AN HLA GATEWAY FOR DIS APPLICATIONS

Douglas D. Wood, Andy Cox, Mikel D. Petty
Institute for Simulation and Training
Orlando, FL

ABSTRACT

The High Level Architecture (HLA) is an architecture for constructing distributed simulations, including virtual, constructive, and live applications. The HLA is intended to facilitate interoperability among all types of simulation models and promote the reuse of simulation components. One of the possible approaches to transition legacy systems to the HLA is a stand-alone internetworking device to provide interoperability between DIS and HLA applications.

IST has previously reported the use of one such device, a prototype version of the IST HLA Gateway, to connect a SIMNET crewed M1 simulator to the HLA Platform Proto-Federation (PPF) using version 0.33 of the prototype Run Time Infrastructure (RTI). This paper presents the redesigned production Gateway, which translates between DIS and the Real-time Platform-level Reference (RPR) FOM. The Gateway uses a direct object-oriented interface to the RTI. Testing has shown that the average translation latency within the Gateway is less than 2 milliseconds.

ABOUT THE AUTHORS

Douglas D. Wood is a Research Computer Scientist at the Institute for Simulation and Training. Mr. Wood has performed research primarily in the area of distributed simulation, including HLA experiments, electronic warfare protocols, and algorithms for computer generated forces. He has also performed research in simulation for emergency management training. Mr. Wood received a M.S. and a B.S. in Computer Science from the University of Central Florida. His research interests are in simulation and distributed systems.

Andy Cox is an Associate Computer Scientist at the Institute for Simulation and Training. He is currently involved in various projects in the areas of Distributed Interactive Simulation (DIS) and the High Level Architecture. Mr. Cox received a B.S. in Computer Science from the University of Central Florida and is currently pursuing a graduate degree. His background includes prior military service as an Infantryman, Quartermaster Officer, and Military Police Officer. His research interests are in distributed simulation and internetworking.

Mikel D. Petty is a Senior Research Computer Scientist at the Institute for Simulation and Training. He has led IST research in Computer Generated Forces, Emergency Management, and High Level Architecture. Mr. Petty received a B.S. in Computer Science from the California State University Sacramento, a M.S. in Computer Science from the University of Central Florida, and is currently a Ph.D. Candidate in Computer Science, also at UCF. His research interests are simulation and computational geometry.

AN HLA GATEWAY FOR DIS APPLICATIONS

Douglas D. Wood, Andy Cox, Mikel D. Petty
Institute for Simulation and Training
Orlando, FL

INTRODUCTION

Background

HLA is a standard for constructing distributed simulations. It is intended to facilitate interoperation between a wide range of simulation types and to promote reusability of simulation software. HLA will encompass virtual, constructive, and live simulations from the training, engineering, and analytic domains, including simulations not from the Distributed Interactive Simulation (DIS) community, such as logical time and aggregate level simulations.

HLA consists of a number of interrelated components. The three that define HLA are: (1) the HLA Rules, which define interoperability and what capabilities a simulation must have to achieve it [DMSO,1996]; (2) the Object Model Template, which is a semi-formal methodology for specifying simulation and federation object classes, attributes, and interactions [DMSO,1997b]; and (3) the Interface Specification, a precise specification of the functional actions or services that a simulation may invoke, or be asked to perform, during an HLA exercise [DMSO,1997a]. The services are organized into six groups: (1) Federation Management, (2) Declaration Management, (3) Object Management, (4) Ownership Management, (5) Time Management, and (6) Data Distribution Management. Services in the Object Management group actually communicate simulation data among interoperating simulations.

A group of simulations interoperating under HLA is called a federation, and one of the simulations is a federate. A federation must agree upon a common federation object model (FOM) that defines the objects to be simulated, their attributes, and their interactions. The RTI transports only data (object attributes and interactions) that has been defined in the federation's FOM.

The HLA Run-Time Infrastructure (RTI) is a software embodiment of the Interface Specification that implements the actions defined therein to be usable by a simulation, provided as a set of services. The RTI's Application Programming Interface (API) is defined by two objects [DMSO, 1997c]. The *RTI Ambassador* is an object through which the federate invokes services on the RTI and thus communicates simulation and control data to the federation. The *Federate Ambassador* is an object through which the RTI invokes services on the federate and thus communicates simulation and control data to the federate. The implementation of the RTI Ambassador is supplied by the RTI library. The implementation of the Federate Ambassador is supplied by the federate.

