# ANALYSIS OF A REAL-TIME HLA DISTRIBUTED MISSION TRAINING FEDERATION

**Robert E. Murray**
**The Boeing Company**
**St. Louis, Missouri**

**Steve L. Monson**
**The Boeing Company**
**St. Louis, Missouri**

**Abstract**

The first phase of the C-5 Distributed Mission Training (DMT) contract has been successfully developed and delivered to the Air Force Air Mobility Command. It has met its goal of using the High Level Architecture (HLA) to link two existing C-5 training simulators to demonstrate DMT technology capable of training missions such as formation airdrop and air refueling.

As this is one of the first real-time manned simulation programs delivered using HLA, it has been closely watched as a demonstrator of HLA's ability to serve the needs of training simulation and real-time simulation in general. To this end, the contract funded an effort to perform objective tests to analyze HLA Run-Time Infrastructure (RTI) performance in a real-time environment.

This paper provides a brief introduction to the C-5 DMT system architecture. The paper's major emphasis is on the purpose and results of the analysis tests that were performed on the C-5 DMT simulators. The four tests were network bandwidth utilization, RTI latency, processor utilization, and entity position error due to dead reckoning. All tests were run with a varying number of aircraft entities to show the trends that result from scaling to larger exercises. Use of the trends to predict system performance in larger DMT exercises is discussed. Also described is a means for simulating the effects of a long-haul network by injecting statistically random delay in the local area HLA network.

The overall conclusion from this project, verified by the analysis tests, is that real-time manned simulation is possible using the current HLA RTI under favorable conditions, but only for federations of up to 10's of entities. Federations of 100's or 1000's of entities will require an RTI with a corresponding order of magnitude increase in performance and capability.

**About the authors**

Mr. Robert Murray is a Principal Engineer at Boeing with 14 years of experience in the Flight Simulation Technology department in St. Louis. He has worked on nearly all hardware and software aspects of applying commercial network products to meet real-time I/O and distributed simulation requirements. Mr. Murray was responsible for the HLA implementation in the C-5 DMT program, and continues to develop and support distributed computing products such as real-time messaging middleware, DIS and HLA interfaces, and high-speed networks used in engineering flight simulators, training systems, and other programs in Boeing. Mr. Murray received an MS degree in Electrical Engineering from Washington University in St. Louis in 1993 and a BS in Electrical and Computer Engineering from the University of Cincinnati in 1983.

Mr. Steve Monson has 13 years of experience in flight simulation, networking and embedded I/O systems. He is currently the Principal Investigator of the Simulation Networking Research and Development, leading DIS and HLA technology development for Boeing Training and Support Systems. He is an active participant in the RPR FOM development effort and an Assigned Reviewer for the RPR FOM standardization effort. He has continuously held a position in the SISO RTI forum Planning and Review Panel beginning with the first Simulation Interoperability Workshop. Mr. Monson received a Master of Science degree in Electrical Engineering from Washington University in St. Louis in 1992. He received a Bachelor of Science in Computer Engineering from Iowa State University of Science and Technology in 1987.

# ANALYSIS OF A REAL-TIME HLA DISTRIBUTED MISSION TRAINING FEDERATION

**Robert E. Murray**
**The Boeing Company**
**St. Louis, Missouri**

**Steve L. Monson**
**The Boeing Company**
**St. Louis, Missouri**

## INTRODUCTION

Boeing Training Systems designed and implemented a networking architecture to add Distributed Mission Training (DMT) capabilities to C-5 WST (Weapon System Trainer) devices at Dover Air Force Base, utilizing the HLA (High Level Architecture). This effort was complicated by many factors. The C-5 WSTs were not designed to be networked training devices. The training devices were not designed by Boeing, which meant limited access to proprietary source code. In addition, HLA had not yet been employed in a real-time man-in-the-loop training device in an operational training environment. Also, existing RTI (Run Time Infrastructure) implementations have not been proven in such a strict real-time environment under the demands of DMT. However, the approach already under development by Boeing Research and Development for use in its own real-time training devices utilizing embedded systems met this need. This approach is a native HLA implementation (no gateways, etc.) and was demonstrated at the Inter-service/Industry Training Simulation and Education Conference (I/ITSEC) '98 in our F/A-18 trainer.

This was the first phase of a multi-phased effort to add DMT capabilities to additional C-5 WSTs, that will be networked together long haul and ultimately include other DMT training devices such as C-17s and KC-10s. This first phase involved the design, implementation, installation and evaluation of an HLA-compliant federation of two C-5 WST simulators at Dover Air Force Base. After successful integration and test, the training devices were subjected to an extensive battery of individual and network tests. After a brief orientation of the system architecture, this paper details the network testing, and provides detailed results and recommendations, including an evaluation of RTI performance under the demands of DMT.

## SYSTEM ARCHITECTURE

The primary components involved in the modification of the system are the two C-5 WST computational systems, their I/O and audio sub-systems, and the Federation Manager (Figure 1).

