Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

HIGH LEVEL ARCHITECTURE REMOTE DATA FILTERING

William S. Murphy Jr.

TRADOC Analysis Center-Monterey
Post Office Box 8692
Monterey, CA 93943, U.S.A.

Galen D. Aswegan

Tapestry Solutions, Incorporated
5675 Ruffin Road, Suite 305
San Diego, CA 92123, U.S.A.

ABSTRACT

The current structure of the High Level Architecture (HLA) puts a tremendous burden on network load and CPU utilization for large distributed simulations due to its limited controls for publishing and subscribing object updates and interactions. Several solutions to this problem have been proposed, but all require cooperation between federate developers and FOM extensions for both the publishing and subscribing Federates. This paper explores an alternative that has the potential to dramatically reduce communications load by allowing subscribing federates to extend and control the publish/subscribe mechanisms that are local to the publishing federate's process without requiring changes to the publishing federate or the Federation Object Model (FOM).

1 INTRODUCTION

United States Department of Defense (DoD) policy mandates the HLA as the common technical framework to facilitate interoperability between DoD simulations. New simulations being developed must implement the HLA/RTI communications architecture (U.S. Department of Defense, 1997). Legacy simulations must be modified to operate in the HLA framework or be retired. Data volume and network traffic problems have been identified by several HLA user communities.

The problems revolve around the methods under which subscriber federates determine which objects, attributes, and interactions they want to receive from a publisher federate, and on the frequency of object updates while the object is changing. Several Data Distribution Management (DDM) Service extensions to the HLA have been proposed that attempt to limit the number of objects published and the frequency of object updates (U.S. Department of Defense, 1997).

The proposed methods involve defining Routing Spaces and Subscription Regions that attempt to classify objects within certain regions, and allow subscribing federates to limit the regions in which they receive updates. While this helps alleviate some of the communication problems, it imposes a burden on the publishing federates to implement classification by region, and does not solve the more general problem of arbitrary filters or filtering logic that is discussed in this paper.

This paper introduces filtering techniques and extensions to the HLA Run Time Infrastructure (RTI) specification that will reduce network data traffic by allowing subscriber federates to define arbitrary filters or filtering logic, distribute the logic through the RTI to a publisher federate, and execute the logic in the publisher federate's local RTI component. Once the decision is made to publish an object or update an attribute, extensions to the RTI's protocol will allow directed RTI communication to take place between the publishing federate's local RTI logic component and the subscriber Federate. This paper proposes a methodology that takes the systems approach to network communications and logic distribution in order to eliminate network traffic without adversely impacting the performance of the local federates.

2 BACKGROUND

The ideas in this paper are an extension of the Analysis Federate research that was conducted by the United States Army Training and Doctrine Command (TRADOC) Analysis Center (TRAC) in Monterey, California. RTI inefficiencies forced researchers and analysts to subscribe to and receive significant amounts of attribute and interaction updates on a continuous basis for data that was only required on a conditional basis during limited time periods. The combination of requirements were not simple value or proximity based filters that are currently addressed by DDM, but custom decision making based on a history of prior observations combined with user requests for analysis information.

2.1 HLA/RTI Services

The HLA communications framework is implemented on a subscription basis during federation execution by the RTI. Simulations (federates) publish only the data that has been

subscribed to by one or more other federates. Publishing federates transmit the subscribed data to the RTI executive process which filters the data and routes it to the appropriate subscribing federates.

In its simplest form, a subscriber federate can subscribe to a specific object "class", and have the RTI notify the federate whenever a new object of the class is "discovered". At this level, all objects of the class are discovered, and all updates to registered attributes of the class are delivered to the federate for processing. Selective discovery of individual objects is not possible. Instead, unneeded objects can only be ignored after they have been delivered to the federate (i.e. after all the overhead of the RTI and network has been incurred).

After an object has been discovered, the subscriber federate can receive updates to the object whenever its attributes are changed. The publishing federate notifies the RTI whenever an object's attributes change value. The RTI then notifies the subscriber federates of the changes. If the subscriber does not require all the updates (due to difference in resolution, timing, precision, etc.) it still must receive the data, and throw it away (i.e. again after the overhead of the RTI and network has taken place).