HLA Interoperability Options

When considered in detail there are many approaches to make an existing simulation HLA interoperable, but in general terms there are only two: (1) modify the existing simulation to communicate using HLA, or (2) leave the existing simulation unmodified and use an external gateway to translate the simulation's communications to and from HLA. We refer to the first approach as "integration" and the second as "gateway.

Interoperability via Integration. For the integration option, the existing simulation is modified so that it communicates using HLA. That means that the simulation must send and receive object attribute updates and interactions that have been defined in a FOM using the appropriate RTI services. It also means that the simulation must be modified to use other RTI services beyond those for communicating attribute updates and interactions, such as Federation Management to join the federation

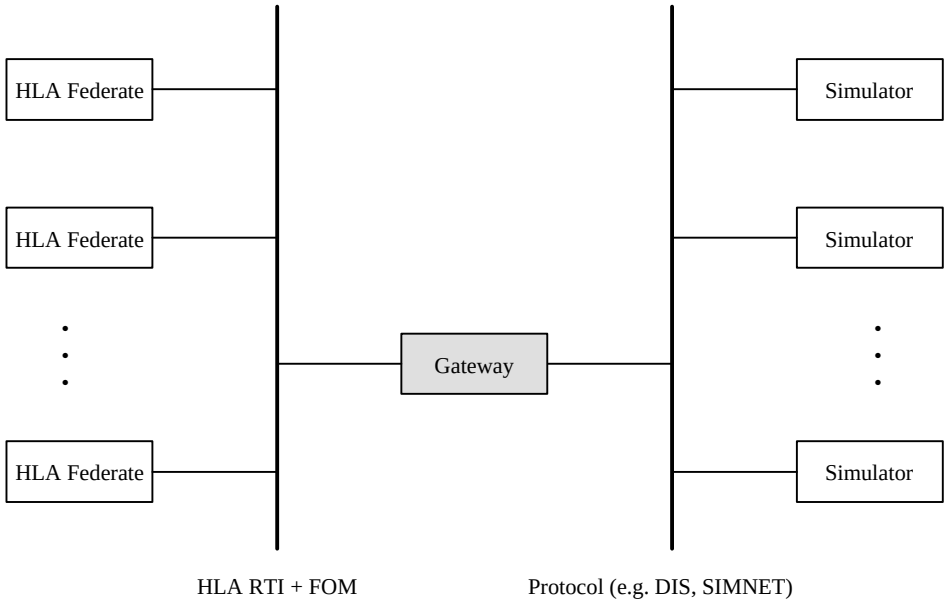

Figure 1 Gateway interoperability

execution, Declaration Management to declare what object classes the simulation will simulate, and so on.

Interoperability via Gateway. The integration option requires changes to the existing simulation. The alternative is to not modify the existing simulation, using a gateway instead. The gateway approach provides an interface node between an HLA Federation and the legacy simulation network protocol (see Figure 1). One or more simulations using an existing protocol, such as DIS, SIMNET, or ALSP are connected via the gateway to a HLA federation. On the protocol side of the gateway, data packets (PDUs for DIS) are formatted, sent, and received according to the protocol. The gateway receives the packets and translates them at two levels: (1) the data in the packets is converted into the data formats defined in the FOM, and (2) the sequence of packets are translated into corresponding RTI service invocations. As an example of the latter, the first DIS PDU received for a simulator triggers not only an Update Attribute Values service but might be preceded by a Join Federation service produced by the gateway on behalf of the DIS simulator. In general terms, the gateway translates and resends traffic that is relevant to the other side, and provides necessary functions needed on one side that have no analog on the other. The details of the translation and service functions are highly dependent on the FOM and the protocol involved.

IST HLA Gateway

The gateway approach was undertaken by the Institute for simulation and Training (IST) under the sponsorship of the US Army Simulation, Training and Instrumentation Command (STRICOM). This approach was first investigated during the HLA Platform Proto-Federation experiments [Cox, 1996a, Cox, 1996b, Harkrider, 1996a, Harkrider, 1996b, Wood, 1997]. The current IST HLA Gateway will initially support translation between an HLA Federation based on the Real-time Platform Reference (RPR) FOM and DIS (2.0.3, 2.0.4, or 1278.1) or SIMNET 6.6.1 [SISO-RPR, 1997, Pope, 1991, IEEE, 1995].