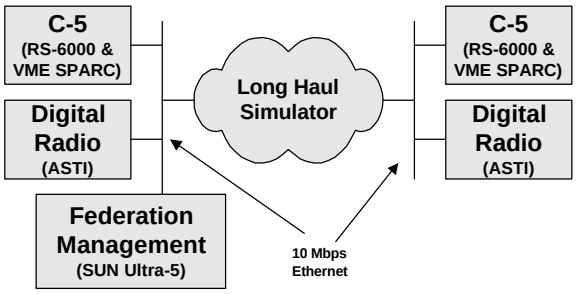


**Figure 1. System Architecture**

### Hardware Architecture

***C-5 Computational System:*** The C-5 trainer host computer was an IBM RS-6000 running the AIX operating system. Executing the HLA RTI on the RS-6000 was not practical because at the start of the program the RTI was not available for AIX, there was no Ethernet interface suitable for RTI use, and there were no available Microchannel slots to add another interface. There did exist a VME chassis connected to the RS-6000 by a VME-to-Microchannel Bit3 interface. To add processing capability to the system, a VME processor board was added. The board is a SPARC-based VME Single Board Computer with a 170Mhz TurboSPARC-II CPU running Solaris 2.5.1, for which the RTI is readily available. A major advantage of using a separate processor to run the RTI is that the main simulation application is completely isolated from the non-deterministic response of the current RTI. This approach has been used very successfully in past designs. (Youmans, 1999).

***Audio System:*** The original audio system was a completely analog system. Prior to the modifications, it provided all simulated radio communications within each individual training device, as well as intercom communications to the device's IOS (Instructor Operator Station). For DMT, the audio system was augmented with a digital communications system from Advanced Simulation Technology inc. (ASTi). At the

time of the modification, a commercial HLA communications system was not available. Although communication was performed using DIS (Distributed Interactive Simulation), the communication was sent over the same physical network as the HLA traffic, and thus contributed to the overall network traffic in much the same way.

***Long Haul Simulator:*** A means to simulate a long-haul network was developed to provide the ability to study the effects of connecting training devices across a Wide Area Network (WAN). The long haul simulator (LHsim), based on previous work (Swaine & Marz, 1995), models a WAN by introducing a statistical delay, limiting overall throughput, and occasionally dropping packets. The LHsim also collects statistics on data passed through it. LHsim applies three models to each packet:

- The Statistical Delay Model
- The Lost Packet Model
- The Rate Limit Model

The Statistical Delay Model employs an exponential delay distribution (Figure 2). Each packet is delayed according to this distribution before being transmitted. This model is parameterized by minimum, average and maximum delays.

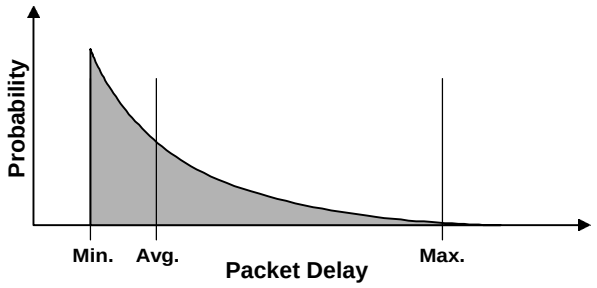


**Figure 2. LHsim inter-arrival PDF (Probability Distribution Function)**

The Lost Packet Model accounts for packets that are discarded due to bit-error rates. Losses due to buffer overruns can occur as a result of the Rate Limit Model.

The Rate Limit Model operates on the outgoing side of the message queue. This model estimates the theoretical end-time of a packet at a given data rate, such as the 1.5 Mbps of a T1 line. Additional packets are prevented from being transmitted until the theoretical end of transmission of the current packet. Packets are not lost directly due to this model. However, packets can be lost at the incoming side of the message queue if the queue overflows.

***Absolute Time Stamps:*** Using LHsim, the C-5s were connected as if they were separated by thousands of miles. In a long haul scenario, absolute time stamps are typically used so the exact age of the data can be determined, and the remote object can be extrapolated to its current position. To accomplish this, VME based IRIG-B time boards were inserted into each C-5s VME chassis. This also allowed correlated timing measurements to be performed between the simulators for testing and analysis. In geographically separated training devices, these boards would normally be receiving time via a Global Positioning System (GPS) satellite receiver. However, since the C-5s were co-located, the satellite receiver was not necessary. In this installation, the boards were connected together via a coax to synchronize them using the IRIG-B time distribution standard.

***Federation Management Workstation:*** A SUN Ultra-5 workstation with a 270Mhz UltraSPARC-IIi CPU was used as the Federation Management Workstation (Figure 1). It runs both the RTI and Federation Executive processes required by the DMSO (Defense Modeling and Simulation Office) RTI. This reduces the processing capacity requirements of the modified WSTs, one of which would need to run these processes if the workstation were eliminated from the design. This workstation is also a federate in the exercise. It is the definer/owner of a DMT exercise. It maintains attributes of a DMT exercise that are common and utilized among the exercise participants. These common attributes of an exercise include atmospheric conditions, run/freeze state, etc. A WST can join the exercise, request a change in these attributes, and leave the exercise, but the exercise environment persists. This eliminates the added complexity of transferring ownership of the shared attributes among the participants, and makes the workstation the ideal platform for a DMT control application.

**Software Architecture**

The software architecture is mapped across the IBM RS-6000 and the SPARC VME processor board (Figure 3).

***Simulation Distribution Layer*:** The SDL is composed of the changes and additions to the simulation application to make the C-5 a distributed trainer. The SDL provides for extraction and accumulation of local C-5 and tanker data, integration of external entity data into the visual system display, weather radar simulation, IOS map displays, etc., and management of the common attributes of a DMT exercise.

***Simulation Adaptation Layer:*** The SAL provides data and communication services required by the SDL to export locally generated entity data, and import data describing entities generated by other DMT exercise participants. The SAL Application Programming Interface (API) presents a simple, well-defined interface to the SDL that hides the implementation details of the services it provides. The SAL is split into two parts, the Upper SAL and the Lower SAL.

