The ITEA Journal of Test and Evaluation
Summer 2005

Joint Experimentation on Scalable Parallel Processors (JESPP)

Dan M. Davis & Dr Robert F. Lucas
Information Sciences Institute, USC
Marina del Rey, California

Dr. Philip Amburn
Science Applications International Co.
Wright Patterson AFB, Ohio

Dr. Thomas D. Gottschalk
California Institute of Technology
Pasadena, California

ABSTRACT

The JESPP project has demonstrated the value of High Performance Computing for forces modeling and simulation. The US Joint Forces Command's J9 has been tasked with creating a test environment for assessing operations of the future. This required expansion of the JSAF code capabilities: number of entities, behavior complexity, terrain resolution, infrastructure details, environmental realism, and analytical potential. Synthetic forces have been programmed to run in parallel on networked computers for decades. The JESPP strategy exploits the scalable parallel processors (SPPs) of the High Performance Computing Modernization Program (HPCMP). SPPs provide a large number of processors, inter-connected with a high performance switch and a collective job management framework. JESPP developed software routers that replaced multicast with point-to-point transmission of interest-managed packets. This article lays out that design and development. It also details several events that have simulated more than one million clutter entities, which were "fought" from Suffolk, VA. The models of these entities were executed on remote SPP's, one in Maui and one in Ohio. This paper sets forth the authors' experience in scoping the hardware needs, developing the project with HPCMP, and implementing the system.

AUTHORS

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute (ISI), University of Southern California, and has been active for more than a decade in large-scale distributed simulations for the DoD. While he was the Assistant Director of the Center for Advanced Computing Research at the Caltech, he managed Synthetic Forces Express, a multi-year simulation project. Prior to that, he was a Software Engineer at the Jet Propulsion Laboratory and he worked on a classified project at Martin Marietta. He saw active duty in Vietnam as a Cryptologist in the Marine Corps and he recently retired as a Commander, Cryptologic Specialty, U.S.N.R. He received B.A. and J.D. degrees from the University of Colorado in Boulder. ddavis@isi.edu

Philip Amburn currently works for Science Applications International Corporation as the Programming Environment and Training On Site for Forces Modeling and Simulation, with an office at Wright-Patterson AFB OH. His research interests are constructive and virtual simulation, interactive 3D graphics, and visualization. He is a Lieutenant Colonel, USAF-Ret. Dr. Amburn received a B.S. degree in Physics from Kansas State Teachers College, an MS degree in Computer Science from the Air Force Institute of Technology, and a Ph.D. degree in Computer Science from the University of North Carolina, Chapel Hill. philip.amburn@wpafb.af.mil

Thomas D. Gottschalk is a Lecturer in Physics at the California Institute of Technology and a Member of the Professional Staff, Center for Advanced Computing Research there at Caltech. For the last decade, much of his research has been on the use of parallel computers to simulate various physical phenomena. His instructional duties include his graduate-level course on Statistics for Caltech Physics students. Dr. Gottschalk received a B.S. in Physics from Michigan State University and a Ph.D. in Theoretical Physics from the University of Wisconsin. tdg@cacr.caltech.edu

Robert F. Lucas is the Director of the Computational Sciences Division of ISI at the University of Southern California. There he manages research in computer architecture, VLSI, compilers and other software tools. He has been the principal investigator on the JESPP project since its inception in 2002. Prior to joining ISI, he was the Head of the High Performance Computing Research Department for the National Energy Research Scientific Computing Center (NERSC) at LBNL. Earlier he was the Deputy Director of DARPA's ITO, and a member of the research staff of IDA's Center for Computing Sciences. Dr. Lucas received his B.S., M.S., and Ph.D. degrees in Electrical Engineering from Stanford University. rflucas@isi.edu

Joint Experimentation on Scalable Parallel Processors

Introduction and Background

This paper will set out a brief review of the needs, early development, and current status of the simulations of interest. It will then cover the architectural constraints and the designs that were implemented, followed by performance testing and analysis. Various tools and test methods are covered and a recitation of experiences is presented. The paper concludes with a look at future work and suggests some conclusions that may be drawn at this time.