2.2 HLA Object and Attribute Filtering

To reduce communications requirements in HLA federations, at least three different data-filtering approaches are currently being pursued. These include the RTI's DDM services, Professor Bernard P. Zeigler's predictive filtering research, and Mr. Keith Briggs' requestable object research. The three approaches implement specific data filtering techniques based on a specific perceived user need.

2.2.1 Data Distribution Management (DDM)

To provide some controls for communication reductions, subscribing and publishing federates were given certain controls in the form of DDM extensions to the RTI. The controls can be divided into two basic types: value filtering for update value resolution and proximity filtering for limiting object discovery. The techniques proposed involve defining routing spaces as multidimensional coordinate systems in which federates express an interest for either receiving data or sending data by regions within the routing space. A publishing federate first defines regions within a routing space, and then assigns object classes to the various regions. Subscribers also define regions as areas of interest in which they wish to receive objects, attributes, and interactions. The RTI determines which update regions overlap with the subscription regions, and send only those objects that overlap to the subscribing federates.

The first major problem with routing spaces is that the federates must first agree on which routing spaces and dimensions are to be supported by the publishing federate. It is impossible to specify a new region with a specific dimension without involving changes to the publishing federate's source code.

A second major problem with routing spaces is that it only determines overlap of publish/subscribe regions, not of individual objects relative to a subscription region. Subscribing federates may still receive objects outside their specified region if the object is within a publisher's region, but outside the subscription region.

A third major problem is that there are no controls on the resolution of the attribute values within the regions. Once the overlap test passes for an attribute update, all values are sent to the subscriber. The subscriber has no controls on the amount of change of an attribute before it's process is notified.

2.2.2 Predictive Filtering

Professor Bernard P. Zeigler has done research in the area of predictive filtering, which is basically an extension of the DDM value filtering technique described above (Zeigler and Lee, 1998). It attempts to define and parameterize more sophisticated filters that can evaluate attribute value updates from a publisher federate and predict when a subscriber needs the value updates.

Once an object has been discovered, predictive filters can be attached using interactions sent through the RTI so that updates to the subscriber are only received once the filters are satisfied.

2.2.3 Requestable Object

The paper published by Keith Briggs entitled "Extending the HLA to support a Requestable Object Service", describes a technique whereby specific objects can be published and discovered through extensions to the FOM and interaction services based on parameters specified by the subscriber federate (Briggs, et al, 1997)

The technique dramatically reduces object traffic when large number of objects exist (such as mines), but are sparsely used by other federates. The technique allows a subscribing federate to issue an interaction which contains the filtering parameters required by the subscriber. The publishing federate is responsible for evaluating the filters and publishing only those objects which pass the filtering criteria. The types of filters supported are negotiated between the subscribing and publishing federates, and are encapsulated as extensions to the FOM.

3 AN ABSTRACTION OF THE PROBLEM

The filtering techniques proposed to date are all solving the same basic problem. By looking at the roles, responsibilities, and needs of each federate and the RTI, it is possible to abstract the problem into a more general issue of flow of logic and process control. First, the responsibilities of data filtering **must** be under the control of the subscribing federate because it is the only process that knows how and why the filtering must take place. Secondly, to reduce network bandwidth and RTI overhead, the filtering must be done in the publishing federate's process (or its local processor) before stimulating the network communications. Third, each subscriber federate may have unique requirements that are completely unrelated, but may overlap with other federates. Fourth, the publishing federate needs to know that someone is subscribed, but should be sheltered from the filtering mechanics.

The last issue is probably the most important to the overall architecture. If the federate is not isolated from the filtering, its functionality must be modified for each new subscriber federate. Since two federates are now involved, extensions to the FOM and potentially the HLA/RTI are required to instantiate and control a new filtering technique. If not addressed, the overall management of the simulation environment and development process will be a paperwork and procedural nightmare.

A summary of the current roles of each Federate and the RTI in the overall process follows. This paper does not propose to eliminate the routing space functionality, or any other RTI functionality. Instead, this proposal seeks to extend the current RTI functionality to make it more responsive to user needs.