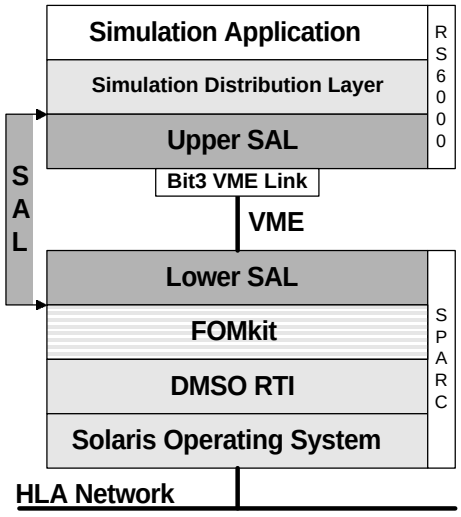


**Figure 3. Software Architecture**

***Upper SAL:*** The Upper SAL provides the FORTRAN- and C-callable functions, specified by the SAL API, to the Simulation Distribution Layer software. It is an AIX object module that is linked into the simulation application. It relays control and data requests to the Lower SAL, where the computationally significant part of handling the requests occurs. It also handles the VME bus exchange of data with the Lower SAL, through the Bit3 device driver.

***Lower SAL:*** The Lower SAL sends and receives the attribute updates utilizing FOMkit. It performs unit translation (meters to feet, etc.), coordinate transformation, and dead reckoning of attributes sent and received. It also sends attribute change requests to the Federation Management Workstation. The Lower SAL calculates a consistent start-of-frame time necessary for smooth updates of dead reckoned aircraft entities.

First order dead reckoning is used. The error thresholds are set at one meter for position and three degrees for rotation. To achieve extremely smooth tanker motion for aerial refueling, the tanker position error threshold is reduced to 0.2 meters when a C-5 is near it. See IEEE (1995) for a description of dead reckoning.

***FOMgen™ (Flexible Object Model generator)***
***FOMkit™ (Flexible Object Model toolKIT)***
The low-level FOMkit middleware provides a value-added abstraction from the RTI interface API. It provides functionality such as automatic handling of attribute update requests from late-joiners, handling of RTI requested flow-control such as start/stopUpdates() and turnInteractionsOn/Off(), as well as abstracting the simulation application from RTI housekeeping. FOMkit is entirely automatic code generated directly from the FOM (Federation Object Model) or SOM (Simulation Object Model) using FOMgen. FOMgen is an internally developed tool. It reads the output of DMSO's Object Model Development Tool in the form of a .omd file. From this, it generates a FOM-specific FOMkit. Typical C++ FOMkit object methods to update attribute values are shown below:

```
C5.updatePosition(PositionStruct &posn);
C5.updateEngineSmokeOn(boolean isOn);

C5.commitUpdates(double time);
```

The first two methods do not cause the RTI updateAttributeValues() service to be invoked – this does not occur until the last method is invoked. The last method causes all the updates scheduled thus far to be sent by the RTI. In a simple Data Distribution Management implementation where all of the attributes belong to the same routing space and are sent via the same transport mechanism, this allows all the attributes for a single object to be sent together through the RTI.

Application development began on the C-5 application using a FOMkit based on RTI 1.0v3, as RTI 1.3 was not yet available. Since FOMkit presents an object oriented API that is independent of the underlying transport mechanism, the C-5 application was able to convert to RTI 1.3 when it became available from DMSO without changing the application.

The Lower SAL, FOMkit, and the RTI are linked as a single process that runs on Solaris at real-time priority.

***FOM:*** This networking architecture used HLA and a FOM based on version 0.6 of RPR FOM (Real-time Platform Reference Federation Object Model). Minor additions were made to the RPR FOM to support atmospheric conditions, additional simulator control, and attributes specific to aerial refueling missions.

## ANALYSIS TEST RESULTS

Four HLA analysis tests were run as part of the C-5 DMT project. These were objective tests with each

measuring a critical aspect of HLA operation. The four tests were:

1. Network bandwidth – the network utilization required by the HLA RTI
2. CPU Utilization – the amount of processor time spent processing pieces of the DMT software
3. Latency – the end-to-end transport delay injected by the HLA RTI network and layers of DMT software
4. Position Error – The error in position of an aircraft generated by one simulator as seen by the other simulator

Subjective tests including Air Force pilot evaluations were also performed as part of this program and are detailed in a companion paper also being submitted for publication and presentation at I/ITSEC 1999.

**Common Setup Conditions**

All tests were run on the two C-5 training simulator devices at Dover Air Force Base, WST5 and WST6, configured as previously described.

The DMSO 1.3 version 5 RTI was used. The RTI was configured for minimum latency – bundling turned off and the UDP polling interval set to zero. Bundling off causes every update to be sent immediately instead of waiting to be combined with other updates. The zero polling interval causes the RTI to process incoming Ethernet messages every time it is ticked. The RTI was ticked at 1 ms intervals to make individual network latency measurements accurate to within one millisecond.

All tests except Position Error used a "C-5 cloner" to vary the number of aircraft in the simulation. This is software that was added to the host computer to generate up to four additional C-5 aircraft per simulator. The additional C-5s are simply ownship position stored and played out at two-second intervals. Each of the two simulators can thus generate five C-5s flying in single-file, two seconds apart plus a tanker for a total of six aircraft per simulator (12 total) in the maximum configuration. The cloned aircraft are transmitted on the HLA network just like the ownship, they are received and processed by the other C-5 simulator, and they are visible on the out-the-window visual and IOS map of both simulators.

In all tests, one of the variables is whether the aircraft are flying straight & level or "dynamic." Straight & level tests use the autopilot to hold a constant heading and altitude at approximately 4000 feet. Dynamic tests use the autopilot to hold the C-5 in a turn with 45 degrees of bank and with approximately a 1000 foot per minute climb or descent. This is considered dynamic because turn accelerations are not included in first order dead reckoning. The error threshold is exceeded (and new updates are sent) about twice per second for a C-5 in a 45 degree bank. The tanker is always straight & level.

