INTEGRATION OF FIELDED ARMY AVIATION SIMULATORS WITH MODSAF: THE EIGHTH ARMY TRAINING SOLUTION

Joseph M. Sardella and Darryl L. High
L-3 Communications Corporation, Link Simulation and Training Division
Binghamton, New York

Abstract

The Eighth United States Army (EUSA) in the Republic of Korea employs UH-60 and CH-47 flight simulators to support individual and crew training for Blackhawk and Chinook pilots, respectively. These simulators are high fidelity, man-in-the-loop, training devices that support initial entry, qualification, and sustainment training in system operations, crew coordination, emergency procedures, and combat skills. As part of the EUSA Korean Simulator Upgrade program, the two flight simulators are receiving an upgrade to the visual image generation system (including a geo-specific database of the Korean Peninsula) while maintaining, as a minimum, existing performance capabilities. One of the key training areas to maintain was the tactical environment. In the existing visual database, target sites and pathways were modeled into the database manually, based on training requirements and customer inputs, using custom database generation tools. The sites and paths, along with the behaviors of these targets, were under instructor controls; thus, providing numerous, realistic, dynamic, yet deterministic and repeatable tactical scenarios. In addition to these real-time scenarios, both training devices provide a reset and playback capability that allows the student and instructor to review the mission and allows fly-out to real-time at any time during the playback. Under the scope of the contract, these capabilities were to be maintained.

The solution needed to be a constructive simulation that not only maintained previous tactical environment fidelity (critical to each helicopter's training environment) but one that added enough robustness to provide a set of routes that can be altered as training requirements change without requiring a large database modeling effort. With an off-line scenario generation capability and realistic target movement models, Modular Semi-Automated Forces (ModSAF) was selected as the constructive simulation. By adding a Distributed Interactive Simulation (DIS) network interface between the legacy device and ModSAF, the Instructor Operator Station (IOS) at the training device can control each ModSAF target as directed by the existing tactics within the legacy training device. The use of DIS as the interface also provides future growth potential for the devices to perform collective training in a DIS or High Level Architecture (HLA) networked environment.

Biographical Sketches:

Joseph M. Sardella is a Principal Software Engineer with L-3 Communications Corporation, Link Simulation and Training Division, in Binghamton, NY with over 16 years of visual system, computer systems, and simulator experience. He has previously worked on several distributed simulation systems including the B-2 HLA program and the CELLNET program which integrated AH-64 and UH-60 flight simulators with Suppressor using DIS technology. Joe is currently the Integrated Product Team (IPT) Lead for the Tactics/Visual IPT on the EUSA program.

Darryl L. High is a Software Engineer with L-3 Communications Corporation, Link Simulation and Training Division, in Binghamton, NY with experience predominantly in the area of Computer Generated Forces (CGFs). Having contributed to programs such as B-2 and the AH-1W, Darryl is now working on integrating ModSAF, using DIS technology, with flight simulation trainers on the EUSA program.

INTEGRATION OF FIELDED ARMY AVIATION SIMULATORS WITH MODSAF: THE EIGHTH ARMY TRAINING SOLUTION

Joseph M. Sardella and Darryl L. High
L-3 Communications Corporation, Link Simulation and Training Division
Binghamton, New York

INTRODUCTION

During times when military budgets require an ever-increasing return on investment, engineering solutions need to achieve the same aggressive return on investment as they balance reusability, innovation, and new technology. This return on investment equilibrium was sought on the EUSA program with an effort to upgrade to the capabilities of new image generation technology. The EUSA upgrade focussed on the installation of a new image generation system for legacy UH-60 Blackhawk and CH-47 Chinook aircrew training devices. This upgrade also promoted the safeguard of existing critical training capabilities such as trainer specific tactical behaviors, the ability to place threats anywhere within the tactical environment, record/playback functionality, and the ability to "fly-out" of a playback and return to real-time flight. The engineering solution, in turn, strived to achieve a similar equilibrium with an effort to balance reusability by keeping existing tactical environment functionality, innovation by designing a DIS interface to command and control constructive simulation entities, and new technology by instituting a new image generation system with a geo-specific database. In order to meet contract requirements and position for potential future growth in a distributive network, ModSAF was chosen as the constructive simulation to provide the off-line scenario generation and real-time target movement. The system design allows the existing IOS to control ModSAF entities using DIS protocol. This paper will summarize the current tactical capabilities that were retained, the modifications made to the off-line and real-time ModSAF functionality, and the DIS interface that was established for communication between the flight simulator's host computer (i.e. the host) and ModSAF.