3.1 Publishing Federate

The role of the publishing federate is to respond to general requests from the RTI to publish its objects and interactions, and to respond to requests for republication if necessary. Ideally, the publisher needs to keep track on an object-by-object basis whether anyone is subscribed or not. The publisher should not need to keep track if more than one subscriber is attached, or if additional filtering is being performed after it publishes its objects or attribute updates. If no federates are subscribed to particular objects, the publisher federate has the ability to turn these objects or their algorithms off for performance reasons.

3.2 Subscribing Federate

The subscriber's role is to subscribe to its required objects, attributes, and interactions. In addition, it must specify any additional filtering requirements that must take place remotely in the publishing federate's process. In a completely general case, the subscriber should be able to provide an arbitrary block of logic that is invoked whenever the publishing federate publishes a new object, or updates an existing object's attributes. The block of logic must be able to run in the publishing federate's process, must be able to communicate to the local RTI, and must be able to communicate through the RTI back to the subscriber's process.

3.3 RTI Executive

The RTI has the responsibility of notifying each federate when subscribers have subscribed to individual objects, attributes, and interactions. It also must be responsible for registering a subscriber's filtering logic, for distributing the logic to the federate's process, and for invoking the logic when the publishing federate notifies the RTI of new objects and updates. In addition, the RTI is responsible for coordinating requests from multiple subscribers for the purposes of minimizing network communications. Lastly, the RTI must reliably deliver the discoveries, interactions and object updates to the appropriate subscribing federates.

4 REMOTE LOGIC FILTERING PROPOSAL

The key to providing a general implementation of the roles previously described is the distribution of arbitrary logic from the subscriber process to the publishing federate's process. With traditional programming languages, such as C, C++, Fortran, ADA, etc. this is impossible without doing a recompilation of the publishing federate's application. With newer technologies, such as JAVA and Tapestry's VISION XXI/Kernel Technology, it is possible to distribute arbitrary logic blocks in an "uncompiled" or neutral binary form, able to run in any process (including remote). As long as the appropriate interpreter is built into the local component of the RTI that is linked with the publishing federate's application, the logic can be invoked under the control of the publishing federate's local RTI component.

The architecture proposed accomplishes the following:

- It allows any arbitrary filtering technique the subscriber desires to be distributed.
- It completely isolates the publishing federate from the subscriber requirements.
- It forces all communications to still take place using the RTI and the current FOM specifications.
- It allows for standardized filters, such as the DDM and predictive filters, to be implemented once and used by multiple subscribers as standard utilities.

- It allows Requestable Objects to be implemented as proposed, but with an open ended ability to change the filtering criteria.

4.1 Logic Definition and Subscription

The first step in registering arbitrary logic is associating the logic with a specific object, attribute, or interaction. This can be accomplished by extending the following RTI services:

- subscribeObjectClassAttribute
- subscribeInteractionClass

The above routines can either be extended, or complimented with similarly named routines, allowing for an additional parameter which identifies the logic script or task to be performed in the remote process. The RTI must then distribute the logic to each of the federates publishing the specified object, attribute, or interaction. When a publishing federate publishes a new object or attribute value, the specified logic must be called in the local component of the RTI residing in the publishing federate's process. If the filtering logic component determines that the update or published object should be forwarded to the subscribers process it should notify the RTI. The RTI should then deliver the message as usual (Note that this also requires a protocol extension to the RTI). If the filtering logic filters the update or object and decides it is not useful, the RTI is not notified, thus eliminating the RTI overhead, the network communications, and the local subscriber processing.

Similar extensions are required to detach the specified logic from objects, attributes, and interactions. This can be accomplished by reattaching a null filter.

The potential benefits of this remote filtering are significant. One example illustrating these benefits is when a federate is being used to collect data for analysis or After Action Review (AAR) purposes. Suppose, the personnel using the data need their federate to collect and process detailed attribute or interaction data about helicopters only if a combination of four independent battlefield parameters exist simultaneously. If the specific combination of attributes do appear, subscription of additional objects, classes, and attributes will take place. By placing a local component in the publishing federate that keeps a history of the observed attributes, all communication to the subscribing federate can be eliminated until the combination of values appears.

4.2 Flow of Control Examples

The following illustrations show the flow of control diagrams in the publishing federate's process for publication, update, and requests for republish. It should be noted that any of the arrows entering or leaving the subscriber federate remote logic circles must be accompanied by extensions to the RTI.

