Each unit is mapped to the VM's VIA service to join the game. At any moment, a player can effectively disrupt communications by shooting and destroying enemy units or by launching electronic attacks (e.g. jamming) to interfere with ongoing communications.

Because each game unit is mapped to an instance of VIA, and since every aspect of the communications network is managed by the cross-layer framework, we can modify the behavior of the unit by extending VIA to implement different coordination and topology control algorithms. Moreover, because VIA abstracts the underlying physical network interfaces, we can also map a game unit to a real platform, for example, a remote controlled car playing the role of a ground vehicle in the battlefield, or a quad-copter acting as a UAV in the game environment. When a unit mapped to a real platform moves within the game environment, VIA translates the virtual position into a physical position (within certain boundaries) and instructs the platform to move. Communication between two real platforms does not need to be emulated and is performed using their physical radios. On the other hand, communication between a real platform and a simulated game unit requires of a control channel over which data packets are transmitted. The control channel ensures that data packets are correctly managed and deliver based on the link characteristics imposed by the position and radio settings of the real and simulated game units.

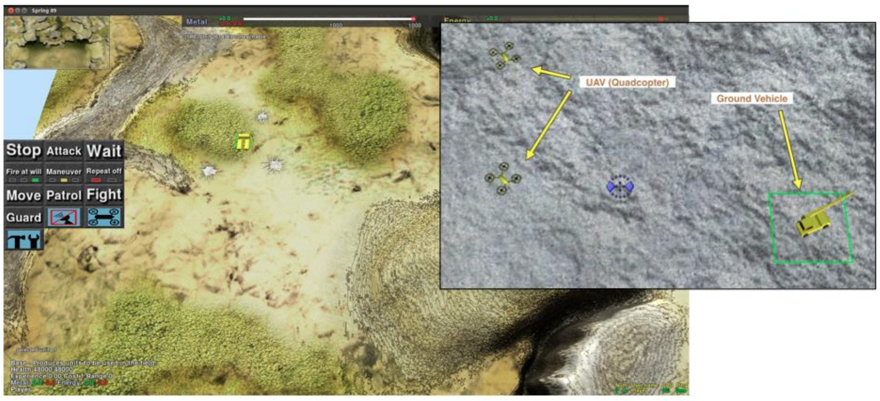

Figure 3. The 3D Real-Time Strategy (RTS) game environment

This integration allows for more realistic emulations enabling the evaluation, in real-time, of field deployments mixing simulated with real game units, both obeying the commands and following the strategies of human players. The infrastructure and game environment have shown promising capabilities for the design, development and evaluation of airborne network protocols for EW and mission-oriented scenarios.

III. Implementation

The EW environment consists of several components, including the required components to integrate the game environment with the emulation infrastructure. The integration of game units with VINE is implemented by the use of the VIA cross-layer framework. VIA (Fig. 4) Works by creating a virtual network interface that manages traffic over multiple physical network interfaces. New and existing applications that bind to the virtual interface (e.g. *via0*) can transparently make use of the framework capabilities. Furthermore, one of the most important advantages of VIA is that it operates below the network layer to transparently manage protocol packets such as ARP and IP, including ICMP, TCP and UDP. Hence, unlike other solutions, VIA does not require encapsulation and existing applications can take advantage of VIA without any modifications. Also, non-VIA and VIA nodes can transparently interoperate. In addition, VIA supports dynamic loading of modules (dynamically loaded libraries) that enables third-party developers to extend the framework's core capabilities with new functions for generation and processing of VIA control packets, information sharing, application packet filtering, and much more. All of these capabilities make VIA an ideal framework for the implementation, testing, and evaluation of protocols for communication, resilience, adaptation and defense within the EW environment, and since VIA abstracts the communications infrastructure, protocols rarely need to change when ported to real environments.

Inside the game engine, a collection of models, graphics, and scripts define the gameplay. Users play the game by interacting with instances of the models called units (a command base, a truck, etc.). One of the scripts, the Remote Control Server, acts as a listener for all incoming messages. The script cannot make changes to the game directly, so it relies on another script called the Remote Control Gadget, which acts as the interface between the control server and game units. Its two main functions are to intercept commands the user gives to units, and put successfully routed commands back into the unit's command queue.

A third script, the Remote Control Client, represents a game unit and runs in the virtual machine. The control client processes commands that the control server issues, and interacts with VIA to parameterize the emulation infrastructure implemented by the Virtual Topology Control (VTC) module.