EXISTING TACTICAL ENVIRONMENT

In general, the differences between the UH-60 and CH-47 simulators' tactical environment varied only in terms of the number of targets allowed in the threat environment, their interactions with the ownship, and the level of control that a training instructor had during a training mission. The environments were analogous and each flight simulator had the capability to create tactical scenarios at its IOS. Tactical missions were created by the training instructor via selecting targets from a predefined list and placing them in the environment at predefined sites and paths. These were only available for selection since they were included as part of the off-line database generation process. The creation of these sites and paths was a labor-intensive task and did not allow real-time modifications once created. Sites were defined in two categories: fixed and moving. A fixed site allowed a target to be placed at a predefined location within the database; however, the site could not be relocated during a real-time training mission. As the name suggests, targets placed on these sites did not move through the database, yet they still interacted with the ownship based on tactical behavior selected at the flight simulator's IOS. Moving sites consisted of multiple pathways providing a means for targets to traverse through the database with a predefined knowledge of the terrain and its features, A large number of sites and paths were created during the database creation process. to provide the instructor with the ability to create a variety of training scenarios.

During a training mission, the instructor could insert targets at the predefined database sites or paths, modify the locations of targets (limited to the predefined sites or paths), remove targets, and save the scenario for use at a later time. When moving targets were inserted into a scenario, they were placed at the beginning of a path. During real-time simulation, movement along that path was handled internally by the host and moved according to instructor controls and tactical behaviors. Examples of instructor controls are: commanding an air target to fly either Nap-Of-the-Earth (NOE) or at a fixed altitude above Mean Sea Level (MSL), instructing a moving target to change its commanded route speed, and having a target reverse direction and proceed back to the beginning of its path. An example of the tactical behaviors would be once a moving target reaches either the start or the end of its path, it would automatically come to a stop. While crude by today's standards, this was exceptional for the computing resources of the late 1980s when it was developed.

Another common feature that is an integral part of training is the record/playback operation. During a training mission, critical data is recorded in order to replay the mission for pilot and instructor review. This data not only includes flight performance of the pilot trainee but the entire tactical environment and its interactions with the ownship. While in playback mode, the training instructor can elect to have the pilot "fly-out" of the playback and return to live flight. When this option is selected, the tactical environment returns to live interactions with the ownship. Preserving this capacity during the EUSA upgrade would preclude the simple approach of logging and replaying an external constructive simulation's activities. Any external tactical environment would need to remain live, even during the playback of a mission, in order to allow for "fly-out."

OPERATIONAL OBJECTIVES

With the addition of the Korean geo-specific database to complement the new image generation system for the UH-60 and CH-47 simulators, it was desirable to provide a set of routes that can be altered as training requirements change without requiring a large database modeling effort. The EUSA solution uses a constructive simulation to control individual targets along selected routes, based on instructor inputs, and provide realistic movement models throughout the database. A high correlation between the visual database and the constructive simulation's database has to be ensured to provide realistic interaction between the training device and the tactical targets. However, with such a correlation, the deterministic and repeatable aspects of legacy scenarios could be maintained by adding a DIS interface between the constructive simulation and the host. This allows the training device's IOS and existing tactical behaviors to maintain control over the targets in the threat environment and concurrently allow delegation of target placement and movement to be placed in the hands of the constructive simulation.

ARCHITECTURE AND DESIGN

With an off-line scenario generation capability, realistic target movement models, DIS protocol compliance (i.e. a foundation for any future effort to become HLA compliant), and a key position in the Army's migration towards synthetic battle-spaces (i.e. ModSAF and OneSAF), ModSAF was selected as EUSA's constructive simulation. After reviewing the existing capabilities that needed to be retained and determining the composition of a platform needed for future growth, the following arrangement resulted: ModSAF 5.0 operating under a Linux Operating System (OS) version 6.1 on an Intel Pentium Personal Computer (PC). This system would then be integrated with the current training devices for real-time control over target movement. The host and ModSAF would communicate via an Ethernet connection using Institute of Electrical and Electronics Engineers (IEEE) 1278 standard DIS version 2.0.4 Protocol Data Units (PDUs). Existing ModSAF DIS capability would need to be customized in order to provide the needed DIS standard interface. This meant that *CreateEntity*, *RemoveEntity*, *StopFreeze*, *StartResume*, and *SetData* PDUs would be used by the host to communicate instructions to ModSAF while *Data* and *Acknowledge* PDUs would be used by ModSAF to properly confirm receipt of PDUs from the host.