The United States is facing an entirely new milieu of politics, economics, threats and conflict environments. (Barnett, 2004). This is forcing a deep and broad re-evaluation of its defensive posture. In looking to the future, the DoD has a vested interest in being able to simulate more than one million vehicles, all with sophisticated behaviors, operating on a global-scale, variable resolution terrain database. This is driven by the government's needs to accommodate new computer and communications technology (Cebrowski, 1998) and simulate more complex human functions (Ceranowicz, 2004) in technically diverse situations (Sanne, 1999). The U.S. Department of Defense (DoD) has begun a series of experiments to model and simulate the complexities of urban environments. In support of their mission, analysts need to conduct interactive experiments with entity-level simulations, using programs such as the Semi-Automated Forces (SAF) family used by the DoD (Ceranowicz, 2002).

This needs to be done at a scale and level of resolution adequate for modeling the complexities of military operations in urban situations. All of this leads to the government analysts' requirement of simulations of at least 1,000,000 vehicles or entities on a global-scale terrain database with high-resolution insets. Experimenters using large numbers of Linux PCs distributed across a LAN found that communications constraints limited the analysts to tens of thousands of vehicles, about two orders of magnitude fewer vehicles than their needs. This paper addresses the benefits of the successful application of computational science and parallel computing on SPPs to this situation. By extension, it lays out a template for those with similar simulation needs, but faced with similar computational constraints. They too can make beneficial use of the SPP assets of the High Performance Modernization Program (HPCMP.)

While there are many Forces Modeling and Simulation (FMS) approaches that are currently in use, simulation and modeling at the entity level (modeling each individual person and vehicle) provides some very attractive features, both for training and for analysis. Many of those who would argue that entity level simulations should be employed, maintain that these generate the most timely, most valid, and most cost-effective analyses. Making these simulations so that the DoD can involve humans, i.e. Human-in-the-Loop (HITL), additionally augments the DoD's ability to assess true impacts on personnel and procedures. (Ben-Ari, 1998) There are several new methods to modeling human behavior (Hill, 2000). While these require significant independent research (vanLent, 1998), they also require significant additional compute power. Current capability does not allow the analyst to conduct these experiments at the scale and level of resolution necessary and much of the FMS community do not currently report using High Performance Computing (HPC). (Davis, 2004) These constraints and disinclinations have also been found in other varieties of simulation. (Kaufman, 1993)

In the present case, newfound emphasis on civilian entities, often called clutter, has expanded the horizons of entity-count by ~ two orders of magnitude. Take any urban setting. The number of civilian vehicles will easily outnumber the combat vehicles by a factor of ten, and more likely, by a factor of 100. Trying to assess the utility of sensors and the efficacy of intelligence analysts in discriminating the former from the latter will be insufficiently served by a simulation that is limited to a few thousand vehicles total.

In order to make good use of the SPP assets currently available to DoD experimenters, the Joint Experimentation on Scalable Parallel Processor (JESPP) project applied approaches that others should find easily and reliably implementable on other, similar, efforts. The discussion of the implementation of the JESPP code into the JSAF code base will not only represent evidence of where JFCOM has been, but show the path for where they and others may go in the future.

The current work on the JESPP Linux clusters enabled successful simulation of 1,000,000 entities. Software implementations stressing efficient inter-node communications were necessary to achieve the desired scalability. One major advance was the design of two different software routers to efficiently route information to differing hierarchies of simulation nodes. Both the "Tree" and the "Mesh" routers were implemented and tested. Additionally, implementations of both MPI and Socket-Programmed variants were intended to make the application more universally portable and more organizationally palatable. The development of a visual and digital performance tool to monitor the distributed computing assets was also a goal that has been accomplished, leading to insights gained by using the new tool. The design and selection of program initiation tools for so large a simulation platform was problematical. The use of existing tools was considered less than optimal. The analytical process for resolving initiation issues, as well as the design and implementation of the resulting initiation tool developed by the group, is a demonstrable result of a computational science paradigm, the foundation for approaching such problems. The design constraints faced are analyzed along with a critical look at the relative success at meeting those constraints.