Because in the game all commands are sent across an emulated network, users may only issue commands to units in the game if they have the infrastructure in the game to support the transmission of that command. In addition, all commands must be issued from a stationary command base, called Command and Control (CC); however, communication between units for information sharing and coordination purposes is also possible by using the mechanisms provided by the underlying cross-layer framework.

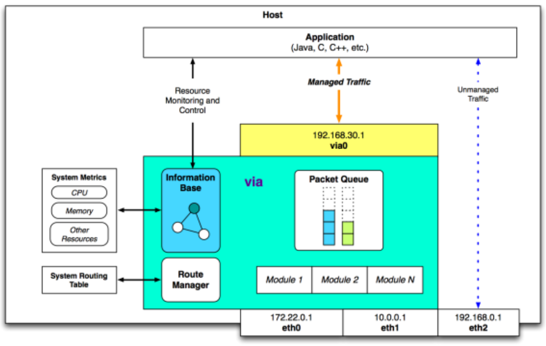

Figure 4. Architecture of the VIA cross-layer framework

The first virtual machine that is made available to the game is assigned the role of the CC unit. Once the CC is ready to operate, the users may create additional game units at the CC node. From the game interface perspective, this action essentially consists on selecting a specific unit (e.g. soldier, UAV, or a jammer platform) from a panel of available unit types. When the selection occurs, a command is issued by the CC that triggers the instantiation of a new virtual machine in VINE. When the VM starts, it broadcasts its presence and supplies the physical network interface and VIA IP configuration information to the game engine. Then, the control server chooses the new unit and maps the unit to the new VM.

At this point, communication between the new unit and other existing units in the game is possible according to the state of the emulation environment. When a player issues a command to one of his units, the control gadget script intercepts the command and supplies its information to the server, which relies the message to the CC VM to be transmitted using the UDP protocol over the emulated network.

The transmission, routing and handling of the message by the receiver node happens transparently from the point of view of the game engine. If the target node is in range (according to the VTC module in VIA), the message is sent back to the control server script and the command is inserted into the unit's command queue to be processed by the game engine as usual.

To maintain the virtual topology synchronized with the location of the game units, the control server sends periodic messages to VIA containing position information so that VIA can update the virtual position attribute used by VTC. When the node's position changes, the VTC module is automatically notified to adjust the characteristics of the links with neighbor nodes following the current propagation model.

If a node becomes out of range, VTC drops any incoming traffic from it. The effects of the packet loss are then propagated to the upper layers triggering route changes to enable alternative communication paths, as it real mobile ad-hoc networks. If the unit becomes completely disconnected from the network, it is hidden from the game since the CC has loss track of the unit, as it would occur in a real scenario.

When the unit becomes in range of the CC or any other friendly unit, the unit reappears and the player can now send new commands to it. Similarly, if a unit is destroyed by an enemy attack, the VM is stopped and released. As a result, the node is removed from the virtual topology and, as expected, communication with the unit is not longer possible.

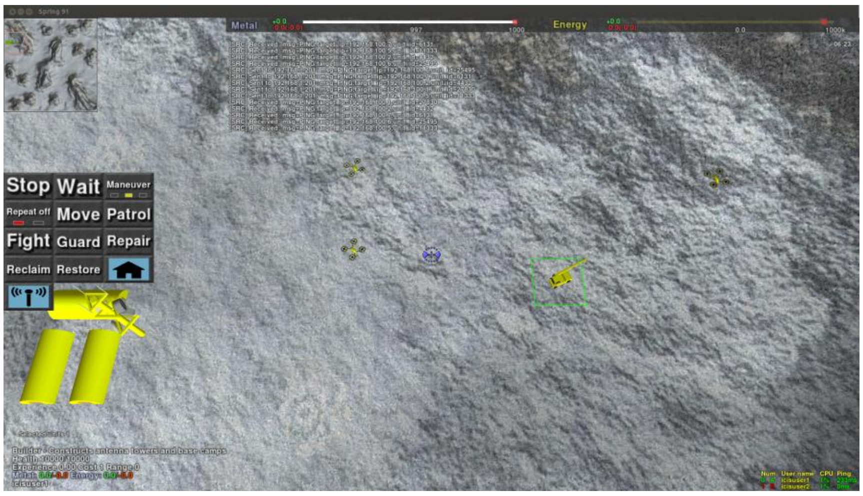

Figure 5. Interactions of active units within the EW environment

IV. EW Environment and Robotic Testbed Integration

We have integrated the EW environment with Dominion (Fig. 6), an experimental robotic indoors testbed used for the design, test and evaluation of hardware-in-the-loop multi-agent infrastructure. In our current integrations with the EW game, we have used six Parrot AR.Drone 2.0 radio-controlled quadcopters as well as robotic platforms.