Off-line Architecture and Design

Off-line, ModSAF's scenario generation tools would be used to create sites and paths within the new geo-specific Korean database. In order to maintain a similar capability with the existing system, only the creation of fixed sites and moving target pathways would be saved in a ModSAF scenario. This means that ModSAF will essentially provide the same off-line capabilities and limitations as the former system: a scenario file will be generated, for the IOS and host, to provide a menu of selectable routes for mission creation during real-time operation. The main difference is that unlike the previous system, a scenario file can be updated at an off-line ModSAF station any time training requirements change. Under the present contract, real-time modification of moving-target routes is not allowed. However, targets placed at a fixed site can be relocated anywhere within the database.

Real-time Architecture and Design

During real-time, tactical behaviors and target weapon fires will remain within the host. Therefore, this functionality will be disabled within ModSAF. Since ModSAF will take over responsibility for target movement, the target movement feature will be removed from the host. To consummate the integration of ModSAF, the host, and its IOS, a Network Interface Unit (NIU) will be added to the host to control the interface with ModSAF. The NIU will convert instructor commands for target control into DIS PDUs and transmit them to ModSAF as well as receive and process any ModSAF acknowledgment PDUs. As ModSAF controls the movement of targets, it will send each target's location to the host via *EntityState* PDUs. Upon receiving a target's location and orientation, the NIU will convert to the host-specific coordinate system and perform any necessary extrapolation so as to provide smooth and continuous movement at a 60 Hz update rate to the visual system and other host applications. Conversely, in order to maintain the highly correlated interaction with the ownship, the host will retain control over entities such as ships and slingloads. In these cases, the NIU will perform any essential dead reckoning, convert the internal entity data into EntityState DIS PDUs, and broadcast the PDUs to ModSAF so that ModSAF and its moving threats will be aware of and avoid collisions with these entities.

The NIU will isolate the host applications from the ModSAF interface by maintaining a ground truth database, where all entity data will be kept, and providing Application Programmer Interface (API) routines. The majority of the NIU will be software developed on former DIS programs, namely *CELLNET* and *F-16 Taiwan*, for host PDU management. The remainder of the NIU will be developed to handle the EUSA host-ModSAF DIS interface. An overview of the system's integration solution is shown below (see Figure 1).

NETWORK INTERFACE UNIT (NIU)

The Network Interface Unit (NIU) provides an interface between the host application software and the Local Area Network (LAN). The main functions of the NIU are:

- Create and maintain a database that contains information, that is generated by the host and external applications, received via the LAN.
- Provide API service routines that host applications can invoke to update internal data.
- Provide API service routines that host applications can invoke to acquire information from externally generated sources.
- Provide services to send and receive PDUs across the LAN in accordance with the standards contained in DIS IEEE 1278.1.
- Perform all coordinate conversions and data extrapolations required to correctly transmit data between the host and external applications.

These functions are accomplished using two separate executable tasks. One is a background task that communicates directly with the LAN, reading PDU data from the Ethernet connection, and performs filtering of the PDUs. If a valid PDU is received, the PDU is stored into the next available location in the input circular queue and the Write pointers updated. Then the output queue is read, checking for PDUs stored by the real-time portion of the NIU. If one is found, it will be validated and sent to the LAN connection and the read pointers of the circular queue are updated.

Upon receipt of a network PDU, the NIU performs a first stage filtering and only processes valid PDUs. This filtering includes the following:

- Host is the designated receiver of the PDU
- Received PDU is for the exercise in which the host is participating
- DIS version contained in the header of the received PDU matches the version used by the host
- PDU type is one that is handled by the host

If the PDU received is not valid, the type is logged and a message is displayed to request further investigation.