The Position Error test used the long-haul network simulator described above to add network delay between the two simulators. The long-haul simulator was also used in the Network Utilization test to measure the network bandwidth. No delay is injected for this test.

Except for the Radio bandwidth test, there was no radio traffic on the simulation network for these tests.

**Network Bandwidth Test**

The purpose of this test is to measure how much network bandwidth is used for various simulation conditions. This can be extrapolated to determine how many aircraft and voice communication channels can be simulated simultaneously on a T1 long-haul network.

***Measurement:*** Network bandwidth utilization was measured in Kilobits per second (Kbps)

***Test Setup:*** The test was run on two C-5 simulators connected via HLA through the Long-Haul Simulator. No extra delay was injected in the HLA network; the Long-Haul Simulator was used only for reporting network bandwidth. C-5 cloner software was used on the host computer to add extra aircraft entities to the simulation.

***Variables:*** The number of entities was varied starting at just one C-5 per simulator, then by adding a tanker generated by each simulator, then adding two and then four C-5 clones per simulator for a total of 12 aircraft in the maximum configuration. This was repeated for two flight conditions: Straight & Level and Dynamic (the tankers were always straight & level). With one tanker and three dynamic C-5s flying per simulator (eight aircraft total), one and then two simultaneous voice transmissions over the simulated radio link were added to measure the bandwidth of voice communication.

***Results:*** Bandwidth utilization test results with no simulated radio communication (Figure 4) show that each additional C-5 flying straight & level adds only 1 Kbps of bandwidth to the network. Each additional dynamic C-5 adds 5 Kbps. The radio bandwidth test results (Figure 5) shows that each channel of voice communication adds 82 Kbps, the expected value for

voice transmitted with DIS signal PDUs (Protocol Data Units).

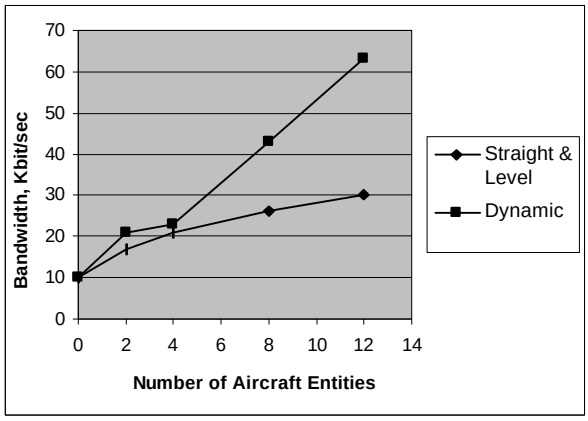


**Figure 4. Network Bandwidth Utilization**

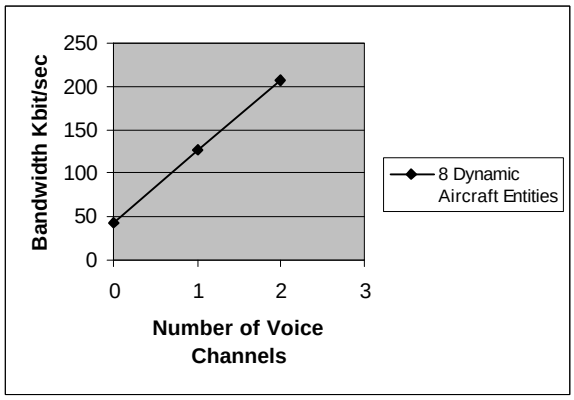


**Figure 5. Network Bandwidth with Voice Communication**

These results can be extrapolated to calculate how many aircraft and voice channels will fit on a T1 long-haul network line. If the T1 line with encryption is assumed to provide a bandwidth of 1 Mbps, then nearly 200 dynamic C-5s could be flown with no voice communication. Each channel of voice eliminates 16 to 17 C-5s from the total. For example, 100 dynamic C-5s can fly with 6 simultaneous voice channels. Since the simulated radio used DIS protocol in these tests, the results can be used only as a very course estimate of the bandwidth required for the HLA radio planned for the next phase of this program. Refer to Nemeth (1999) for additional results on HLA radio.

**CPU Utilization Test**

This test measures the percentage of time spent processing DMT software, both in the IBM RS-6000 computer and on SPARC board.

***Measurements:***

- Total SPARC processor utilization – all useful processing on the SPARC CPU including the Lower SAL, FOMkit, and all RTI processing and RTI callback functions.
- RTI processor utilization – percentage of time spent in RTI calls including RTI tick calls. Time spent in callback functions made from the RTI is not included because it is considered user processing.
- IBM RS-6000 processor utilization – percentage of time spent in DMT software processing on the RS-6000 processor.

***Test Setup:*** Two C-5 simulators were connected via HLA in the standard way. The C-5 cloner software was used on the host computer to add extra aircraft entities to the simulation.

Total SPARC CPU utilization is calculated by tagging the time that host command processing starts or when an RTI tick call is made. The stop time is tagged when the command processing completes or the RTI tick call returns. The difference between these two times is calculated and added to a total that is kept for the duration of each simulation frame. The total was recorded at the end of each frame and then zeroed for the start of the next frame. The test was run for 1000 frames (33.3 seconds).

RTI utilization is measured in a similar manner. The start time is tagged when any call is made to an RTI function including the RTI tick. The stop time is tagged when the RTI function returns and the difference is accumulated and recorded once per frame. Additionally, a stop time is tagged and difference accumulated at the beginning of a user function called back from the RTI. A start time is tagged just before returning to the RTI from the callback. This eliminates from RTI utilization the time spent in user code during callback processing. Time spent in callbacks is included in the total SPARC CPU time because separate start and stop time tags are kept for total utilization and RTI utilization.