A truly interactive simulation requires a system that is scalable along the dimensions of complexity of entity behavior, total quantity of simulated entities, sophistication of environmental effects, resolution of terrain, and dynamism of features. This is a challenge that the authors assert may only be amenable to meta-computing across widely dispersed and heterogeneous parallel computer assets (Foster, 1997). Just achieving scalability in all of these dimensions would be difficult. Fielding a stable, dynamically reconfigurable compute platform is even more challenging. That platform may include large proprietary parallel computers, Linux clusters, PCs on LANs, legacy simulators, and other heterogeneous configurations producing new obstacles to implementation. Several unique and effective approaches are identified and explained.

The current work is based on the earlier work funded by DARPA in the mid nineties, headed by Paul Messina at Caltech (Messina, 1997). The Synthetic Forces Express project (SF Express) began to explore the utility of Scalable Parallel Processors (SPPs) as a solution to the communications bottlenecks that were then being experienced by one of the conventional SAFs, ModSAF. The SF Express charter was to demonstrate the capability of practically hosting a scalable communications architecture on multiple SPPs to simulate 50K vehicles: an order-of-magnitude increase over the size of an earlier major simulation, Synthetic Theater of War–Europe (STOW-E).

SPPs provided a much better alternative to networked workstations for large-scale ModSAF runs. Most of the processors on an SPP can be devoted to independent executions of SAFSim, the basic ModSAF simulator code. The reliable high-speed communications fabric between processors on an SPP typically gives better performance than standard switching technology among networked workstations. A scalable communications scheme was conceived, designed and implemented in three main steps:

1. Individual data messages were associated with specific interest class indices, and procedures were developed for evaluating the total interest state of an individual simulation processor.

2. Inter-node Communications were optimized: Within an individual SPP, certain processors were designated as message routers; the number of processors used as routers could be selected for each run. These processors received and stored interest declarations from the simulator nodes and moved simulation data packets according to the interest declarations.

3. WAN Communications were provided: Additional interest-restricted data exchange procedures were developed to support SF Express execution across multiple SPPs. The primary technical challenge in porting ModSAF to run efficiently on SPPs lay in constructing a suitable network of message-passing router nodes/processors. SF Express personnel implemented point-to-point MPI communications to replace the UDP socket calls of standard ModSAF. The network of routers managed SPP message traffic, effecting reliable interest-restricted communications among simulator nodes. This strategy allowed considerable freedom in constructing the router node network.

In March of 1998, the SF Express project performed a simulation run, with more than 100,000 individually simulated vehicles. The runs used several different types of SPPs at nine separate sites spanning seven time zones. These sites were linked by a variety of wide-area networks. (Brunett, 1997) All used 64 bit processors, but the meta-computing platform was heterogeneous with respect to "endian-ness," shared/distributed memory, processor design, switch topology, and other parameters.

This work depended on the existing DIS standard utilized by the SAFs at that time. That standard was replaced by the HLA/RTI standard that was purportedly more scalable, but several years of use has shown the clear limits of even this new simulation approach. This has not prevented some experimenters from getting very good results while simulating ~30,000 entities (Ceranowicz, 2002). These new standards and additional requirements have driven the development of two new router designs, Mesh Routers and Tree Routers.