The other portion of the NIU is contained within the real-time application (i.e., the synchronous task) that communicates with the other host applications through the use of API routines and to the background portion also via API routines. During the real-time task, the NIU processes all internal and external entities. For internally generated entities, the NIU checks each entity's data to see if a PDU needs to be sent based on the selected dead reckoning algorithm and the minimum time interval established. If a PDU is required, the NIU performs any necessary data

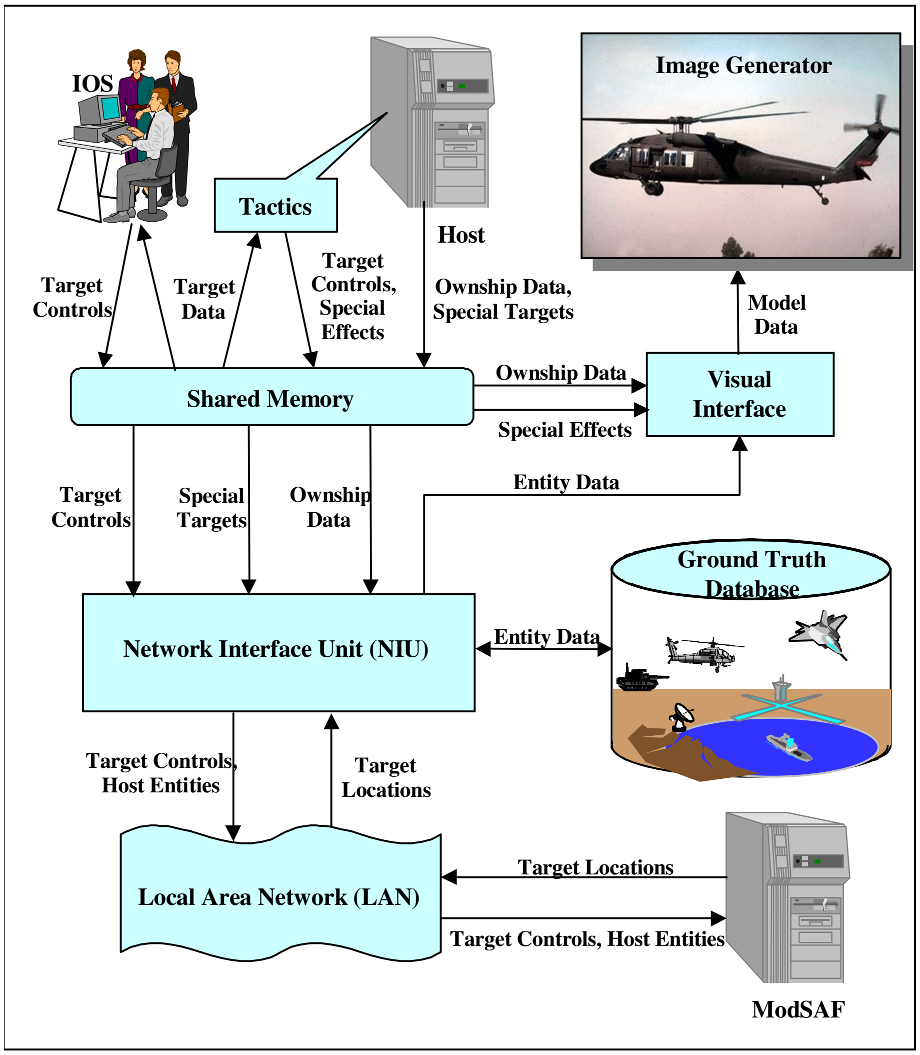

Figure 1. Architecture and Design Overview

conversions, constructs the PDU, and transmits the PDU to the LAN via the Write API routine. For externally generated entities, the NIU performs all required data conversions from the PDU to the host format and any extrapolations to bring the current entity data to the host required iteration rate.

The interface between the real-time and background NIU tasks consists of the service routines to receive and send PDUs. If a PDU needs to be sent, the Write PDU routine is called, the PDU is placed in the next available location in the output queue, and the Write pointers are updated. The background task monitors the output queue and, if one is found, writes the PDU to the LAN. The background task also handles any PDUs received from the LAN and, if one is received, places it in the input queue and the read pointers of the input queue are updated. During the real-time task, the NIU calls the read PDU routine and if a PDU is found in the input queue, the appropriate PDU processing routine is called and pointers are updated. Both portions of the NIU, background and real-time, perform error checking while reading and writing from the circular queues, preventing potential data loss due to overwriting by one process or the other. In other words, wrapping completely around the queue will be detected and logged for further investigation. The data flow for the NIU is shown below (see Figure 2).