The raw SPARC utilization data was reduced for reporting by dividing the utilization time per frame by the 1/30-second frame time to get a percentage, averaging the results of the 1000 frames, and plotting them for each test run.

Host CPU utilization was measured by timing the FORTRAN code that was specific to DMT. The Unix gettimeofday function was called at the beginning and end of each segment of DMT code. The difference

between the start and stop time was added to the total for each frame. The total was then used to calculate an average. The gettimeofday function on the IBM has one millisecond resolution so individual frame results are not accurate. However, averaging the results of thousands of frames gives a very accurate result.

***Variables:*** The number of entities was varied starting at just one C-5 per simulator, then by adding a tanker generated by each simulator, then adding two and then four C-5 clones per simulator for a total of 12 aircraft in the maximum configuration. This was repeated for two flight conditions: Straight & Level and Dynamic (the tankers were always straight & level).

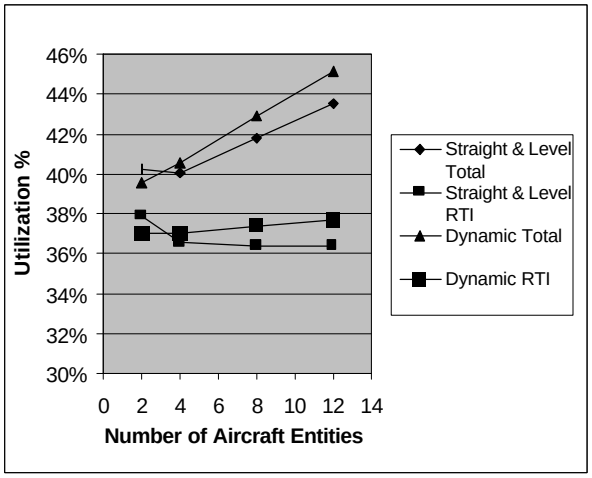


**Figure 6. SPARC Processor Utilization**

***Results:*** There is a very high utilization for only two aircraft (Figure 6) due to ticking the RTI at 1 ms intervals during SPARC processor idle time, even though most of those ticks don't result in any useful processing. Total SPARC CPU utilization for dynamic aircraft shows a linear increase that could be extrapolated to find a limit on the number of aircraft the RTI can handle. However, making such predictions is dubious at best. It has been reported by other RTI evaluation projects that extrapolating RTI results from small federations is not accurate. A more strenuous test of the RTI is the only way to get conclusive scalability information.

Due to the inconclusive results in the first utilization test, a more thorough test of the RTI on identical hardware was performed in a lab test. CPU utilization of the RTI send (update) and receive (reflection) of C-5 spatial data was measured. The receive time was measured by timing the tick call, but does not include any time spent in the user callback. In addition, only those timings that actually resulted in received data were tallied. Calls to tick that did not produce a user callback with reflected data (an Opaque tick) were also tallied (Figure 7). With 4% of the data showing CPU utilization over 400usec, this yielded much less deterministic results.

| | Minimum | Average | Maximum |
|---|---|---|---|
| **RTI Send** | 0.93 | 1.0 | 1.2 |
| **RTI Recv** | 0.88 | 0.96 | 1.3 |
| **Opaque tick** | 0.122 | 0.232 | 1.79 |

**Figure 7. RTI CPU Utilization (milliseconds)**

Obviously, a full millisecond to send or receive data seriously limits the scalability. The nearly 2 ms lost occasionally during an opaque tick may also limit scalability. To meet the requirements of DMT, future RTIs must be designed with scalability in mind.

The result of utilization for the RS-6000 processor was an average of 2.1 ms, or 6.3% of the 30 Hz frame, for all test conditions. The reason this number didn't change for a varying number of aircraft entities was that most of the time is spent in reading and writing data over the VME bus to and from the SPARC processor board. The overhead of calling the Bit3 driver that performs VME transfers is rather high, in the range of 200 μsec. To avoid making more Bit3 driver calls than necessary, the data from all six internal slots is copied every frame, even if they are not all full. Likewise, all six external aircraft slots and all IOS data are transferred as one contiguous read every frame.

**Latency Test (Transport Delay)**

Latency is an important parameter in real-time simulation networks (Monson, Johnston, & Barnhart, 1997). Simulated entities don't appear to behave correctly if too much time elapses in getting data distributed to other simulators and there is some kind of feedback that is affected by this delay.

***Measurements:*** Transport Delay was measured through the RTI alone and through all SPARC board send and receive processing including the RTI (see Figure 8.).

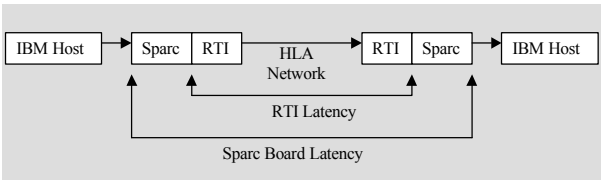


**Figure 8. Latency Measurement Points**

***Test Setup:*** Two C-5 simulators were connected via HLA in the standard way. C-5 cloner software was used on the host computer to add extra aircraft entities to the

simulation. Timestamps were taken from the IRIG-B time board so that a consistent time was used throughout. The send times were tagged at the beginning of SPARC processing and at the final Update call to the RTI for of each outgoing entity. Both timestamps were put into the outgoing update in the user tag parameter. The receive times were tagged at the beginning of the callback function made by the RTI and at the end of SPARC processing for each entity. The send times wee subtracted from the corresponding receive time to get latency. Each individual latency number was recorded.

