Advanced Message Routing for Scalable Distributed Simulations

Thomas D. Gottschalk
Center of Advanced Computing Research
California Institute of Technology
Pasadena, CA 91125
tdg@cacr.caltech.edu

Philip Amburn
SAIC, PET FMS On-Site
Wright-Patterson AFB OH 45433
philip.amburn@wpafb.af.mil

Dan M. Davis
Information Sciences Institute, USC
Marina del Rey, CA 90292
ddavis@isi.edu

On large Linux clusters, scalability is the ability of the program to utilize additional processors in a way that provides a near-linear increase in computational capacity for each node employed. Without scalability, the cluster may cease to be useful after adding a very small number of nodes. The Joint Forces Command (JFCOM) Experimentation Directorate (J9) has recently been engaged in Joint Urban Operations (JUO) experiments and counter mortar analyses. Both required scalable codes to simulate over 1 million SAF clutter entities, using hundreds of CPUs. The JSAF application suite, utilizing the redesigned RTI-s communications system, provides the ability to run distributed simulations with sites located across the United States, from Norfolk, Virginia, to Maui, Hawaii. Interest-aware routers are essential for scalable communications in the large, distributed environments, and the RTI-s framework, currently in use by JFCOM, provides such routers connected in a basic tree topology. This approach is successful for small to medium sized simulations, but faces a number of constraining limitations precluding very large simulations.

To resolve these issues, the work described herein utilizes a new software router infrastructure to accommodate more sophisticated, general topologies, including both the existing tree framework and a new generalization of the fully connected mesh topologies. The latter were first used in the SF Express ModSAF simulations of 100,000 fully interacting vehicles. The new software router objects incorporate an augmented set of the scalable features of the SF Express design, while optionally using low-level RTI-s objects to perform actual site-to-site communications. The limitations of the original MeshRouter formalism have been eliminated, allowing fully dynamic operations. The mesh topology capabilities allow aggregate bandwidth and site-to-site latencies to match actual network performance. The heavy resource load at the root node now can be distributed across routers at the participating sites. Most significantly, realizable point-to-point bandwidths remain stable as the underlying problem size increases, sustaining scalability claims.

Keywords: Linux, cluster, scalability, JSAF, routers, communications

1. Introduction

The modeling and simulation community has just begun to utilize the power afforded by large-scale Linux clusters and other high-end computing [1]. One of the hindrances retarding rapid acceptance of these compute assets is the lack of scalability in many of the codes extant in the 1990s [2]. This paper describes the approach and the results obtained by the application of a novel inter-node communications architecture that has proven robust and has delivered near-linear scalability across configurations of up to 1,900 heterogeneous nodes [3].

Experience has shown the need for scalability for more than two decades in forces modeling and simulation, as well as in the modeling and simulation communities that focus on physics, weather, climate, materials science, computational fluid dynamic, structural mechanics, computational chemistry and biology, as well as a host of other simulations fields [4].

Forces modeling and simulation (FMS) is increasingly important in analysis, evaluation, and training. FMS has been enabled and facilitated by advances in computer science. Increasingly, these programs are being tasked with larger, more complicated and higher-resolution missions. Just the necessary inclusion of civilian personnel and the ability to assess their impact on and vulnerability to combat conditions has increased the number of entities required by a couple of orders of magnitude. Interesting work has been done on scalability across smaller numbers of processors [5], but the authors were tasked with solving problems requiring the use of a much larger number of CPUs. Scalability at these levels is achieved by interest management of various designs, insuring transmissions to only relevant nodes and minimizing "all-to-all" communications. This paper examines an approach to optimizing interest-managed communications in a way that can be generalized.

1.1 Large-Scale Forces Modeling and Simulation

Recent experiments within the Joint Forces Command (JFCOM) Experimentation Directorate (J9) demonstrate the feasibility and utility of forces modeling and simulation applications in a large field of play with fine-grained resolution. Simulating such battle spaces requires large computational resources, often distributed across multiple sites. The ongoing Joint Urban Operations (JUO) experiment uses the JSAF application suite and the Run Time Infrastructure (RTI-s) to scale to over 300 simulation federates distributed across the continental United States and Hawaii [6]. The JUO exercise has shown the scalability of the JSAF/RTI-s infrastructure and of interest-based, router-managed communication. At the same time, the simulation has highlighted a need for improvements in the communication architecture.

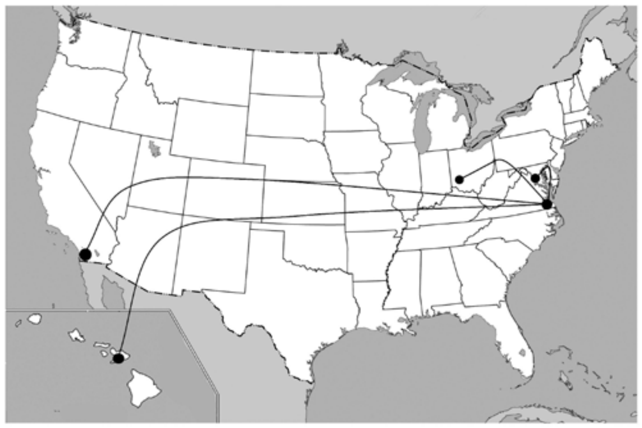

Figure 1. Software tree; WAN routing topology for the JUO exercise

The current JUO network topology is a tree of software routers (see Figure 1 for wide-area-network diagram). The hub and spoke network model introduced by this tree infrastructure increases latency between distributed sites and exposes the entire network to a single point of failure. The tree topology also poses a scalability limitation within the distributed sites. It is our belief that an improved routing infrastructure is required for the continued success of large-scale entity level simulations, particularly as either entity counts or complexity/fidelity increase.

This paper presents an improved routing architecture for large-scale HLA environments, using fully connected meshes as the basic topology. These MeshRouters provide a scalable solution for interest-managed communication, as well as a more accurate mapping of software routing to available network topologies. A desirable side-product of this design is that it is a naturally fault-tolerant architecture.

1.2 Scalable Parallel Processors

The JUO exercise requires a computational ability unavailable using traditional groups of workstations. Scalable Parallel Processors (SPPs) provide the required computational power, with a modest increase in development and execution effort [7]. An SPP is a large collection of processing elements (nodes) connected by a fast communication network. Common SPPs include the IBM SP, SGI Origin, Cray X1, and Linux clusters. Traditionally, SPPs provide services not available in a group of workstations: high-speed networks, massive disk arrays shared across the entire resource, and large per-CPU physical memory. In addition, SPPs generally have uniform environments across the entire machine and tools for scalable interactive control (starting processes across 100 nodes takes the same amount of time as it does across 10).