A new object being published by a federate is shown in Figure 1. The new object's existence is first identified to three subscribed federates filtering logic which have been previously transmitted to the publishing federate, and resides in the publishing federate's local RTI component process. Each subscriber's filtering logic has the ability to notify the RTI to actually forward the object to its subscriber process. In the example, Subscriber S2's filtering logic rejected the object, and does not need future updates to the object.

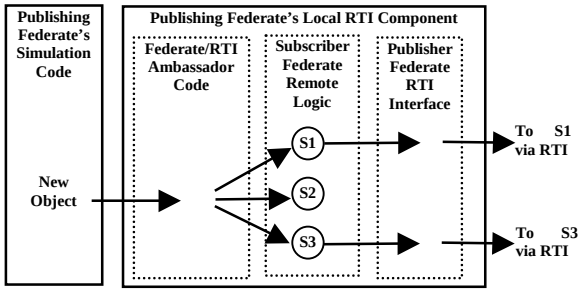

Figure 1: Publishing of New Objects

An update of an attribute to the object just published is shown in Figure 2. Since S2 did not subscribe to the object, it is not notified of the update. S1 was notified, but based on its filtering logic, rejected the update. S3's filtering logic passes, and notifies the RTI to continue to forward the update to S3's process.

Figure 3 shows an incoming request from federate S1 to republish the attributes for the new object. The publishing federate is notified and responds to the republish request. Since S2 and S3 are not involved with the request, they are ignored. The updates are then passed to S1's filtering logic in the publishing federate's local RTI component, which then forwards all or some of the requested attribute values.

The Subscriber S1 sending logic instructions to its local filter in the publishing federate is shown in Figure 4. These instructions tell the publishing federate's local RTI component to request from the RTI a republish request of the object's attributes for S1. As in Figure 3, the publisher federate responds, and updates are only sent to S1's process. The two arrows coming out of S1's process into the Federate/RTI Ambassador Code indicate that the logic can be executed numerous times for several different objects, even though the invocation is only sent once. The difference between Figures 3 and 4 is that the initiation of the request is done local to the publisher's federate in Figure 4, not from S1's process as in Figure 3. For extensions such as those proposed by the Requestable Object Service, arbitrary logic and instructions can be sent from the subscriber's process, both limiting network

bandwidth, while accomplishing the ability to perform arbitrary logic filtering.

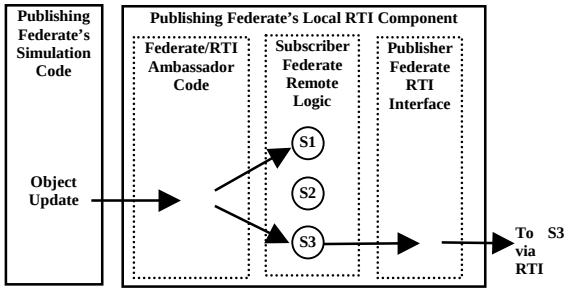

Figure 2: Update an Attribute Value

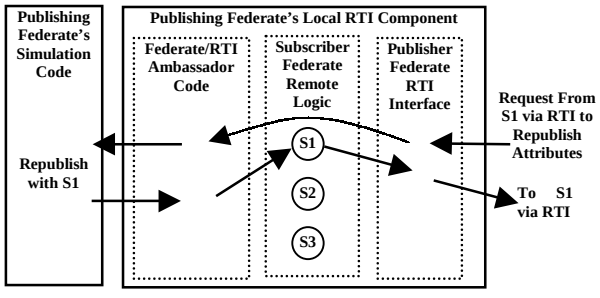

Figure 3: Request for Republication

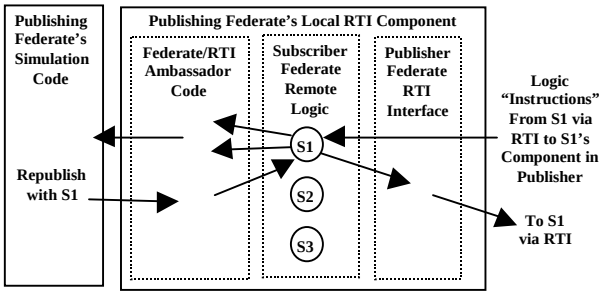

Figure 4: Subscriber Logic Communication

There are many variations possible from the above examples. The most important aspects of the above criteria is that all communications occur through the RTI and all queries follow the HLA rules

5 DIRECTED COMMUNICATIONS

In the prior examples, each subscriber's logic is making an individual determination whether the communications to the subscriber's process is to take place. Note that the current HLA/RTI specification contains no prevision for identifying specific federates for communication. Extensions to the protocol can be made which uniquely identify each federate as a handle, independent of process. When forwarding object discoveries, interactions, and attribute updates, this federate handle must be added to the protocol to assist the RTI in limiting the communications.