The Gateway is an adaptation of the IST Computer Generated Forces (CGF) Testbed [Smith, 1992]. The CGF Testbed provided a SIMNET and DIS compliant framework for coordinate conversions, dead reckoning, visual displays, and a protocol-independent simulation core. The adaptation of the CGF Testbed required the addition of a Gateway Manager and an HLA RTI Interface. The Gateway Manager

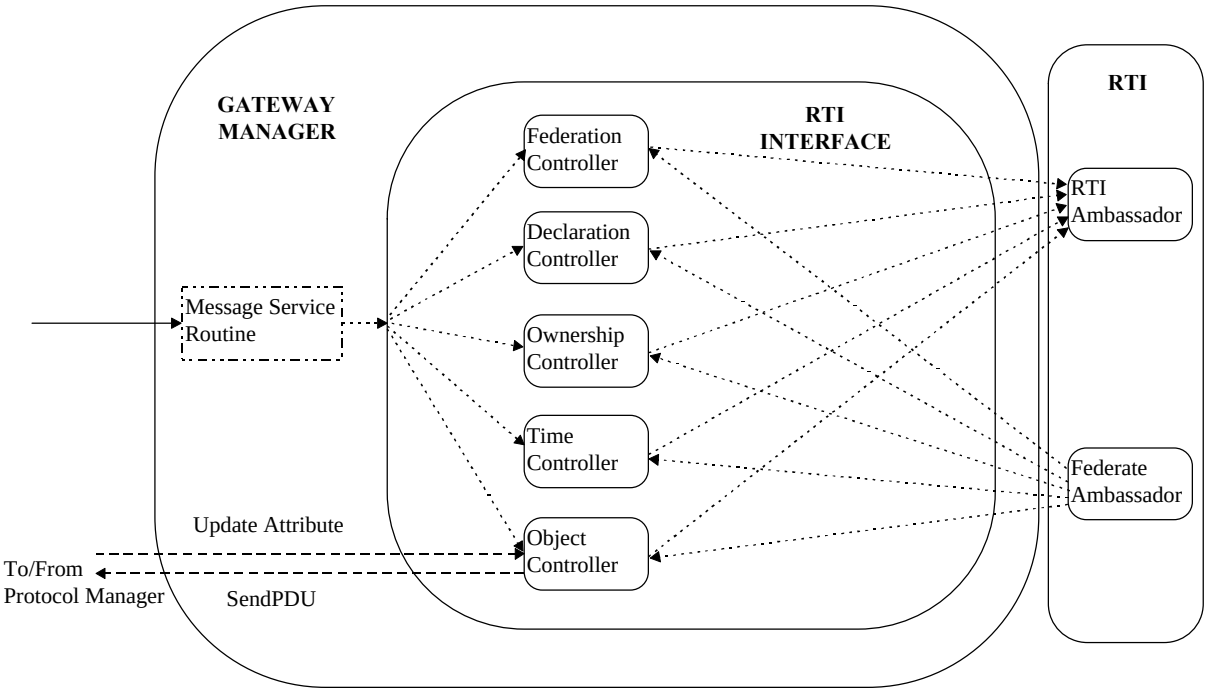

Figure 2 RTI Interface data flow

component provides high level support for the translation between the two network simulation architectures. It supports those parts of each network which have no counterpart in the other, such as creating/destroying and joining/resigning federations in HLA or heartbeat PDUs in DIS. The HLA RTI Interface provides the low level support for the HLA RTI calls in a fashion that insulates the Gateway from the RTI and the HLA FOM implementation details. The RTI Interface follows an object-oriented design that mirrors both the HLA Interface Specification and the target HLA OMT FOM.