Linux clusters recently have become a suitable platform for the high performance computing community and are therefore readily available at Department of Defense High Performance Computing Modernization Offices' Distributed and Major Shared Resource Centers. These clusters are ideal platforms for use in the JUO exercise because of their close heritage to the Linux workstations typically used in the interactive test bays. Although there is additional software required to tie the cluster into one SPP, the basic libraries, compiler, and kernel often are the same on a cluster as on a workstation.

1.3 RTI-s

RTI-s provides the HLA Run Time Infrastructure for the JUO federation. RTI-s originally was developed after the STOW exercises to overcome the scalability and performance limitations found in RTI implementations at the time. It should be noted that RTI-s is not a fully compliant HLA/RTI implementation {8,9}. Specifically, it does not implement timestamp ordered receives, ownership transfer, and MOM interactions. In addition, federates discover new objects at first update rather than at creation time. The JSAF applications are receive-ordered by design and are optimized to respond best to delayed object discovery, so these limitations are not constraining in the existing environment.

RTI-s utilizes a flexible data path framework, an example of which is shown in Figure 2, allowing the use with a number of communication infrastructures.

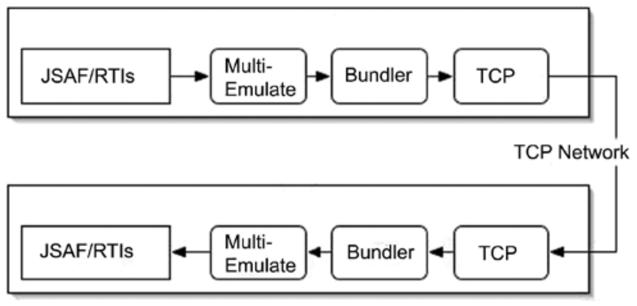

Figure 2. RTI-s data path architecture for TCP communication

Currently there is support for multicast User Datagram Protocol (UDP), point-to-point UDP, point-to-point Transmission Control Protocol (TCP), and Message Passing Interface (MPI) — using a send/receive architecture. Bundling and fragmenting of messages is provided by components that can be reused for TCP and UDP communication. Kerberos authentication for data packets has been implemented for TCP communication.

Point-to-point modes in RTI-s uses separate routing processes for communication. The routers provide data distribution and interest management for the federation, which would be too heavy for a simulator to handle. Presently, a tree topology (Figure 3) is used for connecting routers. A tree presents a simple structure for preventing message loops, as there are no potential loops in the system.

1.4 Synthetic Forces Express

The Synthetic Forces Express (SF Express) [10,11] project first demonstrated the suitability of both the SPP and MeshRouter concepts for discrete entity modeling. The SF Express project extended the ModSAF simulation engine [12], focusing on the communication protocols to extend scalability. It was driven by the announced needs of DARPA to support up to 50,000 vehicles on a battlefield.

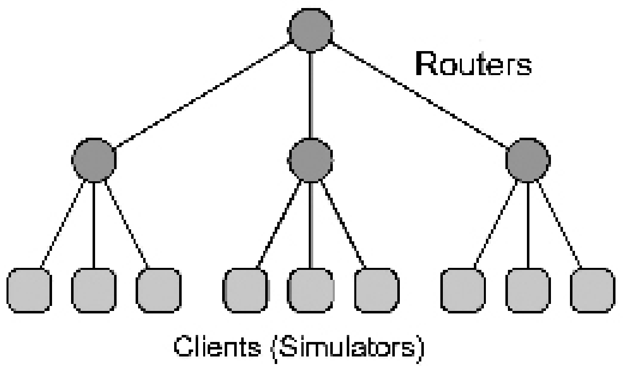

Figure 3. Tree topology used by RTI-s for point-to-point message traffic

In December 1996, the SF Express team first achieved a 10,000-vehicle simulation using a single 1,024-node Intel Paragon machine. Message routing within the SPP used MPI [13]. Later work allowed the code to run on multiple SPP installations across a variety of networks by introducing gateways between SPPs. Implementing a conceptual architecture that was similar to the internal inter-node architecture, these I/O nodes greatly reduced redundant or superfluous traffic on the net. Most of the runs were done without exceeding T-1 capacities. The gateway routers were connected using UDP. With these improvements, the project achieved a simulation of 50,000 vehicles using 1,904 processors over six SPPs. These were housed in 13 different machines, in nine locales and resident in states from Ohio to Texas, and from Maryland to Hawaii.

The structure of the SF Express router network is shown in Figure 4. The basic building block for this architecture is the triad shown on the left, with a Primary router servicing some numbers of client simulators. Two additional routers (known as the PopUp and PullDown routers) complete the basic triad. These routers distribute (PopUp) and collect (PullDown) messages from client simulators outside the Primary's client set. The SF Express architecture scales to increased problem size by replicating the basic triad and adding full up⇔down communication links among the triads, as shown in the right-hand side of Figure 4 below.

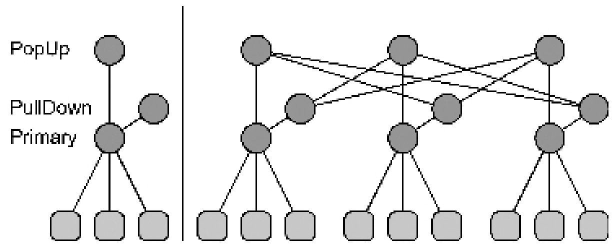

Figure 4. Basic building block of the SF Express routing network (left) and an example mesh topology (right)

While the SF Express project was quite successful, it had no life beyond a number of 50,000–100,000 entity simulation demonstrations. This was expected for a number of reasons. For example, the algorithms and software developed for that project were not compatible with ongoing SAF developments (e.g., the move to RTI). Finally, the MPI-based communications used within the SPPs did not tolerate the restarts and process failures found during a long-running exercise. The MeshRouter framework described in this work provides a more general architecture for scalable distributed simulations than the original SF Express program.

2. Designing for Scalability

As previously mentioned, the JSAF/RTI-s application suite currently scales to over 300 federates and over 1 million entities (including simple clutter). However, current routing topologies limit the scalability of the overall system. In order for an interest-based communication infrastructure to scale, three conditions must hold over an arbitrary interval of simulation time:

- A given client must generate a bounded number of messages.
- A given client must receive a bounded number of messages.
- Given that, the communication through any given router also must be bounded.