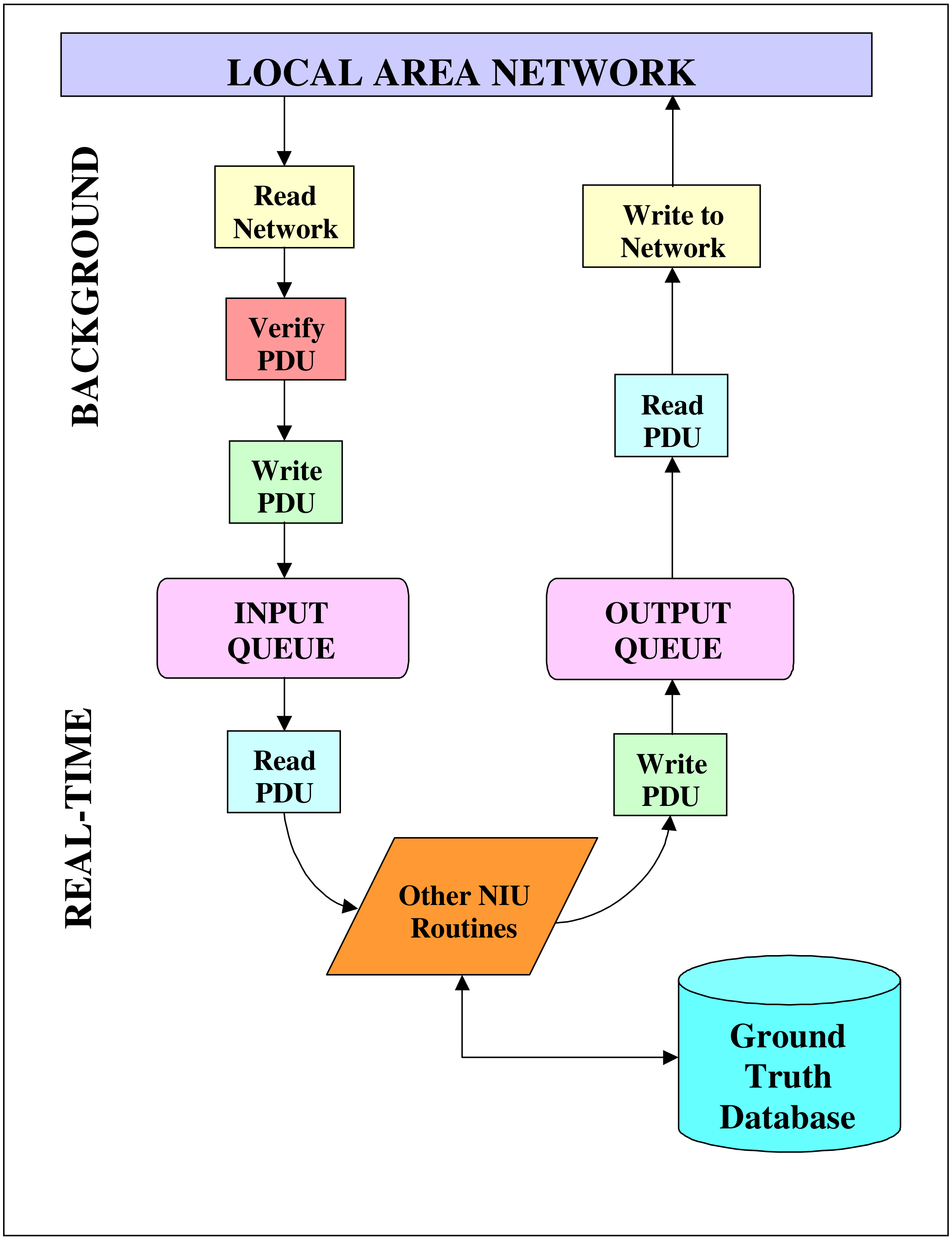

Figure 2. Network Interface Unit Data Flow Diagram

The NIU is responsible for maintaining internal data structures, called the Ground Truth Database (GTDB), which contains all the interface data between the host applications and the network. The GTDB also contains data for each of the entities that are active and a list of active requests made by the host that are waiting for replies. The GTDB can be expanded to contain other data such as fire and detonation events, emission systems, active laser designators, and radios.

The NIU interfaces with the other real-time host applications through the use of API routines for updating and retrieving data from the GTDB. These routines provide a layer of insulation between the host applications and the network interface. Therefore, if a different protocol for network communication is required, it will be transparent to the host applications. For any entities generated internally within the host, an entity update routine is called that passes the data and the local host identifier. The NIU update routine determines if the entity is new and, if so, allocates a slot within the entity data portion of the GTDB. A DIS Entity Identifier (ID) is assigned and the data within the slot is initialized. In any case, the host entity data is stored in the slot for subsequent processing. The host also requests current data for all active entities through an API routine. This routine stores the current data into the input arguments that are passed from the host application for each of the requested entities.