End-to-end latency from one IBM RS-6000 to the other was not measured because the Software running on the IBM does not have quick access to the VME time board. Also, the host simulation frames are not synchronized and it is difficult to eliminate the time that received data sits waiting to be used by the host in the next frame. The Air Force Research Lab plans to run a SNAP (Simulation Network Analysis Project) test on the C-5 HLA network. It will give a much better measure of overall latency through the entire system (Bryant, Douglass, Ewart, & Slutz, 1994).

***Variables:*** The number of entities was varied starting at just one C-5 per simulator, then by adding a tanker generated by each simulator, then adding two and then four C-5 clones per simulator for a total of 12 aircraft in the maximum configuration. This was repeated for two flight conditions: Straight & Level and Dynamic (the tankers were always straight & level).

***Results:*** The results of all test runs were virtually identical. The average RTI latency was 2.3 milliseconds for all tests from two to 12 aircraft, straight & level or dynamic. The latency through the SPARC board layers was 2.8 ms. Again, this was same for all runs, two to 12 aircraft, straight & level or dynamic. Apparently, 12 aircraft was not a sufficient load on the system to cause a trend that could be extrapolated to determine the scalability.

To show the variance in latency, the RTI latency results from the 12 dynamic aircraft test run were rounded to 0.1 ms values, the number of each value occurrence was counted (binned), and the counts charted on a histogram (Figure 9). This is a probability distribution function (PDF) of the RTI latency results.

It can be seen that most of the samples fall within ±0.5 ms of the average. This one millisecond variation was expected due to the fact that the RTI is being ticked once per millisecond. Incoming updates cannot be processed until the RTI is ticked, so the tick rate directly affects the measured RTI latency. A 1 ms tick period adds 0.5 ms to the average latency and ±0.5 ms of variance.

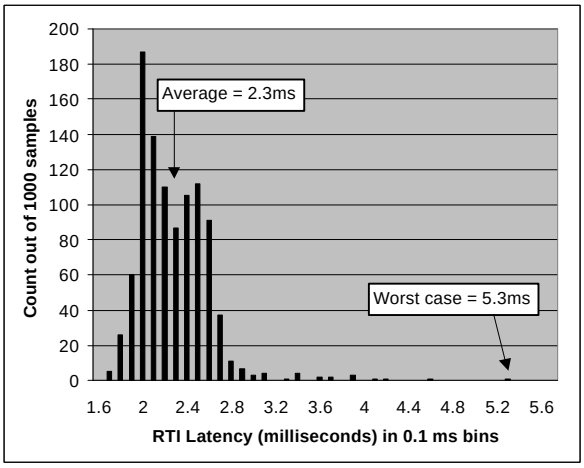


**Figure 9. RTI Latency Distribution, 12 Dynamic Aircraft**

The lightly loaded tests had virtually no samples outside ±0.5 ms range. Even the highest loaded test (12 dynamic aircraft shown in Figure 9) had only 4% of its samples fall outside the range, mostly at the high end. The single highest RTI latency sample was 5.3 ms.

**Position Error Test**

The position in simulated space of an entity generated in one simulator as seen by another simulator is rarely exact if the entity is moving. Since there is always a finite amount of elapsed time between the generation of a new entity position and the distribution of that position, the entity has moved before other simulators can get and use its position.

The causes of the elapsed time are threefold. The first is the latency through the distribution software and over the network. The second is the delay caused by simulation frames not being synchronized, that is, updated position data received by a simulator waits for the next simulation frame to begin before it can be used. The third is dead reckoning and the associated smoothing algorithm.

***Measurement:*** Positional error was measured between the "true" position sent from one C-5 simulator and the dead-reckoned position received and calculated by the other C-5 simulator.

***Test Setup:*** Two C-5 simulators were connected via HLA through the long-haul network simulator. A separate private Ethernet network connected the two SPARC boards and bypassed the long-haul simulator.

This network was used to send the true C-5 position every simulation frame. The difference between the true position received on the private network and the position dead reckoned from HLA data is the error. It was calculated once per frame on the receiving simulator and recorded.

There is one complication to this due to the fact that the frame timing for two simulators is not synchronized. The position sent by the originating simulator as valid for a point in time in that simulator's frame. It is not exactly correct at the receiving simulators frame time, hence the "true" position is not really truth. To correct for this, the truth data was extrapolated on the receiver to its current frame time using the sender's time and velocity sent with the true position. Of course, this can't be a perfect extrapolation but the inaccuracy is minimal since truth data is extrapolated at most one 30 Hz frame time (33.3 ms).

***Variables:*** The amount of delay injected by the long-haul simulator was varied between low (zero), medium (25 ms), and high (50 ms) values. The test was run for three flight conditions: straight & level C-5s, dynamic C-5s, each with a 1-meter dead reckoning error threshold, and a straight & level tanker with a 0.2-meter threshold.

***Results:*** Following are plots of instantaneous positional error for each of 1000 frames for all long-haul delay values and flight conditions.