An interest management system and careful federate design achieve bounded client communication. Bounded router communication is a function of network design and can be achieved using a mesh topology. The focus of this work is significantly enhancing scalability using router communications that are effective for arbitrary schemes of interest enumeration and management.

2.1 Interest Management Using RTI-s

The aggregate amount of data produced by the JUO federation is greater than any one federate is capable of processing. An interest management system is used to limit the amount of data a federate must process [14]. The federate declares which information it is interested in (e.g., red force tanks in position cell X) and the RTI is responsible for ensuring only this subscribed information is received by the federate.

When used in a multicast environment, RTI-s utilizes the concept of multicast channels for filtering, with interest states having associated channels. The message is multicast to the federation's network and filtered on the receiving side. The receiver filters the message at the kernel level, so that the application never sees messages for states in which it is not interested. Overhead when no interest states are set is relatively small, but non-zero. Due to the limited number of available multicast channels, the number of interest states is limited, which increases the amount of traffic associated with each interest state.

When running in point-to-point mode (using either TCP or UDP), interest management is send-side squelched. Software routers maintain interest state vectors for each connection and only send messages to clients that have expressed interest in a message type. The overhead for a federate to exist in the federation without any expressed interest is almost zero. Because interest states are not tied to hardware and operating system limitations, the number of available interest states is bounded only by how much memory can be allocated to interest vectors. This is an enormous improvement over multicast IP. It also was one of the innovations of SF Express.

An interest management system provides only the infrastructure for bounding the data flowing out of and into a particular simulator. The simulator must show care in declared interest states to prevent subscribing to more data than it is capable of processing. For the purposes of analyzing the scalability of routing infrastructures, we assume that the simulator limits interest declarations to guarantee bounded communication. In both the earlier SF Express and current JUO experiments, this assumption appears valid. Again, the extensibility of this approach to other schemes of interest management should be straightforward using this router architecture.

2.2 Routing Scalability

The scalability of the basic MeshRouter network is easily argued as follows. It is first necessary to assume that the underlying simulation problem itself has a scalable solution. This posits a bounded message rate on the Primary$\Rightarrow$PopUp and PullDown$\Rightarrow$Primary links within a basic triad, and bounded Up$\Rightarrow$Down message rates within the interconnection links of the full network. The impediments to complete scalability of the mesh architecture have to do with interest declarations among the upper router layers. Each PullDown must announce its interest to every PopUp. In principle, these interest broadcasts could be made scalable through an additional network of communication nodes (at the associated cost of increased latencies for interest updates). In practice, however, these interest updates were not frequent enough to cause any difficulties in SF Express simulations with nearly 100 triads, each supporting around 20 simulation nodes, in the full mesh. An experiment with a similar setup using the current infrastructure shows similar results. This formally non-scaling component is, in fact, a sufficiently tiny component of the overall communications load that implementation of the "formal" scalability cure is not warranted for present or near-term simulation scenarios.

3. Routing Flexibility

The scalability issues with the TreeRouter topology of RTI-s have been discussed previously. The tree topologies also map poorly onto physical wide-area networks. Figure 1 shows the route taken for any message crossing multiple sites in the JUO exercise. The path taken for a message to go from Maui to San Diego is sub-optimal; the data must first travel to Norfolk, then back to the west coast. This extra transmission time increases the latency of the system, which lowers overall performance. Since wide-area links often have less bandwidth available than local area networks, such routing also places a burden on the Virginia network infrastructure, which must have bandwidth available for both the incoming and outgoing message in our Maui to San Diego example.

The mesh routing infrastructure provides a better utilization of physical networks by sending directly from one source to destination router. The network infrastructure is free to route messages in the most efficient way available. Figure 5 shows one possible routing topology for the JUO exercises, using MeshRouters to minimize the distance messages must travel.

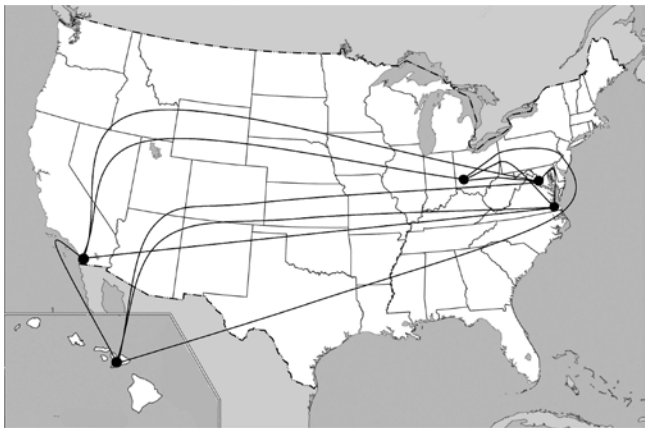

Figure 5. Advanced routing topology for JUO runs

The MeshRouters developed for RTI-s adopted many of the design decisions made in the SF Express project. The router triad concept is perhaps the most obvious of the design decisions adopted from SF Express, providing an elegant method of avoiding "message looping" in the mesh, while allowing an arbitrary number of routing decisions to be made when transferring messages. However, significant design changes have produced a radically more advanced and flexible infrastructure that is discussed below. The details focus on the overall MeshRouter framework's applicability and utility for large simulations requiring hundreds of CPUs, more than the scalability of any particular simulation. The underlying scalability of various simulations may vary considerably, with most of them producing "one-to-one" or "one-to-many" communications, but some being conceptually forced to use "one-to-all" communications. The presented architecture enhances the former case.

3.1 Flow Control

Flow control is a technique that was necessary to ensure adequate communications performance when using Message Passing Interface (MPI) instructions to establish communications. While not directly leading to scalability, it is described briefly here to show the general applicability of standard methods within this architecture. A tight flow control with Request to Send/ Clear to Send (RTS/CTS) behavior was used in the SF Express design. SF Express used the MeshRouters only within a single SPP, where latency was extremely low and available bandwidth greatly exceeded expected message transfer rates. The overhead of sending the RTS and CTS messages would not negatively impact the performance or scalability of the system. The communication medium of choice (MPI) requires pre-posted receive buffers of a known size, requiring an RTS/CTS protocol for sending large messages. However, recent trends have shown CPU power improvements far outpacing network latency and bandwidth improvements. On modern networks, an RTS/CTS protocol poses a significant performance burden. Therefore, the MeshRouter architecture now utilizes an eager send protocol with messages dropped by priority when queues overflow.