As part of the EUSA upgrade, ModSAF is used to control the movement of the external entities. However, the entity type, route, altitude (for air players), and speed are controlled within the host. The NIU handles all requests from the host applications for insertion, modification, or deletion of entities as well as any host initiated mode changes (i.e., freeze or restart). The NIU receives the host requests via API routines and generates the appropriate Simulation Management PDU in accordance with DIS IEEE 1278.1 standard and transmits the PDU to the LAN. The NIU also handles any Simulation Management PDUs received from the LAN that are acknowledging receipt, and performs the desired action on the ones that are sent. The PDUs utilized include:

- *CreateEntity* and *RemoveEntity*
- *StopFreeze* or *StartResume* to control the state of an entity or group of entities
- *SetData* to send the initial or modified control data for an entity
- Acknowledge received for *CreateEntity*, *ResumeEntity*, *StopFreeze*, or *StartResume*
- Data received to confirm receipt of the SetData

When required, the NIU will send the appropriate Simulation Management PDU to the network via the Write API routine. The NIU waits for the appropriate response PDU from the LAN and verifies the correct action has been acknowledged. When the request is not acknowledged within a fixed time period, another request is made. If a maximum number of retries occurs without response, the error is reported to the system for message display and logging. When the correct response is received, the NIU performs the necessary updates and removes the request from the wait list within the GTDB.

MODSAF UPGRADES

At a high level, the ModSAF upgrades can be categorized into two realms: off-line and real-time. The off-line modifications included tailoring ModSAF and its Graphical User Interface (GUI) to provide only the functions necessary to be able to generate a threat environment's route scenario off-line. The real-time modifications focussed on implementing a DIS interface to compliment the host's NIU. The DIS interface would allow the host computer to command ModSAF entities, during real-time simulation, based on the host's tactical behaviors and instructor inputs from the host's IOS.

Off-line ModSAF upgrades

ModSAF already comes with an innate ability to create and save scenarios. A ModSAF user can draw lines that traverse the Compact Terrain Database (CTDB), draw lines that adhere to roads in the CTDB, create and task entities to follow the lines/routes, etc. The first ModSAF change was made as a precautionary measure: to customize the GUI, catering to the EUSA program's needs. With this purpose in mind, the GUI was mitigated to offer only selections and features applicable for EUSA scenario creation and testing. Historical ModSAF features, such as *Chemical Editor*, which do not offer any added EUSA application value or subscribe to the theme of generating/testing a EUSA route scenario, were removed. Even vehicle and task selection menus were altered to offer only EUSA suitable options. Once the GUI was tailored for EUSA scenario generation, a ModSAF user can generate a scenario of routes (fixed sites and paths) for real-time usage. For EUSA purposes, a fixed site is merely a route with one waypoint, using ModSAF's *Point Editor*, and a path is a route with more than one waypoint, using ModSAF's *Line Editor*. Additional off-line changes were required within scenario saving and scenario loading to make the routes available for real-time usage.

ModSAF saves a scenario in its own format. In order for the host's IOS to instruct a ModSAF entity during real-time training to follow an IOS selected route, both ModSAF and the IOS need to be able to reference the same list of routes. Consequently, when a scenario is saved using ModSAF's GUI, an additional scenario file needs to be saved. This file matches what is in ModSAF's scenario file but is in an IOS readable format. A second scenario saving aspect that required change related to what ModSAF saved in a scenario. Lines and points (i.e. routes), their waypoints, and any related comments/labels are the only items of particular interest to the IOS. Anything else is superfluous to what is needed by the EUSA program. Therefore, ModSAF's scenario saving feature was revised to match what was saved in the IOS-specific scenario file (i.e. only those items that related to lines and points).

To ensure compatibility between ModSAF and IOS route specification, both ModSAF and the IOS needed to coordinate the loading of particular scenario files, each referencing the same routes. Normally, a ModSAF user will load a saved scenario via the GUI. On the EUSA program, only the back end of ModSAF (i.e., the simulation engine, not the GUI) will be operational during real-time training. To facilitate a coordinated loading of compatible scenario files, ModSAF was modified so that it automatically loaded a scenario with an arbitrary name. Likewise, the IOS was constructed so that it would load its particular arbitrarily named scenario file that correlated with ModSAF's scenario file. Consequently, upon start-up of a training session, both ModSAF and the IOS would each, automatically load their respective content-correlated scenario file. Once the real-time simulation was operational, the instructor at the IOS could create ModSAF entities and assign them to any one of the automatically loaded and pre-defined routes. To allow the IOS the ability to create, control, and remove ModSAF entities and allow the host's tactical behaviors to maintain control over ModSAF entities, a real-time DIS interface needed to be developed.