JSAF

The Joint SemiAutomated Forces (JSAF) is used by the US Joint Forces command in its experimentation efforts. JSAF runs on a network of processors, which communicate, via a local or wide area network. Communication is implemented with High Level Architecture (HLA) and a custom version of Runtime Infrastructure (RTI) software version RTI-s. A run is implemented as a federation of simulators or clients. Multiple clients in addition to JSAF are typically included in a simulation. The typical interface is a map visualization, Figure 1

Figure 1
Plan View display from a SAF

HLA and RTI use the publish/subscribe model for communication between processors. Typically, these processors are relatively powerful PCs using the Linux operating system. A data item is associated with an interest set. Each JSAF instance subscribes to ranges of interest. A JSAF may be interested in, for example, a geographic area or a range of radio frequencies. When a data item is published the RTI must send it to all interested clients.

A typical JSAF run simulates a few thousand entities using a few workstations on a local area network (LAN). A simple broadcast of all data to all nodes is sufficient for this size simulation. The RTI on each node discards data that is not of interest to each receiving node. Broadcast is not sufficient when the simulation is extended to tens of thousands of entities and scores of workstations. UDP multicast was implemented to replace the simple broadcast. Each simulator receives only the data to which it has subscribed, i.e. in which it has a stated interest.

The simulation can also produce a three dimensional, rendered view of the urban area, as is shown in Figure 2.

Figure 2
3D Rendered display from a SAF

Operational imperatives drive experimental designs that now required further expansions of JSAF capabilities. As noted before, some of the requirements justifying these extensions are the need for:

- More entities
- More complex entities
- Larger geographic area
- Multiple resolution terrain
- More complex environments

The most readily available source of one or more orders of magnitude of increased compute power is the capability presented by Scalable Parallel Processors. In the JESPP project, JSAF was ported to run on multiple Linux clusters, using hundreds of processors on each cluster. Future runs will

require thousands of processors on multiple clusters. The primary difficulty in using these resources is the scaling of internode communication.

UDP multicast is limited to approximately three thousand different channels. Based on geography alone, worldwide simulations using JSAF require many more interest states. UDP multicast has been replaced and enhanced by point-to-point software routers.

Software routers were implemented on individual nodes on a local mesh network that includes all of the client simulators. Each simulator is connected to only one router. Routers are connected to multiple clients and multiple routers. Each connection is a two-way connection. Two types of information are present in the network. One is data along with interest description. The other is the current interest state of each client. The interest state changes as each node subscribes and unsubscribes to specific interest sets, as is appropriate depending on the simulation progress.

Each router must maintain the interest set of each node to which it is connected, including other routers. A router's interest set is the union of all connected nodes. A router then uses the interest state associated with data it receives to determine how to forward the data. For a given topology communication is minimized such that each client node receives exactly the data in which it is interested.

The initial router implementation was a tree router. Each router has multiple clients but only one parent. There is one router that is the top of the tree. A second topology has subsequently been implemented that the authors refer to as a mesh router. Instead of a single router at the top of a tree, there is a mesh of routers with all to all communication. Each simulator is a client of one of the mesh routers. Like the tree router, the primary task of the mesh router is to maintain the interest state of all clients so as to forward only data that is of interest to each client and router. Further hybrid topologies are possible with little or no code modification, such as a mesh of meshes or a mesh of trees. Conceptually, the mesh should provide better scalability.

Another use of routers is the implementation of gateways providing an interface between different RTI and communication implementations. Both TCP and UDP are used for communication. Routers can use a different protocol on different connections and perform required data bundling, unbundling, etc. Different RTI implementations, required by simulators developed by different groups, can communicate via router-based gateways.

The ultimate goal is for the capacity of a simulator network to scale easily as the number of processors is increased by several orders of magnitude. Comprehensive testing and measurement is required to document the performance of various topologies and router implementations. This testing will identify performance bottlenecks and suggest alternative implementations to be tested. Multiple simulation scenarios must be tested to construct guidelines for assigning simulators, routers and topologies to multiple SPPs.