3.1.1 Application-Independent Message and Interest Objects

The MeshRouter software is object-oriented (C++), with a limited number of standard interfaces to user message and interest base classes. For present purposes, the implications of this factorization are:

- The MeshRouter system is designed to be compatible with ongoing changes and evolution within the RTI-s system, requiring little more than a re-compile and re-link.
- The MeshRouter system can support applications other than SAF/RTI, given appropriate different instances of the message and interest objects.

3.2 Simplified, General-Purpose Router Objects

The many distinct router varieties (Primary, PopUp, PullDown, Gateway) of the SF Express router network have been replaced by a single router object, as indicated by the schematic in Figure 6. Routers simply manage interest-limited message exchange among a collection of associated clients. The distinctions that had been hardwired into the various router types of SF Express are now summarized by sets of flags associated with the clients. The flags (simple Boolean variables) specify whether the:

- client is a source of data messages
- client is a sink of data messages
- client is persistent (non-persistent clients are destroyed if the communications link fails)
- client is upper or lower (this simple hierarchy provides the mechanism to prevent message cycles)

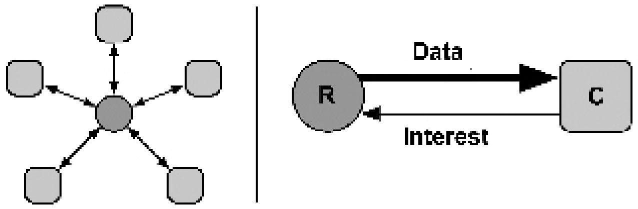

Figure 6. Schematic of a router process (left) and the interest declaration/data flow of a typical router/client connection (right)

These four flags are sufficient to reproduce the specific communications model of Figure 4 and a number of other networks, such as the TreeRouter model available in the JSAF/RTI-s library.

3.3 Factorized Communications Primitives

The MeshRouter object design relies on a very careful isolation/factorization of the underlying message exchange protocol from the rest of the software. The essential object design is indicated in Figure 7 and has three layers.

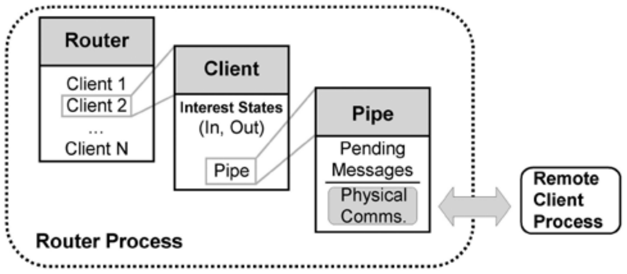

Figure 7. Schematic design of the MeshRouter

Router objects: These are little more that smart lists of objects associated with the clients in Figure 7. In normal operations, routers simply execute the fundamental message and interest manipulation methods for the associated clients. Routers also are responsible for management of the overall client list, including:

- removal of clients that have stopped communicating;
- initiation of communications links, as needed, to specified (persistent) clients; and
- client additions, in response to requests from external processes.

Client objects: Managers of the interest declarations and pending message queues on external client process.

Pipe objects: The interface between the Message/MessageList formalism of the MeshRouter software and the real world bits on the wire communications to the actual external processes. The pipe object base class provides the last essential factorization of application specific details from the overall, general MeshRouter framework.

The communication factorization within the pipe class is essential to the general applicability and ease of use of the MeshRouter system. A number of specific pipe classes have been implemented to date, with the most important being:

- RTIsPipe: Message exchange using the RTI-s framework. (This object has been built entirely from objects and methods in the RTI-s library)
- MemoryPipe: Message "exchange" within a single process on a single CPU. This is used when two or more router processes in the sense of Figure 6 and Figure 7 are instanced as distinct objects within a single management process on a single CPU

The factorization of application-specific mechanisms is, in fact, slightly more complicated than that. The pipe object has sufficient virtual interfaces for data exchange between a router and a general client. An additional virtual object/interface (the Connection-Manager) is needed to support dynamic addition and deletion of clients during router operations.

3.4 Router Configurations/Specifics

The numerical experiments described in this work explore two different overall communications topologies built from basic MeshRouter objects: the tree and mesh topologies shown in Figure 8.

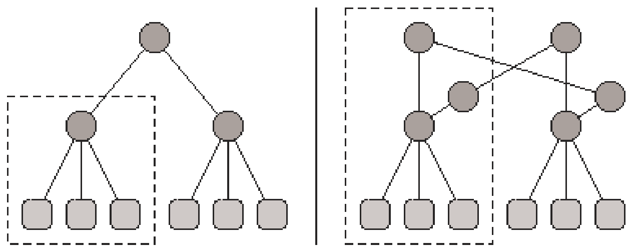

Figure 8. Tree and mesh topologies used for studies

In the tree topology, there is an entire CPU allocated to each router. All connections (simulator to router or router to router) use the full RTIsPipe instance. The persistent router clients in the sense of Section 2 are the upper router clients (if any) for each component router. All other communications links are generated dynamically. The tree topology used in the work discussed below is functionally identical to that used with JUO/RTI-s.

For the mesh topology simulations, all three routers within the basic triad of Figure 4 are instanced as distinct objects on a single CPU, while MemoryPipe connections are used for the Primary⇔PopUp and Primary⇔PullDown links within a single triad. All other links in Figure 8 use the RTIsPipe, with the cross-triad PullDown⇒Primary links persistent.

As noted, the current RTIsPipe implementation is based entirely on objects and method calls within the current RTI-s library. This is important for demonstrating ease of insertion of the MeshRouter formalism into the RTI-s libraries, but it does result in a few minor inefficiencies. These include one extra memory copy per message and duplicate interpretations of incoming interest declaration messages. These inefficiencies can be removed in future, more finely tuned pipe instances. Indeed, the careful communications factorization within the MeshRouter package supports mixed pipe instances tailored to communications specifics for any of the individual links in Figure 8. In particular, the optimal pipe instances for WAN and LAN links may be quite different. Though supported by the overall design, these refinements are beyond the scope here.

4. Results

The Koa cluster at the Maui High Performance Computing Center was utilized for the first round of testing of the MeshRouters. Koa is a 128-node Linux cluster with two 3.06 GHz Intel Xeon processors and 4 gigabytes of memory per node. Nodes are inter-connected via gigabit ethernet. All routing topologies were generated using the standards for the JUO experiment: 5 federates per router and 4 routers per router (the second only applies to TreeRouters). The default configuration parameters were used for both RTI-s and the MeshRouter. Since the MeshRouter utilizes the RTI-s communication infrastructure, we believe that any parameter tuning done to one system would apply equally well to the other system. To highlight the importance of topology in routing infrastructure, we show the MeshRouters running in a tree configuration as well as the standard RTI-s tree.