RTI INTERFACE DESIGN

The RTI Interface was designed using object-oriented principles and implemented using the C++ programming language. An RTI Interface class was defined to mirror the RTI API's centralized RTI Ambassador services and to internally handle the RTI's Federate Ambassador invocations [DMSO, 1997c]. The names of the functions presented by the RTI Interface match those of the RTI Ambassador. However, some of the functions were hidden within the interface and the service parameters were simplified and changed to match the data coming from user input and the integrating simulation system. The RTI Interface handles the translation of the incoming data into the format required by the RTI API and the targeted FOM.

The implementation of the RTI Interface is broken down into component Controller classes that compartmentalize the centralized services into their respective HLA Interface Specification service groups (e.g. Federation Management, Declaration Management, and Object Management). The RTI interface is simply a package for these Controller objects. Each of the Controller objects handle the service calls initiated from both the simulation system and the Federate Ambassador (see Figure 2). The RTI Interface's implementation of the RTI API's Federate Ambassador simply makes the identical call on the appropriate RTI Interface Controller object.

The concept of defining Controller classes to handle RTI services calls directly correlates the RTI Interface design with the HLA Interface Specification. Another principal design feature of the RTI Interface correlates the handling of objects and interactions with the targeted FOM class hierarchy. An Object Actor class hierarchy

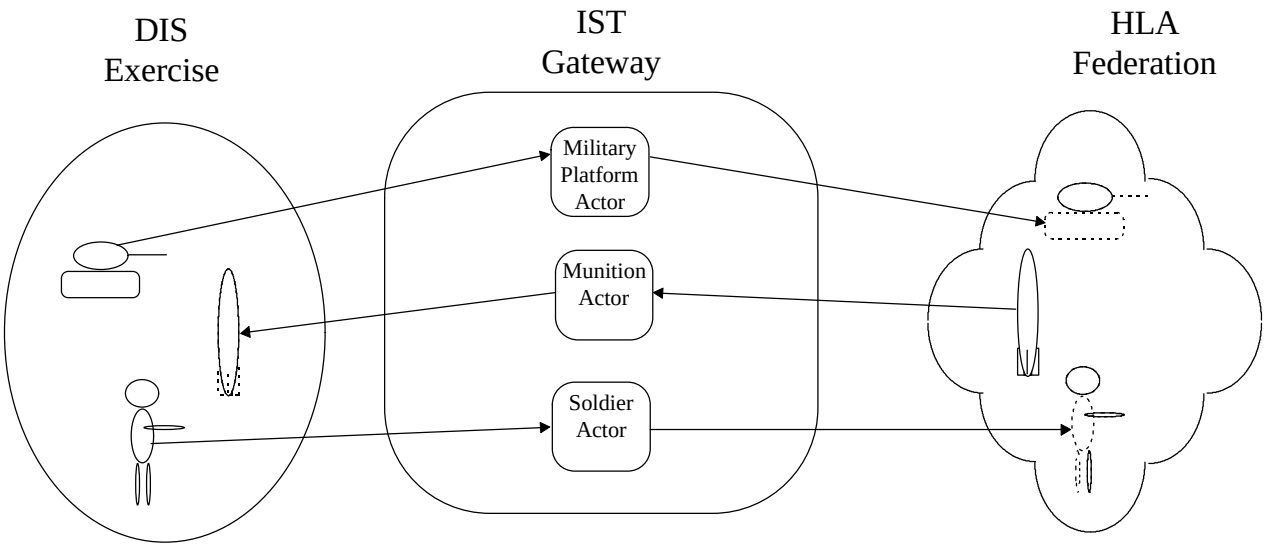

Figure 3 Entity-Actor relationship

is defined that essentially mirrors the targeted FOM object class hierarchy. Each derived Object Actor class handles the data transformations for the attributes defined in the associated FOM object class. An Object Actor is instantiated for each registered and discovered object to maintain its state (see Figure 3). The entity state is necessary because the Gateway must send only changed data (compare old against new) and will most likely receive partial updates (to be applied to dead reckoned state). Interactions are treated somewhat differently than objects because each one operates on a completely different set of data and an interaction has a transient life span. Only one Interaction Actor instance is needed for each FOM class. The idea of using Object and Interaction Actors was introduced in the JPSD Federate Common Software [Briggs, 1996].