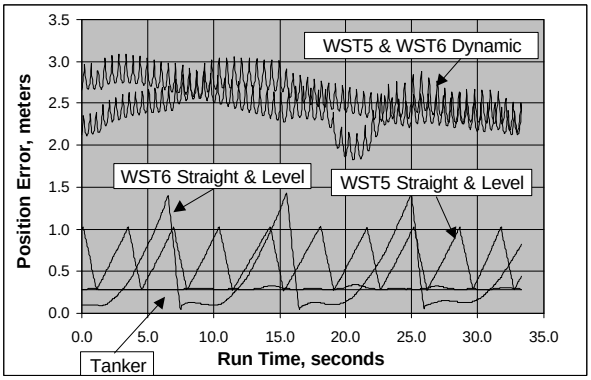


**Figure 10. Position Error, LAN Network (Zero long-haul latency)**

C-5 Straight & Level: For each of the charts (Figure 10, Figure 11, and Figure 12), the lower sawtooth plots are the two straight & level C-5s. One is WST5 as seen by WST6 and the other is WST6 as seen by WST5. The uphill portion of the ramp shows the error slowly increasing. Once it hits the 1-meter threshold, a new update is sent. Without smoothing, the error would instantaneously drop to near zero. Smoothing is enabled in these tests, so the updated position is smoothed over one second which can be seen in the downhill portion of the plots. The plots don't drop all the way to zero since after one second error has already built up somewhat.

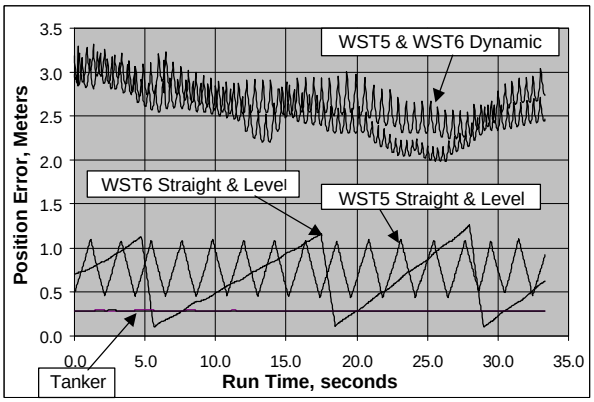


**Figure 11. Position Error, 25ms Average Network Latency**

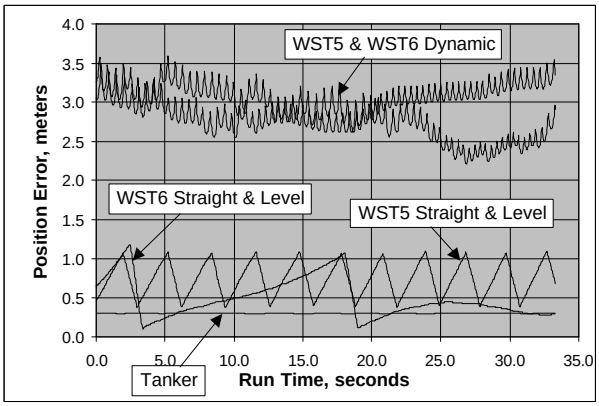


**Figure 12. Position Error, 50ms Avg Network Latency**

One can easily see the actual position update rate in these charts. For example, extrapolation works so well for the WST6 Straight & level case that it updates only every 10 to 15 seconds.

But mysteriously, WST5 updates more often. The reason was found to be in a fault in the DMT software in converting from the latitudinal and longitudinal velocities provided by the C-5 host software to the linear velocities required by the RPR FOM and by the dead reckoning algorithm. In an ellipsoidal Earth model, the north/south radius of the earth is not the same as the East/West radius. The DMT code failed to take this into account. The difference is only a few tenths of a percent but that is enough of an error in the

resulting linear velocity to cause the updates to occur five times more often.

C-5 Dynamic: The upper jagged plots in all three figures are the dynamic C-5s. It can be seen that updates occur fairly regularly at about twice per second. The fact that the error stays in the two to three meter range is an artifact of smoothing. With no smoothing, the error would grow to one meter and drop nearly to zero twice per second. That is, it would be a sawtooth waveform with a steep rise and near-instantaneous drop. The following diagram (Figure 13) shows what happens when smoothing is included.

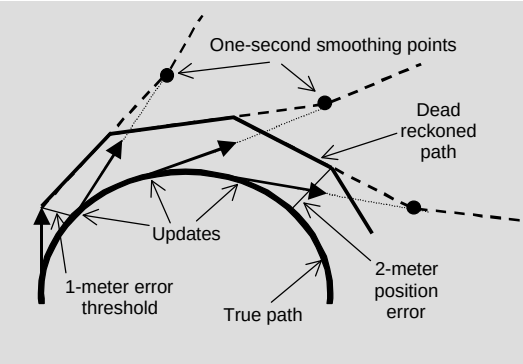


**Figure 13. Effect of Smoothing**

The dark curved line is the true path of the aircraft. The angular solid line is the dead reckoned path. Position updates are shown by arrows indicating the velocity direction at that instant. Since only aircraft velocity is used to dead reckon the position (accelerations were not readily available), the extrapolated path is always a straight line. The straight-line dead reckoned path and the curved true path must, of course, diverge and it takes about 0.5 seconds to diverge one meter. When a new update arrives the smoothing algorithm effectively picks a point on the new extrapolated path one second into the future and moves toward it along the dashed line in the diagram. If no new update arrives, after one second smoothing is finished and the extrapolated path is followed exactly, again showed by the dashed line. But in the dynamic case, the one-second point is never reached. A new update is received when only halfway there so again a new point is picked one second into the future and extrapolation moves toward it. As can be seen from the diagram, the net result is that the extrapolated path is always much more than one meter from the true path.

If the smoothing algorithm could know when the next update is going to arrive, it could smooth toward a point extrapolated at that time, reducing position error to the 1-meter range. Attempts have been made in the past to predict the next update arrival time based on the time elapsed between the previous updates for each entity. The results have been mixed for fighter aircraft because their update times change too abruptly. The predicted time is often wrong and the resulting visual effect is worse rather than better. Predictive smoothing might work better for larger cargo and tanker aircraft because their parameters change less rapidly. However, this leads to the complication of having different smoothing algorithms for different types of entities. The improvement of a meter or two in position error of a turning aircraft may not be worth the extra complication.