A number of tests ensured the MeshRouters performed as required in JSAF experiments. They were of a size where the TreeRouters architecture still scaled well. The mesh infrastructure was used for an extended simulation using the JSAF suite. As expected for a small-scale simulation, the MeshRouter and RTI-s TreeRouter looked comparable to the JSAF operator.

Latency measurements were taken on the Koa cluster. The MeshRouter performed slightly better in mesh configuration than in either tree configuration, but were within the measured error. Koa's low latency network combined with a short tree (only 3 levels deep) account for this measurement.

The authors developed a test federate that used pair-wise communications. This enabled them to assess the communications typically found on an RTI-s platforms where interest management was reasonably effective (i.e. one-to-all communications were minimized). They assert that this would similarly be applicable to other simulations capable of reasonable interest management. This use of a tool allows easy control of the communications load and testing at the extreme end of that load. Anecdotal evidence from the JSAF experience substantiates the validity of this approach. This simplified case is suitable for demonstrations and the scalability is similar to that seen in the SF Express project.

4.1 System Throughput

For testing the maximum throughput of the routing infrastructures, pair-wise communication was used. Attribute updates were sent between process pairs as fast as possible, with loose synchronization to ensure multiple pairs were always communicating. The average per-pair throughput, specified in number of `reflectAttributeValues()` calls per second for a given message size, is shown in Figure 9. For the test, 50 pairs were utilized, with 28 TreeRouters or 20 MeshRouters creating the router infrastructure.

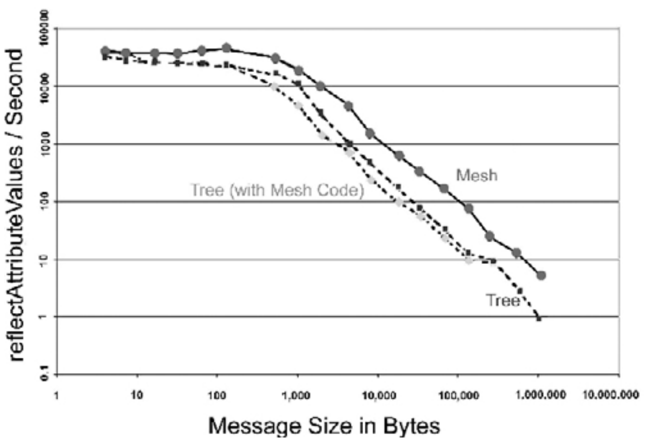

Figure 9. Log/log graph of bandwidth of mesh (solid line) and tree (dashed lines) routers

As expected, Figure 9 shows that the maximum number of updates per second goes down as message size increases. The MeshRouter in a mesh configuration is able to move more traffic, and thereby cause more (~ one half order of magnitude) updates than either the RTI-s tree infrastructure or the MeshRouters mapped into a tree topology. The RTI-s and MeshRouter tree configurations both would slow down at the root node of the tree, causing both lower realized aggregate bandwidth and an increase in dropped messages as message queues increased in length.

In a tree configuration, the RTI-s TreeRouter module performed better than the MeshRouter. This is not unexpected, as RTI-s has been finely tuned to reduce memory copying and contention. The MeshRouter lower level has only started to be tuned for optimal performance on a Linux system. We see no detail of implementation that would prevent the MeshRouter from matching the performance of the TreeRouters and believe that further tuning will increase the performance of the MeshRouter in any configuration.

4.2 Large-Scale Performance Testing

After the early indications of the success of the scalability reported above, a new set of performance runs were accomplished in the first quarter of calendar 2005. These runs were conducted at the ASC-MSRC and were supported by the staff there as well as the HPCMP PET organization. The cluster there (Glenn) is the same configuration as Koa at MHPCC; but in this case, there are 60GB hard disks on each local node (a configuration subsequently installed on Koa).

The underlying simulation for the Glenn performance studies, using the test program "rtiperf," again involved timed message exchanges between pairs of simple federates, as indicated in the schematic of Figure 10.

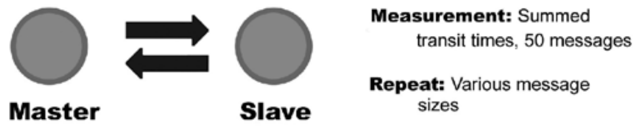
Master
Slave
Measurement: Summed transit times, 50 messages
Repeat: Various message sizes

Figure 10. Schematic for a pair-wise message exchange

One processor within each pair initiates a sequence of 50 fixed-size message exchanges with its partner, adding the times for each there-and-back message exchange. The process is repeated for a number of different message sizes. The primary output for each master-slave pair is simply a list of average exchange times versus message size.

The results presented in this section involve 96 total rtiperf federates (48 master-slave pairs) that are communicating through the router networks. The TreeRouter network is shown in Figure 11.

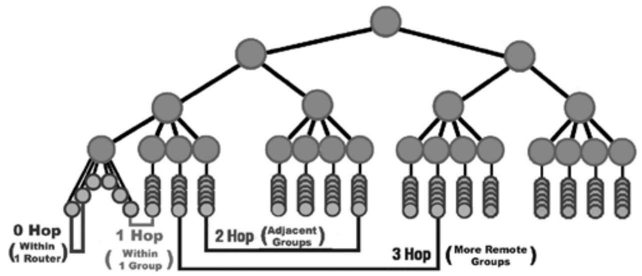

Figure 11. Connectivity for TreeRouter tests

The rtiperf applications are associated with specific lowest-level routers in groups of six. Three layers of higher-level routers provide connectivity throughout the system.

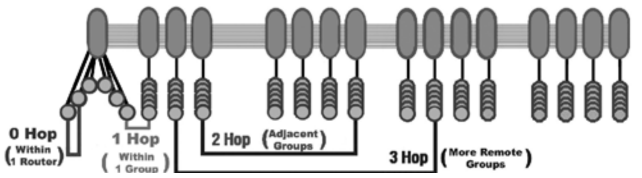

Figure 12. Connectivity for MeshRouter tests

The corresponding connectivity/network for the MeshRouter is shown in Figure 12. The ovals in this figure represent a full Primary-PopUp-PullDown router triad in the sense of the dashed box in the right hand side of Figure 8. The entire triad is instanced on a single CPU, with software pipes (MemoryPipes) connecting Primary⇔Popup and Primary⇔PullDown within an individual triad. The shaded band in Figure 12 represents the standard, full mesh connectivity among triads, as discussed above.