RTI Interface Controller classes

The design of the RTI Interface calls for the implementation of a Controller class for each of the HLA Interface Specification service groups. The Federation Management and Declaration Management Controller classes are straight forward implementations that simply forward service calls to the RTI Ambassador. At the time that this paper was written, Time, Ownership, and Data Distribution Management were not implemented other than as empty functions to handle RTI Federate Ambassador invocations.

The Object Controller class splits the implementation of the Object Management services between itself and the Object Actor and Interaction Actor classes. The services that act on objects (e.g. register, discover, delete/remove) are implemented by the Object Controller. The services that act on attributes (e.g. update/reflect, change transport/order type) or interactions (e.g. send/receive interaction) are implemented by the Object Actor and Interaction Actor classes. The Object Controller also manages the Object Actor instance lists and supplies functions for accessing Object Actors.

Insulation from FOM specification

The Actor Class hierarchy insulates the RTI Interface from the FOM specification. Each of the Controller classes are implemented without any hard coding of FOM identifiers or data types. In addition, a mapping module defines a mapping between internal and RTI class, attribute, and parameter identifiers or handles (used to identify what is being transmitted). The mapping module can be modified without affecting the RTI Interface implementation. By keeping the FOM specification separate from the basic RTI Interface implementation,

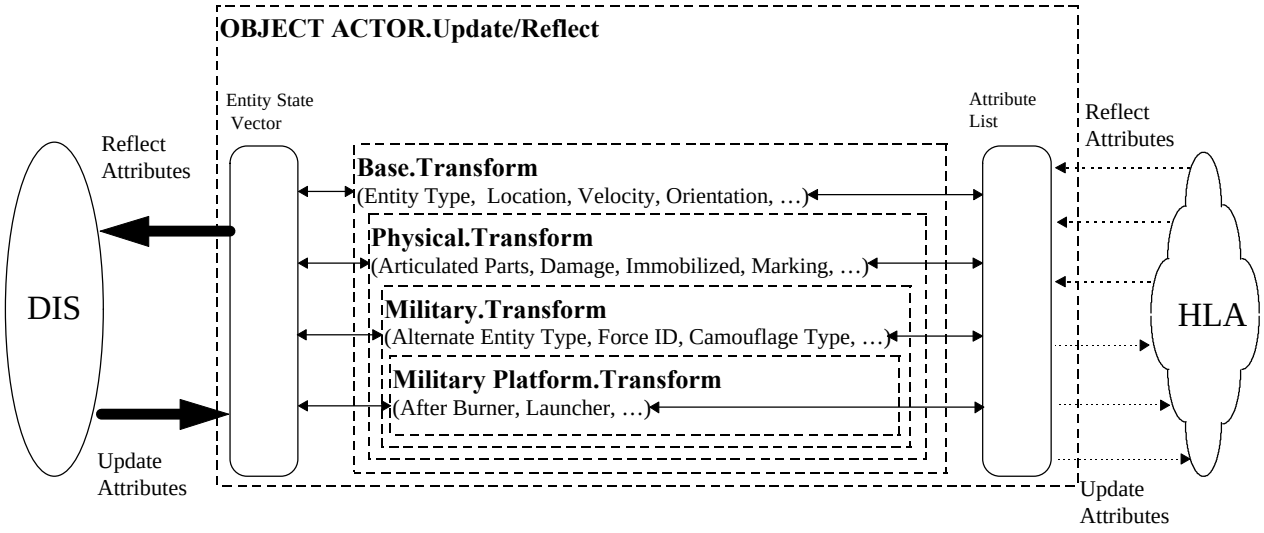

Figure 4 Data translation within Actor class derivation

changes to the FOM have little or no affect on other parts of the Gateway.

FOM Class Actors

The attribute/parameter based Object Management services are implemented by Object Actor and Interaction Actor classes. These classes are defined through class abstraction, where the abstract Actor class implements the service calls through the use of virtually defined functions to perform handle mapping and data transformations. The Actor class derivation mirrors the FOM class derivation. Each derived Actor class specifies the attributes/parameters associated with its FOM class and implements the mapping and transformation functions which act upon them (see Figure 4). The derived Actor classes inherit the service calls (and various entity state data helper functions); only the mapping and transformation functions are redefined. Changes to the FOM do not affect the abstract classes; only the derived (FOM specific) classes need be modified.