Fault tolerance is another area being studied. JSAF simulators are not affected by the loss of other simulators. The use of routers creates a single point whose failure eliminates multiple simulators. The use of dynamic topology will be studied and implemented to minimize the consequences of single node failures. Several different concepts of providing redundancy or instantaneous recovery are being considered.

JFCOM's Experimentation Directorate's recent Joint Urban Operations (JUO) experiments have demonstrated the viability of Forces Modeling and Simulation in a distributed environment. The JSAF application suite, combined with the 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 communications in the large, distributed environments, and the current RTI-s framework provides such routers connected in a straightforward tree topology. This approach is successful for small to medium sized simulations, but faces a number of significant limitations for very large simulations over high-latency, wide area networks. In particular, traffic is forced through a single site, drastically increasing distances messages must travel to sites not near the top of the tree. Aggregate bandwidth is limited to the bandwidth of the site hosting the top router, and failures in the upper levels of the router tree can result in widespread communications losses throughout the system.

To resolve these issues, this work extends the RTI-s software router infrastructure to accommodate more sophisticated, general router topologies, including both the existing tree framework and a new generalization of the fully connected mesh topologies used in the SF Express ModSAF simulations of 100K fully interacting vehicles. The new software router objects incorporate the scalable features of the SF Express design, while optionally using low-level RTI-s objects to perform actual site-to-site communications. The (substantial) limitations of the original mesh router 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 can now be distributed across routers at the participating sites.

Large Scale Forces Modeling and Simulation

Recent experiments within the Joint Forces Command's Experimentation Directorate, J9, demonstrate the feasibility 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 utilize the JSAF application suite and the RTI-s Run Time Infrastructure to scale to over 300 federates distributed across the continental United States and Hawaii (Ceranowicz, 2002). 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 3: Software routing topology for the JUO exercise.

The current JUO network topology is a tree of software routers (see Figure 3 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 entity counts and complexity/fidelity increase.

Scalable Parallel Processors

The JUO exercise requires a computational ability unavailable using traditional groups of workstations. SPPs provide the required computational power, with modest increase in development and execution effort (Lucas, 2003). Common SPPs include the IBM SP, SGI Origin, Cray X1, and

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

Below, an improved routing architecture for large-scale HLA environments is presented, using fully connected meshes as the basic topology. These mesh routers provide a scalable solution for interest-managed communication, as well as a more accurate mapping of software routing to available network topologies.

RTI-s

RTI-s provides the HLA Run Time Infrastructure (RTI) for the JUO federation. RTI-s was originally developed to overcome the scalability and performance limitations found in RTI implementations at the time and even greater limitations found in the Distributed Interactive Simulation (DIS, IEEE 1278). RTI-s is arguably not a fully compliant HLA/RTI implementation, a matter of less and less consequence these days. Specifically, it does not implement timestamp ordered receives, ownership transfer, and Management Object Model (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 4), which allows for use over a number of communication infrastructures. Currently, there is support for multicast UDP, point-to-point UDP, point-to-point TCP, and MPI (using a send/receive architecture). Bundling and fragmenting of messages is provided by components that can be reused for TCP and for the typical, UDP communication. Kerberos authentication for data packets has been planned and should, by the time of this publication, have been implemented for TCP communication.

Figure 4: RTI-s data path architecture for TCP communication.

Point-to-point modalities in RTI-s use distinctly 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 5) is used for connecting routers. A tree presents a simple structure for preventing message loops, as there are no potential loops in the system.

Figure 5: Tree topology used by RTI-s for point-to-point message traffic. Synthetic Forces Express

The Synthetic Forces Express (SF Express) (Brunett & Gottschalk, 1998) project first demonstrated the suitability of both the SPP and mesh router concepts for discrete entity modeling. The SF Express project extended the ModSAF simulation engine (Calder, 1993), focusing on the communication protocols to extend scalability.