The timing results presented next are based on three sets of runs for different separations of the Master-Slave rtiperf federates of Figure 10. The three configurations are labeled 0-Hop, 2-Hop, and 3-Hop (1-Hop configurations were not measured), according to the distance between the groups of simulator nodes, hence the depth of the tree-router communications path (ignoring the lowest-lying leaf routers). Representative Master-Slave pairings for the three configurations are shown in Figures 11 and 12.

The n-Hop label was retained to indicate the relative dispersion of the compute notes even in the MeshRouters where hops are not relevant. This is convenient, but perhaps a bit misleading. Table 1 lists the number of distinct inter-processor (network) communications lengths for the actual message paths in the router networks of Figures 11 and 12.

Network	0-Hop	1-Hop	2-Hop	3-Hop
TreeRouter	2	4	6	8
MeshRouter	2	3	3	3

Table 1. Number of network links for n-Hop message exchanges in the networks of Figures 11 and 12

Note that the TreeRouter generally involves much longer communications paths than the MeshRouter. Before proceeding to measurement results, two remarks are in order:

1. The pipe objects (in the sense of Section 3.3) for Figure 12 are RtisPipes, built entirely from standard RTI-s library objects.
2. The standard message bundling mechanisms within RTI-s have been disabled for the comparison studies.

The first point is important. It means that the on-the-wire communications for Figures 11 and 12 are essentially identical, and the performance differences noted below are dominated by architectural differences.

Message Size Versus Times, Tree

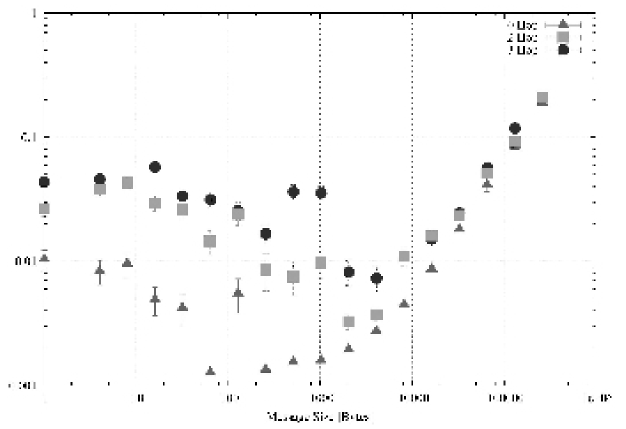

Figure 13. Mean task times versus message size (log/log) for the TreeRouter configuration

The basic data for the performance timing runs of TreeRouter configurations are shown in Figure 13, and the corresponding timing results for the MeshRouter configuration are shown in Figure 14. The individual data points and error bars in these plots are evaluated for each message size/hop count as follows:

1. The mean task times for the 48 contributing rtiperf pairs Figure 10 are sorted.
2. To minimize outlier-induced fluctuation, the two largest time values of each set are discarded.
3. The data/error values included in the plots are the statistical means and standard deviations derived from the retained 46-value samples.

Message Time Versus Size, Mesh

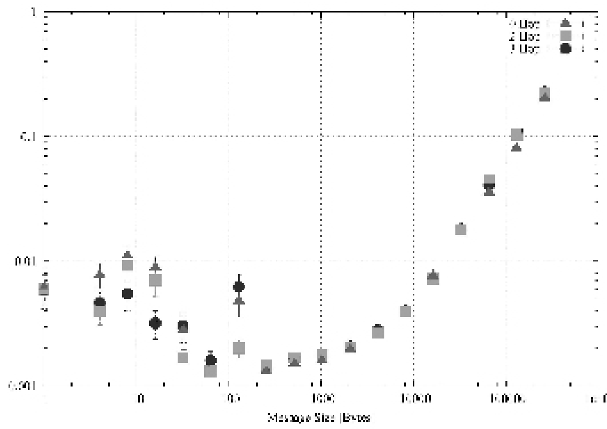

Figure 14. Mean task times versus message size (log/log) for the MeshRouter configuration

There are several general features to be noted from the results in Figures 13 and 14:

1. The six distinct curves approach a common curve for very large messages. This is reasonable, as large message rates are bandwidth limited.
2. The 0-Hop, 2-Hop, and 3-Hop results for the MeshRouter are remarkably similar. Similarity of the 2-Hop and 3-Hop results is reasonable given the identical number of associated network messages from Table 1.
3. Except at the bandwidth-limited, large message tails of the curves, the TreeRouter results show significant performance degradations (i.e., larger mean message times) for 2- and 3-Hops.

Throughput Enhancement: Mesh Versus Tree

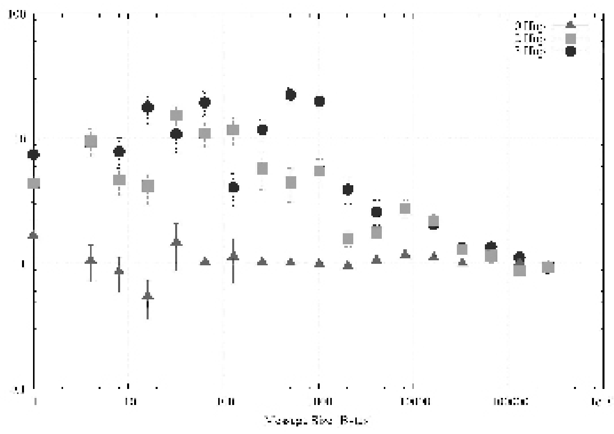

Figure 15. Mesh versus tree performance ratio (log/log) during 0-, 2-, and 3-Hop configurations

A more direct comparison of the Mesh⇔Tree performance is shown in Figure 15, where the ratios,

R = [Mean Tree Time]/[Mean Mesh Time],

are plotted versus message size.

Note that the 0-Hop ratio is essentially one throughout the entire range of message sizes. This is reasonable, as 0-Hop messages never move beyond the lowest or Primary routers in Figures 11 and 12. The fact that this (empirical) ratio is consistent with unity is evidence that the MeshRouter formalism introduces no additional significant inefficiencies due to the high-level objects of Section 3. The 0-Hop times are, for both MeshRouter and TreeRouter, dominated by performance of the standard RTI-s TCP/IP connection.