DATA STANDARDIZATION

One of the key elements of the Federation Development and Execution Process (FEDEP) Model is the generation of a Federation Object Model (FOM) specifying the commonly understood object classes, attributes, interaction classes, parameters, data types, and additional information in a format consistent with the Object Model Template (OMT).

The Real-time Platform Reference (RPR) FOM is largely an expression of the DIS protocols and enumerations in the format defined by the OMT. The RPR FOM is a work in progress to evaluate the feasibility of defining standard “reference” FOMs that may take advantage of data standardization efforts and reduce the level of effort required to achieve interoperability among federations. Several organizations such as IST, NAWC-TSD, and GEC-Marconi (UK) have already implemented HLA-based simulation applications which utilize developmental versions of the RPR FOM. In many cases these HLA federates are adaptations of legacy DIS systems, and the RPR FOM’s similarity to DIS significantly reduces the effort required to design an appropriate Simulation Object Model (SOM) or FOM. The Gateway supports the RPR FOM. By doing so, the Gateway is interoperable with HLA federations using the RPR FOM as their common data model.

PERFORMANCE ANALYSIS

The use of a Gateway introduces an additional source of latency beyond that of a direct RTI interface. In a federation consisting of DIS applications, a Gateway, and HLA federates, the use of a Gateway would in general add two components to the performance of native RTI communication: 1) one internal Gateway latency and 2) one DIS transmission latency.

One of the objectives of the Gateway development is to provide a detailed performance analysis to ascertain the factors associated with the use of a gateway for legacy system HLA integration. Primary measures of Gateway performance include:

- Message processing time (internal latency)
- Throughput
- Scalability

Internal Latency is defined as the time required to completely process either of the following sequences of events:

1. HLA to DIS: Gateway receives RTI *reflectAttributeValue* or *receiveInteraction* call, builds DIS PDU, sends DIS PDU to network.

2. DIS to HLA: Gateway receives DIS PDU, converts to FOM data representation, invokes RTI *updateAttributeValues* or *sendInteraction.*

Throughput is determined by the maximum rate of DIS PDUs and RTI service calls which can be processed within acceptable thresholds for latency, reliability, and resources. The specific value of these thresholds may vary based on the requirements of a particular federation.

Scalability is the ability to support large numbers of DIS entities and HLA objects, possibly involving multiple processors, or a scheme for distributed processing among several gateways. It is likely that for federations distributed across a Wide Area Network (WAN), a Gateway would be situated at each local network to eliminate the requirement to incur additional transmission latency across the WAN.

Measures of native RTI performance are of interest to determine the relative impact of the

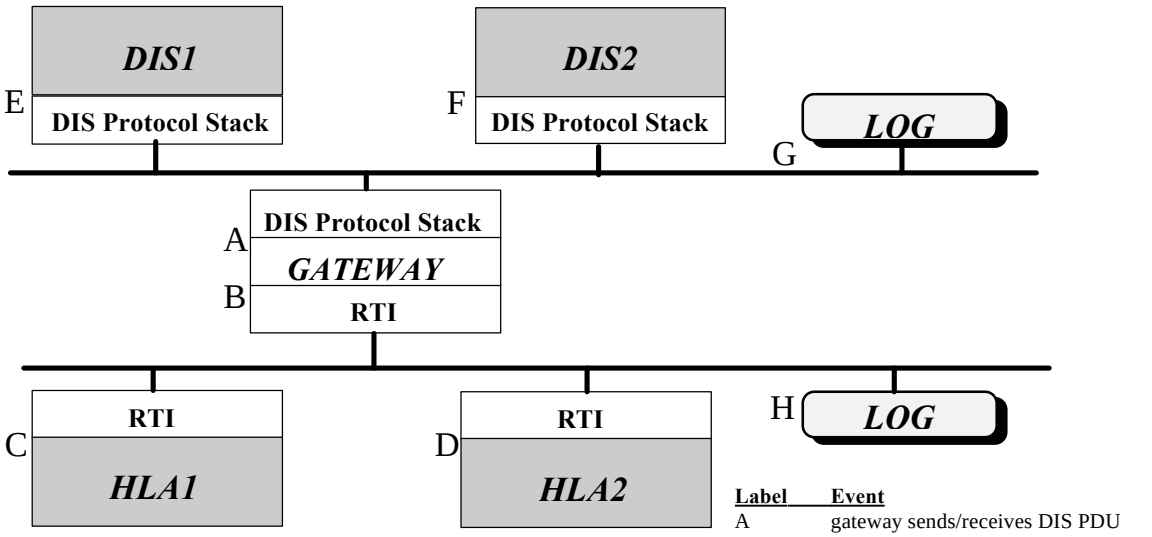

Figure 5 Gateway performance experiment configuration

Gateway in comparison to direct RTI integration. Key elements of RTI performance include:

- Message Latency
- Message Throughput
- CPU Utilization

Figure 5 depicts the standard configuration for performance tests to be conducted on a local network. The labels A through J represent points at which key events occur. The data logging at points G and H refer to DIS application data loggers or network performance monitors, collecting data on as DIS PDU transmission rates, network bandwidth usage, multicast group usage, and multicast datagram transmission rates.

Host	Platform	CPU	OS
DIS1 (CGF)	SGI Indy	R4400-150MHz	IRIX 6.2
DIS2 (CGF)	Dell PC	486/66	DOS 6.2
GATEWAY	SGI 02	R10K/150	IRIX 6.3
HLA1(CGF)	SGI 02	R5K/180	IRIX 6.3
HLA2(CGF)	SGI Indy	R4600/133	IRIX 6.2

Table 1 Federate information

Table 1 shows information on the composition of the Gateway Integration, Analysis, and Network Test (GIANT) federation used for performance analysis. Host names refer to Figure 5. The use of two physically distinct networks is commonly used to isolate the DIS and RTI networks, exchanging data through the Gateway. To support this, the workstation running the Gateway software is configured with two network interfaces. Alternatively, the DIS applications, HLA federates, and Gateway may all reside on a single network.

GATEWAY PERFORMANCE

Federation executions conducted during performance experiments consist of HLA-based Computer Generated Forces (CGF) applications connected to DIS CGF via a single Gateway. Scripted scenarios were developed to provide varying levels of DIS and RTI network traffic. RTI version 1.0 (IRIX port) was used for all measurements involving RTI performance.

To evaluate the Gateway, DIS/HLA Federations were constructed consisting of from 2 to 150 simulation objects, primarily M1/T72 tanks, BMP/M2 AFVs, and AH64/MI28 attack helicopters. Fire and Detonation interactions consisted of direct fire 120/125MM HEAT rounds, which were not instantiated as objects, and tracked anti-armor munitions which required the instantiation of dynamic simulation objects, typically of short duration. The Gateway software was modified to record data on internal processing and to calculate RTI end to end latency. Preliminary results indicate that the Gateway requires an average of approximately 0.7 milliseconds (ms) to translate a DIS Entity State PDU into RPR FOM attributes and send an RTI update, or approximately 0.9ms to translate a DIS Fire, Detonation, or Collision PDU into a corresponding RTI interaction. RTI service calls are converted into DIS PDUs with a similar penalty of approximately 1 millisecond per event. In many distributed simulation applications, additional latency on the order of 10 milliseconds would not significantly effect performance.

RTI PERFORMANCE

As this is written, there are no standard benchmarks to evaluate RTI performance, although work to define such benchmarks is in progress. Performance measurements of the RTI are useful when presented in the context of the federation under which they are collected. All measurements in this section refer to the federation characterized in the previous section.

Initial measurements in these experiments indicate that RTI 1.0 performance in best effort delivery of attribute updates and interactions is roughly equivalent to DIS performance for the Federation Executions conducted in these tests. The measurements show that attribute updates and interactions are delivered with an average of 2-8 ms latency on a local network under favorable network conditions. This result is comparable to the performance of DIS PDUs broadcast using UDP datagrams under similar conditions. However, the RTI delivery appears to have a wider variance than what is commonly seen in DIS. We observed occasional “spikes” where updates and interactions had RTI latencies measuring greater than one second.

The cause for these large latencies were undetermined and are still under investigation.