In a similar manner, extensions to the HLA/RTI protocols should allow buffer communication through the RTI between the subscribing federate and its remote filtering logic residing n the publishing federate's local RTI component.

6 PERFORMANCE AND CONTROL ISSUES

The proposed methodology takes the systems approach to simulation design by providing extensions to the RTI for the purposes of control and optimization. The distribution of logic from one process to another will affect the number of processing cycles required for each individual federate. While this may be perceived as a disadvantage for any individual federate, it is a tremendous advantage for balancing overall network and processor load, especially if good statistical tools are implemented by the RTI for tracking processor utilization and network communications.

The administrators of the RTI Executive, and in particular, the simulation designers must have the control mechanisms to override each of the subscribing federates if necessary. The trade-offs between local machine performance, federate process overhead due to distributed logic, and network bottlenecks should be under external control from any individual federate. By extending override controls in the MOM classes and the RTI Executive it should be possible to disable abusive logic distribution.

Aside from the CPU utilization, care must be taken in allowing distributed logic to create local objects in the remote process for the purposes of tracking history. State of the art object technologies allow for the tracking of object storage and should be adequate for monitoring and automated garbage cleanup of unused objects.

7 CONCLUSIONS

The potential benefits of this remote filtering are significant, not only in pure reduction of messages required to be communicated by the RTI, but in the unlimited potential for customization and extensions. The most significant advantage to the entire approach is the isolation of the filtering concepts from the publishing federate. Negotiations for FOM requirements and extensions are no longer required for the purposes of optimization. All existing DDM proposals, requestable objects, and predictive filtering techniques can be implemented using a common protocol.

The proposed approach is in keeping with HLA systems engineers' conclusion that future modeling and simulation uses and requirements could not be anticipated and that future technological innovations could not be

predicted. It adheres to the rules and spirit of the HLA architecture without causing major disturbances to the existing RTI implementation.

REFERENCES

Briggs, Keith, et al. 1997. Extending the HLA to Support a Requestable Object Service. Paper 97F-SIW-040 in *Proceedings of the 1997 Fall Simulation Interoperability Workshop (SIW).* Simulation Interoperability Standards Organization, Orlando, FL.

US Department of Defense. 1997. High Level Architecture Interface Specification, Version 1.1. Defense Modeling and Simulation Office.

Zeigler, B.P. and J.S. Lee. 1998. Theory of Quantized Systems: Formal Basis for DEVS/HLA Distributed Simulation Environment. In *Enabling Technology for Simulation Science(II), SPIE AeoroSense 98*. Orlando, Florida.

AUTHOR BIOGRAPHIES

WILLIAM S. MURPHY Jr. is a Major in the U.S. Army Corps of Engineers with 16 years of commissioned service. He graduated with a B.S. from the United States Military Academy in 1982 and with a M.S. in Mathematics and Computer Sciences from the Colorado School of Mines in 1992. Major Murphy is a licensed Professional Engineer (PE) in the state of Virginia. He is currently an operations research analyst at the U.S. Army Training and Doctrine Command (TRADOC) Analysis Center (TRAC) in Monterey, California. He is also the Chair of the Military Operations Research Society's (MORS) Computing Advances in Military Operations Research working group.

GALEN D. ASWEGAN is currently President of Tapestry Solutions in San Diego, CA. He graduated with Honors from Iowa State University with B.S. degrees in Computer Science, Mathematics, and Physics. He has over 20 years of experience in developing large scale, distributed, graphical applications in both commercial and DoD sectors. He has developed the REVUE technology and is directly responsible for architecture and implementation of the Vision XXI and Tapestry Kernel distributed object architectures.