The AR.Drone quadcopters have several motion sensors and feature a miniaturized inertial measurement unit (IMU), ultrasound altimeter, two cameras (front and bottom), Wi-Fi b/g/n, USB 2.0 port, and a 1GHz 32 bit ARM Cortex A8 processor with 1GB DDR2 RAM running a Linux 2.6.32.

The robotic platforms consist of remote-controlled cars with on-board Raspberry Pi computer platforms and Wi-Fi dongles for wireless communication with airborne and ground nodes. We have also used COTS wireless routers to act as fixed game units (e.g. towers), such as the ASUS WL-500G running a custom firmware based on the OpenWRT Linux embedded operating system.

To enable the evaluation and experimentation capabilities, the quadcopters have been modified to establish a common ad-hoc network for resource information sharing and coordination using VIA. VIA also provides the necessary mechanisms for autonomous control of the quadcopters, as well as from external controller applications, facilitating the integration of protocols for coordination and navigation. VIA is also installed on the Raspberry Pi to provide the same capabilities for coordination and resource information sharing available to airborne and ground nodes.

In order to perform indoor flights, we have developed a localization system for indoor environments using image recognition to identify and track each object as it moves within a predefined, limited testing area. However, since the area used for indoor experimentation is often too small and the transmit power of the wireless adapters cannot be modified or controlled, we rely on VIA's virtual topology mechanisms to limit, through emulation, the range of the unit's radio transmitters.

The integration also uses VIA as the bridge between the real world and the virtualized environments where realistic network conditions such as link and path disruptions and interference are emulated. The deployment and management of the communications infrastructure supports command centers, towers, jamming vehicles, UAVs and ground troops.

As a result, real world airborne and ground nodes have a representation in the simulated environment, and by using a control channel and the VTC module VIA enforces link characterization by sharing information bout the node position and its transmission power. Packets sent from real nodes are captured and sent to the game server, where VIA re-routes them based on actual topology information collected via the control channel.

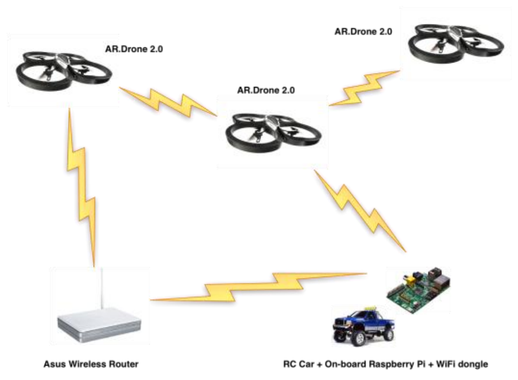

Figure 6. The dominion experimental robotic environment

This integration takes advantage of VINE for larger scaled simulations, while enabling the evaluation, in real time, of field deployments. Some of the implementation details are still a work in progress, but the infrastructure shows promising capabilities for the design, development and evaluation of airborne network protocols for mission-oriented scenarios.

V. Conclusions

In this work we introduce the preliminary design and implementation of an adversarial game for the study of resilient networks and electronic warfare. The proposed approach relies on human-interfaces to evaluate the effects of attacker co-evolution in adaptive defense strategies.

While proposed and implemented for an adversarial battlefield gaming environment, the approach should be extensible to more complex applications such as cyber defense, and critical infrastructure protection (cyber-physical systems). The emulation infrastructure is based on the high-fidelity EMANE network emulator (NRL), which was integrated with the virtualization environment (VINE) developed at Intelligent Communication and Information Systems Laboratory, at the Florida Institute of Technology.

References

[1] The ns-3 Network Simulator. (2009). [Online]. Available: http://www.nsnam.org

[2] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "GloMoSim: A scalable network simulation environment," Technical Report 990027, UCLA, Computer Science Department, 1999.

[3] D. Kotz, C. Newport, R. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," in *Proc. 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2004, pp. 78–82.