FUTURE WORK

The development of the Gateway's functionality is continuing. The functionality and performance described herein is associated with the version 1.0 Gateway, released in July 1997. A second major release will have been made in late October 1997. The second release enhances and expands the support for the RTI services. Because the RTI version 1.1, which adds support for Data Distribution Management (DDM), is not scheduled to be released until December 1997, the October release of the Gateway will not include DDM services. The second release will support one or more additional DIS PDU families such as SIMAN, Radio, Emissions, or Logistics. As in DIS, the transmission of these simulation components present different requirements and stresses on applications and the simulation network. Experiments using these additional HLA services and PDUs will be conducted.

The Gateway's design is inherently adaptable to different HLA FOMs. Modification or replacement of the FOM is only reflected in changes to the FOM Actor classes. One intent of extended Gateway development is to build up a set of FOM Actor libraries. These FOM Actor libraries would provide Gateway configuration for different HLA FOMs.

SUMMARY AND CONCLUSIONS

The gateway approach can make it possible for some existing simulations to achieve HLA interoperability without modification. The IST HLA Gateway serves that purpose, providing interoperability between DIS simulations and HLA federations based on the RRP FOM. The additional latency imposed by the Gateway is only 1-2 milliseconds, well within the range of acceptability for many simulation applications.

ACKNOWLEDGMENTS

The support of the U. S. Army Simulation, Training, and Instrumentation Command under contract N61339-96-C-0035, supervised by Susan M. Harkrider, is gratefully acknowledged.

REFERENCES

Briggs, R., Miller, G., "The JPSD Experiment Federation Common Software", *Proceedings of the 15th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Orlando FL, September 16-20 1996, pp. 625-629.

Cox, A., Wood D. D., Petty, M. D., and Juge, K. A. (1996a). "Integrating DIS and SIMNET Simulations into HLA with a Gateway", *Proceedings of the 15th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Orlando FL, September 16-20 1996, pp. 517-525.

Cox, A., and Wood, D. D. (1996b). "Design and Implementation of the BDS-D/HLA Gateway", *Proceedings of the 18th Interservice/Industry Training Systems and Education Conference*, Orlando FL, December 3-6 1996.

Defense Modeling and Simulation Office (DMSO) (1996). "Department of Defense High Level Architecture Rules Version 1.0", Online document at URL http://www.dmso.mil/projects/hla/, August 15 1996.

Defense Modeling and Simulation Office (DMSO) (1997a). "Department of Defense High Level Architecture Interface Specification Version 1.1", Online document at URL http://www.dmso.mil/projects/hla/, February 12 1997.

Defense Modeling and Simulation Office (DMSO) (1997b). "Department of Defense High Level Architecture Object Model Template Version 1.1", Online document at URL http://www.dmso.mil/projects/hla/, February 12 1997.

DMSO, MITRE, SAIC, Virtual Technology Corporation (1997c). “High Level Architecture Run-Time Infrastructure Programmer’s Guide, Version 1.0”, 15, May 1997

Harkrider, S. M. and Petty, M. D. (1996a). “Results of the HLA Platform Proto-Federation Experiment”, *Proceedings of the 15th DIS Workshop on Standards for the Interoperability of Defense Simulations*, Orlando FL, September 16-20 1996, pp. 441-450.

Harkrider, S. M. and Petty, M. D. (1996b). “High Level Architecture and the Platform Proto-Federation”, *Proceedings of the 18th Interservice/Industry Training Systems and Education Conference*, Orlando FL, December 3-6 1996.

IEEE (1995), Standard for Distributed Interactive Simulation—Application Protocols, IEEE Standards Department, Piscataway, NJ, 1995

Pope, A. R. The SIMNET Network and Protocols. (1991) Version 6.6.1. BBN.

SISO-RPR (1997), Real-Time Platform Reference FOM, Simulation Interoperability Standards Organization Reflector - Plug and Play Family of Models, http://chat.sc.ist.ucf.edu/confs/SISO-RPR, 1997

Smith, S. H., Karr, C. R., Petty, M. D., Franceschini, R. W., & Watkins, J. E. (1992) “The IST Semi-Automated Forces Testbed.” IST

Wood, D. D., Petty, M. D., Cox, A., Hofer, R., and Harkrider, S. M. (1997). “HLA Gateway Status and Future Plans”, *Proceedings of the 1997 Spring Simulation Interoperability Workshop*, Orlando FL, March 3-7, 1997, pp. 807-814.