However, for message sizes below the bandwidth-dominated, large-message end, the performance of the MeshRouters is significantly better than that of the TreeRouter. This is, in fact, an expected result, given the larger number of distinct physical communications for the TreeRouter, as noted in Table 1.

Throughput Enhancement: Mesh Versus Tree

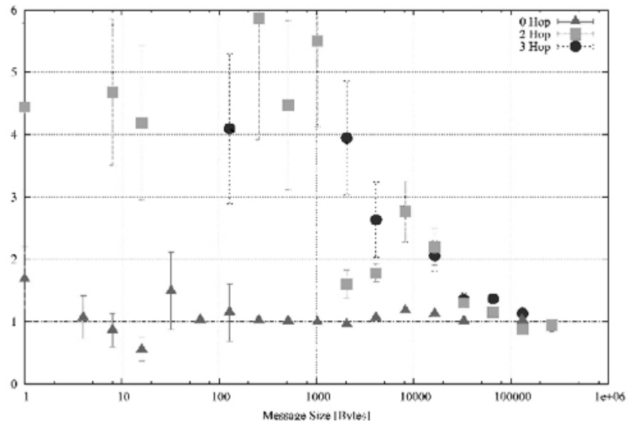

Figure 16. Linear scale plot of mesh versus tree performance ratios for 0-, 2-, and 3-Hop runs

Figure 16 presents the same information as in Figures 15 using a linear vertical scale. The MeshRouter 2-Hop and 3-Hop message delivery times typically are 2 to 4 times faster in the 1,000 byte to 10,000 Kbyte message range — a range including most messages in typical JSAF applications. It should be noted that these already significant performance enhancements largely are consequences of the fundamental topology differences noted in Table 1. As such, the MeshRouter performance increases will become even more dramatic as the underlying problem size increases beyond the simple 96-federate test case of Figures 11 and 12.

Typical Example of Gaussian Cumulative Probability

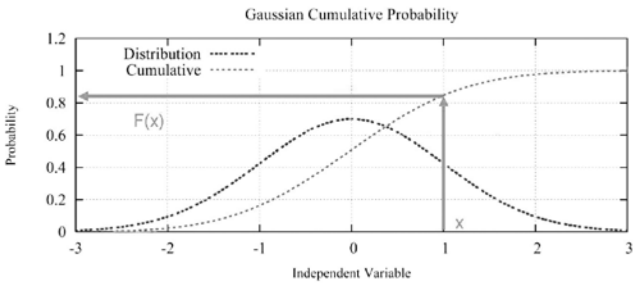

Figure 17. Portrayl of relationship of typical distribustion to cumulative probability

In order to explore causes for the Mesh⇔Tree differences seen in Figures 13–16, it is useful to look at the actual distributions in task times for the contributing rtiperf pairs. To set notation, Figure 17 compares the usual probability distribution function (the bell curve) and cumulative probability function (the curve asymptoting near one) for a standard Gaussian distribution. The value of the blue curve at any point X simply is the probability that the unit Gaussian distribution will yield a value at or below X.

The first panel of Figure 18 presents approximate cumulative probability distributions for message times for 8,000 messages using the MeshRouter. The three curves are nearly coincident, and, in the sense of Figure 17, indicate a rather narrow empirical distribution of message delivery times for the 48 contributing rtiperf pairs.

Distribution: 8Kbyte Message Delivery Time

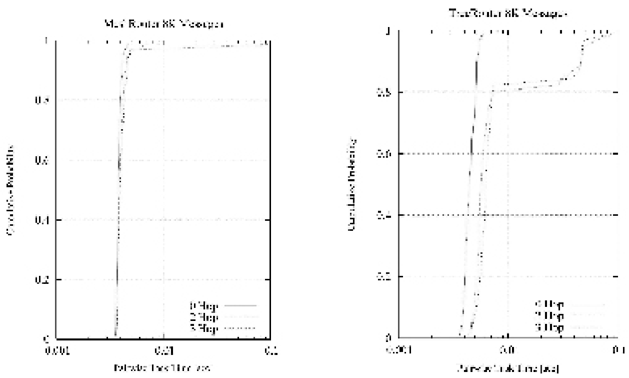

Figure 18. Performance of MeshRouters (left) is consistent; TreeRouters (right) degrade

The observed message delivery time distributions for the TreeRouter are qualitatively different in two important senses.

1. The fastest-time edges of the 2-Hop and 3-Hop distributions are significantly longer that the 0-Hop results. This is a direct consequence of the increased number of communication links in Table 1.
2. The plateaus in the 2-Hop and 3-Hop, up to 10 times longer, indicate a distinctly bi-modal nature in the distribution of message times, with about 80% of the rtiperf pairs completing the task in ~ 0.006 Sec., while the remaining 20% take much longer, ~ 0.05 Sec.

The bi-modal timing distribution indicates significant contention, as the individual messages are all pushed through a limited number of high-level routers. Put differently, 20% of the communications pairs are left waiting while the first arrivals get out of the way.

Average Point-to-Point Bandwidths

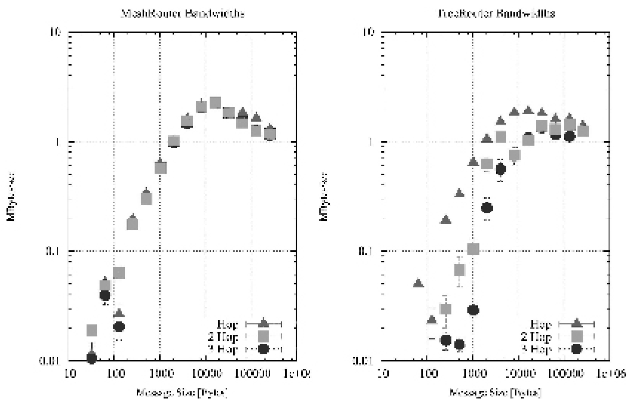

Figure 19. Log/log mesh and tree performance

A final, useful representation of the basic performance results is shown in Figure 19, where the (un-normalized) bandwidths,

B = [Message Size]/[Average Task Time],

are compared. These reinforce the same conclusions:

1. MeshRouter performance measures are remarkably insensitive to the Master/Slave separations.
2. TreeRouter performance degrades substantially as the underlying physical message path increases.

The MeshRouter scales, with point-to-point performance that is largely insensitive to the overall problem size. The TreeRouter performance degrades noticeably.