[4] S. Kurkowski, T. Camp, and M. Colagrosso, "Manet simulation studies: The incredibles," *SIGMobile Mobile Computing Comm. Rev.*, vol. 9, no. 4, 2005, pp. 50–61.

[5] G. Setiwan, S. Iskander, Q. Chen, S. Kanhere, and K. Lan, "The effect of radio models on vehicular network simulations," in *Proc. 14th World Congress on Intelligent Transport Systems*, Beijing, China, 2007, pp. 1-10.

[6] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim, "Core: A real-time network emulator," in *Proc. IEEE Military Communications Conference*, 2008, pp. 1–7.

[7] W. Chao, J. Macker, and J. Weston, "NRL mobile network emulator," NRL Formal Report, NRL/FR/5523-03-10,054, January 2003.

[8] CenGen Labs. (2010). EMANE User Training Workbook 0.6.2. [Online]. Available: http://downloads.pf.itd.nrl.navy.mil/emane/archive/training/emaneusertraining-0.6.2.20100301-1.pdf

[9] K. Pawlikowski, H. D. Jeong, and J. S. Lee, "On credibility of simulation studies of telecommunication networks," *IEEE Communications Magazine*, vol. 40, no. 1, pp.132 –139, January 2002.

[10] S. G. Ficici and J. B. Pollack, "A game-theoretic approach to the simple co-evolutionary algorithm," in *Proc. Sixth International Conference on Parallel Problem Solving from Nature*, Springer, 2000, pp. 467–476.

[11] M. Carvalho, A. Granados, M. Arguedas, M. Muccio, *et al.*, "The mLab-PENGWUN hybrid emulation environment for airborne networks," in *Proc. Military Communications Conference*, November 2011.

[12] M. Carvalho, A. Granados, C. Perez, M. Arguedas, *et al.*, "A hybrid emulation environment for airborne wireless networks," in *Advances in Intelligent Modeling and Simulation*, Studies in Computational Intelligence, A. Byrski, Z. Oplatkova, M. Carvalho, and M. Kisiel- Dorohinicki, Eds., vol. 416, Springer Berlin / Heidelberg, 2012, pp. 111–129.

[13] CenGen Labs. (2010). EMANE - Extendable Mobile Ad-hoc Network Emulator. [Online]. Available: http://labs.cengen.com/emane/

[14] M. Carvalho, A. Granados, K. Usbeck, J. Loyall, M. Gillen, *et al.*, "Integrated information and network management for end-to-end quality of service," in *Proc. Military Communications Conference*, 2011.

[15] M. Carvalho, "Xlayer: A cross-layer communications substrate for tactical environments," Tech. Rep. AFRL-RI-RS-TR-2010-127, Air Force Research Laboratory, Information Directorate, Rome, N.Y., June 2012.

Dr. Marco Carvalho is a Research Scientist at the Florida Institute for Human and Machine Cognition (IHMC), and a Graduate Faculty at the Florida Institute of Technology. He received his Ph.D. from Tulane University, following a M.Sc. in Computer Science from the University of West Florida, a M.Sc. in Mechanical Engineering from the Federal University of Brasilia (UnB), and a B.Sc. in Mechanical Engineering, also from UnB. Dr. Carvalho is a faculty member of the Center for Applied Optimization (CAO) at the University of Florida. He is also a permanent member of the editorial board of the Scientia Magazine (ISBN 0104-1770), a member of the editorial board for IEEE Transactions on Systems, Man and Cybernetics: Part B, and the organizer of several conferences and workshops in the areas of distributed systems resilience, smart grids, and biologically inspired resilience. His primary research interests are in the areas of tactical networks and information management systems, cognitive MANETs, cyber security, and critical infrastructure protection.

Adrian Granados is as a Research Associate at the Intelligent Communication and Information Systems Laboratory of the Florida Institute of Technology in Melbourne, FL. He has a B.Sc. and M.Sc. in Computer Science, and his work involves the design, implementation and support of emulation environments and infrastructures for MANETs and airborne wireless networks.

James McLane is currently pursuing his Masters' degree in Software Engineering at the Florida Institute of Technology, where he is researching multi-agent, decentralized coordination.

Evan Stoner is currently pursuing his Masters' degree in Software Engineering at the Florida Institute of Technology, where he is researching cyber emulation and visualization.