At the SC'96 conference in November 1996, the SF Express team achieved a 10,000-vehicle simulation using a single 1,024-node Intel Paragon machine. Message routing within the SPP used the Message Passing Interface (MPI) (MPI Forum, 1993). Later work allowed the code to run on multiple SPP installations across a variety of networks by introducing gateways between SPPs. The gateway routers were connected using UDP. With these improvements, the project was then able to field a simulation of 50,000 vehicles using 1,904 processors over six SPPs.

The structure of the SF Express router network is shown in Figure 6. 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 Figure 6.

Figure 6: Basic building block of the SF Express routing network.

As a final *magnum opus* accomplishment, the Caltech/ISI/JPL team was able to run a 108,000-entity simulation in March of 1998. These simulated entities were full ModSAF Operational vehicles: tanks, trucks, Bradleys, HumVees, etc. This demonstration was part of a DARPA meeting and had to be stable enough to run when scheduled. It used 13 different heterogeneous parallel computers, located from Maryland to Maui, spreading across a six time zones.

While the SF Express project was dramatically successful, it had no life beyond a number of 50K-100K entity simulation demonstrations. One issue was the need of one live operator to effectively control each 20 tanks. While the tanks behaviors were correct within local environments (Move North, shoot all enemy vehicles), they did not have sufficient behavioral sophistication to plan and execute long-range missions (proceed to next available target and approach surreptitiously, then either engage or move on depending on commander's judgment). For another 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. These issues were anticipated for a number of reasons, but did not detract from the concept proof offered by the success.

New missions have now made the provision of several million limited-functionality entities mandatory. They do not require either sophisticated behaviors or operator control. These are the so-called "clutter" entities that provide civilian entities, such as pedestrians, cars, motorbikes, *etc.* in the simulated environment.

.

Designing for Scalability

As previously mentioned, the JSAF/RTI-s application suite currently scales to over 300 federates and over a 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 the previous two points, the communication through any given router must also be bounded.[

However, any given router must also 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.

Interest Management

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 (Rak, 1997). 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 the application never sees messages for interest states it is not interested in. 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 (increasing 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 was also 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, the authors 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.

Routing Scalability

The scalability of the basic Mesh Router network is easily argued as follows. It is first necessary to assume that the underlying simulation problem itself has a scalable solution. This means a bounded message rate on the Primary ⇒ PopUp and PullDown ⇒Primary links within a basic triad, and bounded Up ⇒ 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 as many as thirty triads 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.

Routing Flexibility

The scalability issues with the tree router topology of RTI-s have been discussed previously. Tree topologies also map poorly onto physical wide-area networks. Figure 3 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 the authors' Maui to San Diego example.

Figure 7: Advanced routing topology for JUO exercises.

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 7 shows one possible routing topology for the JUO exercises, using mesh routers to minimize the distance messages must travel.

In an ideal world, the entire federation would use one fully connected mesh for message routing. The actual routing of messages would be left to the physical network infrastructure, which has over 30 years experience in optimizing data. However, such a configuration is often not feasible due to performance or protocol availability. Local area communication is usually over TCP, pushing error detection from RTI-s to the network stack. Over wide area networks, however, TCP suffers bandwidth degradation proportional to latency, so UDP is used for these connections. Some SPPs provide neither TCP nor UDP on computer nodes, instead providing MPI over a high-speed network) or provide public access only on a small subset of the machine. Given these restrictions, a fully connected mesh is often not a feasible design

Figure 8: The basic building blocks for a Mesh Router topology: tree (left) and mesh (right).

The mesh router provides the ability to design a flexible network topology that meets the constraints of the network infrastructure while providing the ability to design a scalable system. The mesh router's topology is constructed by combining two building blocks: a tree (Figure 8, left) and a fully connected mesh (Figure 8, right). The two building blocks can be combined to form meshes of meshes, trees of meshes, meshes of trees, etc. (Figure 9). The process can be repeated as often as required to build a suitable topology. The topology, however, cannot have any loops, as the routers are not currently capable of detecting this condition.