Real-time ModSAF Upgrades

On the host computer, the real-time DIS interface developed is the previously discussed NIU. In ModSAF, its counterpart needed to be conceived. The interface would allow for coordination between a real-time, man-in-the-loop, flight simulation and an event-driven synthetic tactical environment. Specifically, the interface would allow the host and its IOS to elicit control over ModSAF in the following manners:

- Freeze, unfreeze, and quit ModSAF
- Create, position, orient, pause, resume, and remove ModSAF entities
- Task ModSAF entities to follow pre-defined routes
- Command moving entities to reverse direction along a route, change an entity's commanded route speed, and change an aircraft entity's commanded route altitude

To utilize this interface during real-time training, the host's NIU would need to take advantage of the following DIS PDUs: *CreateEntity*, *RemoveEntity*, *StopFreeze*, *StartResume*, *SetData*, *Data*, and *Acknowledge*.

Although version 5.0 ModSAF is DIS compliant, its DIS communication abilities needed to be embellished in order for this DIS interface to succeed. ModSAF possesses the capacity to display entities from external simulations, provided the external simulations are part of the same DIS exercise. However, these external simulation entities are not controlled by ModSAF, but are controlled by an external simulation. ModSAF knows only of their presence via *EntityState* DIS PDUs. As previously mentioned, the EUSA program desired differently. The EUSA program needed ModSAF to take control of an entity in the sense that it would assume responsibility for placing stationary entities within its environment and for moving threats throughout its terrain database (i.e. its CTDB that is correlated with the host's visual database). This meant that entities needed to be created locally within ModSAF in order for ModSAF to assume responsibility for them. This resulted in the following ModSAF changes.

Additional *SetData* DIS PDU attributes needed to be added to ModSAF's existing repertoire. As part of the EUSA program's DIS network interface, many of the entity controlling commands were implemented as an attribute within the *SetData* PDU. Therefore, when an instructor relocates a threat at the IOS, ModSAF is subsequently instructed to relocate its corresponding entity to the location contained within the *SetData* DIS PDU that it received from the host's NIU. Other instructional information that required unique *SetData* DIS PDU attributes includes: tasking an entity to follow a route, changing an entity's commanded route altitude or speed, and setting an entity's initial velocity or orientation.

In order for ModSAF to appropriately respond to instructions from the host's NIU, ModSAF's existing DIS interface was modified filtering out the appropriate NIU generated PDUs from the rest of the simulation's network traffic. Without such changes, ModSAF would not respond to an NIU's *CreateEntity* DIS PDU as a request to create an entity locally within ModSAF. Rather, ModSAF would default to treating the PDU as a request, from an external simulation, to simply place the entity within ModSAF as an "external entity." That is, one whose movement and behavior would be controlled by an external simulation through *EntityState* DIS PDUs.

To provide the functional foundation behind the DIS network interface, a new library of EUSA specific code needed to be added to ModSAF. With this addition, ModSAF would appropriately respond to the host's NIU DIS PDUs. When the host's NIU generates instructions through DIS PDUs, there exists EUSA specific code to interpret the instructions. This means that when the NIU requests an entity to be created within ModSAF, via a *CreateEntity* DIS PDU, EUSA specific ModSAF code will create an entity locally within ModSAF, retrieve a DIS identifier for that entity, and send an *Acknowledge* DIS PDU back to the host's NIU with that entity's new DIS identifier. Upon receiving the *Acknowledge* DIS PDU, the host's NIU will then be able to use the entity's DIS identifier to further address PDUs it sends to ModSAF. In this fashion, control over the behavior of EUSA's ModSAF entities was established.

As control was established, it became necessary to tame ModSAF. Many of ModSAF's entities, by default, have a fundamental ability to respond and react to more than obstacles in the path they are tasked to follow. While an entity may have been tasked to follow a certain route, avoiding collisions and obstacles along the way, some entities will take alternate actions when confronted with the enemy. In the EUSA program, all tactical behaviors were to be controlled by the host computer. Autonomous reactions from ModSAF entities would only disrupt an effort to preserve a predictable and deterministic threat environment. Therefore, ModSAF entity reactions were disabled. Entities no longer engage the enemy nor do they retreat under control of ModSAF behaviors. These entities will only be cognizant of and respond to another entity's presence as it relates to following a route.