It should be added that the networks in Figures 11 and 12 actually are on the small side for networks of interest in current JSAF applications and in the near term. As the depth of the TreeRouter network in Figure 11 increases, the performance differences will increase even more, e.g., as seen in Figure 19.

5. Future Work

The MeshRouters currently provide a scalable solution for message routing in an RTI-s based federation. Future work will focus on fault tolerance, performance tuning, and investigation of supporting a fully compliant RTI implementation.

We have taken care to design a system that should allow plug-in adaptation to any RTI with a point-to-point communication infrastructure. Provided the client bounding assumptions are followed, the scalability shown for RTI-s should also apply to other RTI implementations. It is important to note, however, that a federation relying on timestamp message ordering will not see increased scalability with the MeshRouter architecture. Timestamp ordering requires all-to-all communication, placing enormous stress on the communication fabric. Previous experiments have shown abysmal scalability and the authors see no reason to expect any improvement using a mesh topology [15].

As the size of a simulation increases, the chance of failure in the network or hardware increases. With the ever-increasing size of simulations, the ability of the routing infrastructure to handle failures is becoming critical. The routers handle very little state, so the data loss when a router fails is not critical. However, until the router is restored, messages will not be delivered properly. If the lost router is the connection point for a site, a large portion of the simulation suddenly is not available. One potential solution is to allow loops in the mesh topology. This provides N + 1 redundancy for the connections, as there can be multiple paths between sites. If one path fails, the system will adjust and use the other available paths. The long-term solution is to provide an adaptive, dynamically configuring topology that adjusts to failures and new resources. The basic MeshRouter objects could accommodate these generalizations.

There are some common communication patterns for which the fully connected mesh is not well-suited (e.g., broadcast, which requires the router triad for the sending federate to contact every other router in its mesh). The solution is to use a hypercube or similar topology, which provides scalable broadcast capabilities while maintaining bisectional bandwidth. The work required to develop such topologies should be minimal, with most of the effort spent on reducing the work required to specify the topology.

6. Conclusion

The MeshRouter infrastructure presents a scalable routing infrastructure for both local and wide-area communication. The routers are capable of being organized into a number of topologies, and should be easily extensible into new routing topologies. For wide-area networks, the flexible routing topologies allow communication over all available network links, without the hub and spoke problem of the TreeRouters. MeshRouters provide a scalable communication architecture capable of supporting hundreds of federates within a local area network.

Perhaps more important than the demonstrated efficacy in this project, JUO, and on JSAF, is the ability to provide scalability to an existing 2,000,000-line program and to enhance scalability that approaches linearity. These dramatic increases in computing power as additional nodes are added present both a promise and a challenge to the modeling and simulation community. The promise is that of adequate compute power when needed; the challenge is to be open to a new understanding of where we are currently being artificially limited. While this paper has concentrated on the number of entities, the linearity of the scaling described also would enable advances along other dimension (e.g., size of simulated terrain, sophistication of entity behaviors, complexity of environmental phenomena, and the resolution of both entities and terrain).

7. Acknowledgements

We would like to thank the Joint Forces Command, especially the Joint Experimentation Directorate professionals for their support and encouragement. This work could not have been done without the unstinting backing and hard work of the teams at the Aeronautical Systems Center Major Shared Resource Center and the Maui High Performance Computing Center, the authors are indebted to the High Performance Computing Modernization Program for the use of computer resources for performance measurement. We also would like to thank Bill Helfenstein for advice on integrating the MeshRouter code with the RTI-s code base. Much of the performance testing was done with the indispensable assistance of Tammy Brown of the ASC MSRC staff.

8. References

[1] Davis, D., G. Baer, T. Gottschalk. 2004. 21st Century Simulation: Exploiting High Performance Parallel Computing and Advanced Data Analysis. *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*.

[2] Brunett, S., T. Gottschalk, T. 1998. A Large-scale Metacomputing Framework for the ModSAF Real-time Simulation, Parallel Computing 24.

[3] Messina, P., S. Brunett, D. Davis, T. Gottschalk. 1998. Distributed Interactive Simulation for Synthetic Forces. *Proceedings of the International Parallel Processing Symposium*. Geneva, Switzerland.

[4] Kaufman, W., L. Smarr. 1993. Supercomputing and the Transformation of Science. New York: Scientific American Library.

[5] Funkhouser, T.A. 1996. Network Topologies for Scalable Multi-User Virtual Environments. *Proceedings of the 1996 Virtual Reality Annual International Symposium (VRAIS 96)*. Washington, D.C.

[6] Ceranowicz, A., M. Torpey, W. Hellfinstine, J. Evans, J. Hines. 2002. Reflections on Building the Joint Experimental Federation. *Proceedings of the 2002 I/ITSEC Conference*. Orlando, FL.

[7] Lucas, R., D. Davis. 2003. Joint Experimentation on Scalable Parallel Processors. *Proceedings of Interservice/Industry Training, Simulation, and Education Conference*. Orlando, FL.

[8] Defense Modeling and Simulation Office. 1998. High Level Architecture Interface Specification, v1.3. Washington, D.C.

[9] Dahmann, J., J. Olszewski, R. Briggs, R. Weatherly. 1997. High Level Architecture (HLA) Performance Framework. Fall 1997 Simulation Interoperability Workshop, Orlando, FL.

[10] Messina, P., Davis, et al. 1997. Synthetic Forces Express: A New Initiative in Scalable Computing for Military Simulations. *Proceedings of the Distributed Interactive Simulation Conference*. Orlando, FL.

[11] Brunett, S., D. Davis, T. Gottschalk, P. Messina, C. Kesselman. 1998. Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments. *Proceedings of the Heterogeneous Computing Workshop*. IEEE Computer Society Press, 29–42.

[12] Calder, R.B., J.E. Smith, A.J. Courtemanche, J.M.F. Mar, A.Z. Ceranowicz. 1993. ModSAF behavior simulation and control. *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation.* Orlando, FL: Institute for Simulation and Training, University of Central Florida.

[13] MPI Forum. 1993. MPI: A Message Passing Interface. *Proceedings of 1993 Supercomputing Conference*. Portland, OR.

[14] Rak, S., M. Salisbury, R. MacDonald. 1997. HLA/RTI Data Distribution Management in the Synthetic Theater of War. *Proceedings of the Fall 1997 DIS Workshop on Simulation Standards*.

[15} Fujimoto, R. P. Hoare. 1998. HLA RTI Performance in High Speed LAN Environments. *Proceedings of the Fall Simulation Interoperability Workshop*.