The use of acceleration (second order) dead reckoning should lessen the amount of position error induced by smoothing. With acceleration, the extrapolated smoothing point will be closer to the actual position. Position error should be reevaluated after acceleration is added before more radical approaches are tried.

Tanker: The tanker error is shown as the nearly perfect straight line at 0.27 meters of error. This error is near its 0.2-meter error threshold as expected. The surprise is that it does not have the sawtooth-shaped waveform like the C-5s have. It does explain why the visual tanker motion is extremely smooth, but it is not currently understood why there is no error buildup and decline at position updates. Apparently there is some combination of factors here that converge at the 0.2-meter threshold. This should be investigated further so that its results can be repeated in future phases of the program.

Trend: Since the position error plots show nearly identical results for three values of network delay, it is difficult to see any trend in position error due to varying delay.

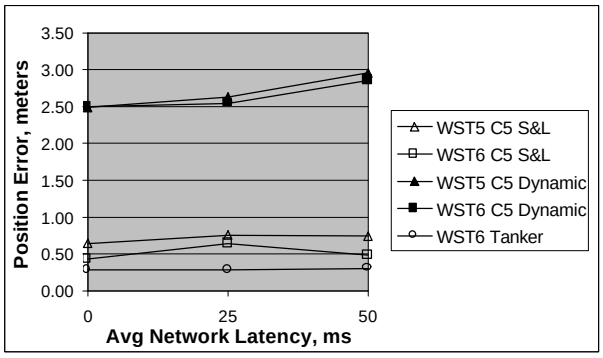


**Figure 14. Position Error vs Long-haul Latency**

To better see a trend, the position error for each test is averaged and plotted against average network latency (Figure 14). This plot shows that increased network

latency has no discernible effect on straight & level aircraft. The use of global time hardware to get an accurate elapsed time for dead reckoning completely eliminates the effects of latency for the straight & level case. The dead reckoning error threshold has far more impact on position error than network delay. Delay does have a small effect on position error in the dynamic case, with an additional 0.5 meter of average error incurred when average delays increase to 50 ms. Again, error threshold and smoothing are still the bigger factors in the position error.

## CONCLUSIONS

The project was deemed a resounding success by all involved. It was proven that HLA can be used to perform training missions. In addition, the current RTI was sufficient for real-time simulation of a small federation of dynamic aircraft.

Performance was a key consideration in deploying this HLA implementation. The DMSO RTI was not specifically developed to address the demands of highly dynamic real-time simulation and had not yet been released for any real-time operating system. Under favorable conditions, RTI performance was found to be sufficient and fairly deterministic. These conditions are the RTI running on a dedicated processor, federation size of a few 10's of entities, RTI bundling turned off, UDP polling interval set to 0, execution at real time priority, and the RTI ticked at a high rate (every 1 ms was used in these tests). Average RTI latency was around 2 ms; worst-case latency was measured at 5.3 ms.

Dead reckoning error and smoothing effects are no different than DIS. Also like DIS, absolute time stamps are crucial in getting smooth motion and low positional error of remotely generated entities. With the absolute time stamp, long-haul network delay had little affect on dead-reckoned position error. A 0.2-meter dead reckoning error threshold for the tanker gave excellent results for air refueling. Smooth tanker appearance is critical for air refueling training, and the pilots could not tell the difference between a locally generated tanker and one received via HLA through a long haul network with highly variable delay.

Extrapolated network bandwidth results indicate 200 cargo/tanker type aircraft could run on a T-1 line with no voice channels. A federation of 100 cargo/tanker entities on a single T1 long-haul network leaves enough bandwidth for 6 simultaneous voice channels, or a few more if no T1 bandwidth is lost to encryption.

However, with the current RTI, the T1 line will not be the limiting factor in scaling to lager federations. The one millisecond CPU utilization for each sending and receiving of HLA attributes through the RTI is much too high to allow federations with 100's of entities. This is especially a concern for aerial combat exercises with varied DMT assets including highly dynamic entities, bringing with them more attributes and interactions to accomplish the accurate simulation of weapons, data links, emissions, chaff, flare etc.

An RTI designed for real-time simulation of highly dynamic entities is obviously necessary to achieve DMT exercises containing 100's or 1000's of entities.

## REFERENCES


Bryant, R.B., Douglass Capt D.S., Ewart, R.B., and Slutz, G.J. (1994) "Dynamic Latency Measurements using the Simulator Network Analysis Project (SNAP)", 16th I/ITSEC Proceedings. Paper No.: 4-2.

IEEE Standard 1278.1-1995 Standard for Distributed Interactive Simulation - Application Protocols, Annex B

Monson, S. L., Johnston Capt. R. R., and Barnhart D.J. (1997) "Latency – The Adversary of Real-Time Distributed Simulation", 19th I/ITSEC Proceedings. Paper No.: 6-16, pp. 726-737.

Nemeth, D. (1999) "Benchmarking the RTI for Use in a Simulated Radio Environment" Proceedings of Spring 1999 Simulation Interoperability Workshop, Paper No. 46.

Swaine, S. D., and Marz, T. F. (1995), "DIS at Nine Gs", Proceedings of the DIS Workshop on Standards for the Interoperability of Defense Simulations, Institute for Simulation and Training, Orlando FL, September 18-22 1995, pp. 259-266.

Youmans, R. L. (1999). "Improving the Performance of HLA Applications By Multi-threading RTI Services Using the Proxy Design Pattern", Proceedings of Spring 1999 Simulation Interoperability Workshop, Paper No. 19.