As a result of the above changes, ModSAF became ready for real-time integration with the host. After creating a scenario at an off-line ModSAF station, an instructor at the IOS can create players during real-time, assign them to follow routes, alter their commanded route following speed, insert threats at fixed sites, and so forth. The host's NIU handles the translation of host tactical behaviors and IOS directives into DIS PDUs. The EUSA modified ModSAF discriminately listens for the NIU's DIS PDUs and reacts accordingly: appropriately acknowledging the receipt of the DIS PDU, creating entities locally within ModSAF, assigning entities to follow routes, changing an entity's traveling speed, etc. Since all EUSA entities were created locally within ModSAF, ModSAF will automatically, by default, manage an entity's movement through the CTDB, perform any necessary dead reckoning, and broadcast *EntityState* DIS PDUs. These PDUs inform the NIU's ground truth database of the entity's current geographical location and orientation.

Even the solution to implement record/playback and a "fly-out" functionality, within ModSAF, fell out of the above DIS network interface capabilities. Only the host computer needed to worry about record/playback and the ability to "fly-out" of a playback. During a recording of a training exercise, all threat environment data the host needs to record can be found in its NIU's ground truth database. All ModSAF does is continue to behave in the same real-time manner as it always does when it is not in freeze. To implement a mission playback with ModSAF, the host merely needs to reset ModSAF. In other words, the NIU needs to place ModSAF in freeze (*StopFreeze*), remove all ModSAF entities (*RemoveEntity*), reinsert the entities present during the recording (*CreateEntity*), setup each entity according to its recorded location, orientation, velocity, and assigned route (*SetData*), and take ModSAF out of freeze (*StartResume*). ModSAF then resumes its duty of moving entities along their assigned routes just as it did during the real-time recording. Again, most of ModSAF's control over entity behaviors, aside from route following abilities, was curtailed. The increased degree of repeatability and determinism gained from this behavior allows a moving threat to virtually traverse a path in the same manner as it did during the original mission. Since ModSAF is already “live” during playback, the ability to “fly-out” of a playback is already taken care of.

FUTURE ENHANCEMENTS

Although a DIS interface capability was added to each of the EUSA flight simulators, only a portion of the ModSAF functionality was utilized. Most of the control for each tactical environment, still resided within the host computers. As a future upgrade, this control could be removed. This would allow for a more encompassing usage of the constructive simulation and more robust scenarios could be generated. With an expanded ground truth database to maintain threat emission data and fire detonation events, the data could be used to stimulate existing host interfaces providing appropriate interactions with the ownship. Additionally, with a modified NIU to handle the appropriate PDUs, the UH-60 and CH-47 would be able to participate in distributive network exercises. As part of the proof-of-concept program, *CELLNET*, this capability has been previously demonstrated on the UH-60 device.

A second area with potential for improvement during a future upgrade, is in training communications. Each of the simulators contain on-board radios for use during a training exercise and communication with the role playing instructor and fellow crew members. The voice transmissions are not broadcast onto the DIS network as part of the EUSA program. Alternately, transmitter and signal PDUs could be generated and transmitted. This would permit communication between EUSA flight simulators (UH-60 and CH-47) as well as other participants in a distributed exercise. This too, has already been done on the *CELLNET* and *F-16 Taiwan* programs.

Finally, a third area with potential for improvement is in the area of HLA compliance. While the EUSA program was not under the mandate to provide an HLA compliant system, the use of ModSAF greatly helps to ease the burden of such a transition. Due to existing “gateway” products for DIS/HLA translations, HLA compliance for many DIS compliant products such as ModSAF has been made possible.

SUMMARY

The use of ModSAF on the EUSA program provided a cost effective means to meet contract requirements of maintaining existing capabilities within legacy training devices. In doing so, an alternate implementation of the functionality of ModSAF was utilized allowing dynamic scenario generation while maintaining the existing host controlled features.

As the EUSA program comes to closure, two legacy flight simulators have taken their first steps toward interoperating in a global synthetic battlespace. Updating legacy simulators as a whole can easily lend itself to high cost and maintenance of stagnant technology. The EUSA solution curtails unnecessary spending and bridges the legacy technology into the next generation of simulation.