Figure 9: Combinations of the basic building blocks used to generate advanced routing topologies

Mesh Router Architecture

The mesh routers 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 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

Comparing Tree and Mesh Routers

In a nearly perfect metaphor for the decision to use HPC rather than just staying with workstation technology, one can easily decide that the straightforward Tree Router design is adequate, but eventually, one is confronted with the need to optimize. When the user demands new capabilities, more entities, or behavioral veracities, the potential benefits of the Mesh Router are not only desirable, they quickly become necessary.

How might one demonstrate these differences while waiting for the day when the user demands the immediate implementation of techniques capable of supporting their needed simulation environments? The needs are real and compelling, so waiting for them to be announced does not seem prudent.

One of the authors, Gottschalk, along with Brian Barrett, developed a performance analysis tool, described hereinbelow, to measure differences in the two-router architectures. After measuring the comparative rates across several germane message sizes, the following data was produced and is here presented as a chart comparing the relative performance of the two techniques.

Flow Control

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 mesh routers 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 a 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, a RTS/CTS protocol poses a significant performance burden. Therefore, the Mesh Router architecture has an eager send protocol with messages dropped by priority when queues overflow.

Application-Independent "Message" and "Interest" Objects

The Mesh Router 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 Mesh Router system is designed to be compatible with ongoing changes and evolution within the RTI-s system, requiring little more than "re-compile and re-link".

- The Mesh Router system can support applications other than SAF/RTI, given appropriate different instances of the message and interest objects.

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 8. 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:

- 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 10: High-level schematic of a router process (left) and dataflow of router/client connection (right).

These four flags are sufficient to reproduce the specific communications model of Figure 6 and a number of other networks, such as the tree router model available in the JSAF/RTI-s library, and the schematic Tree/Mesh mixture of Figure 7.

Factorized Communications Primitives

The Mesh Router 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 11 and has three layers:

Figure 11: Schematic design of the Mesh Router application.

Router Objects: These are little more that smart lists of objects associated with the clients in Figure 11. In normal operations, routers simply execute the fundamental message and interest manipulation methods for the associated clients. Routers are also 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.
- Client additions, in response to requests from external processes.

Client Objects: Managers of the interest declarations and pending message queues for each (external) client process.

Pipe Objects: The interface between the Message / MessageList formalism of the Mesh Router 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 Mesh Router framework.

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

- RTI-s Pipe: Message exchange using the RTI-s framework. (This object has been built entirely from objects and methods in the RTI-s library).
- Memory Pipe: Message "exchange" within a single process on a single CPU. This is used when two or more router processes in the sense of Figure 10 and Figure 11 are instanced as distinct objects within a single management process on a single CPU.

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

Router Configurations/Specifics, This Work

The numerical experiments described in this work explore two different overall communications topologies built from basic Mesh Router objects: the "Tree" and "Mesh" topologies shown in Figure 12.

Figure 12: Basic Topologies available using the Mesh Router

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 RTI-s Pipe instance. The persistent router clients in the sense of Section II are the upper router clients (if any) for each component router. All other communications links are generated dynamically.

For the Mesh Topology simulations, all three routers within the basic triad of Figure 6 are instanced as distinct Router objects on a single CPU, with Memory Pipe connections are used for the Primary $\Leftrightarrow$ PopUp and Primary $\Leftrightarrow$ PullDown links within a single triad. All other links in Figure 12 use the RTI-s Pipe, with the cross-triad PullDown $\Rightarrow$ Primary links persistent.

As noted, the current RTI-sPipe 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 Mesh Router 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 Mesh Router package supports mixed Pipe instances tailored to communications specifics for any of the individual links in Figure 12. 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 of this particular paper.

Results

The "koa" cluster at the Maui High Performance Computing Center was initially utilized for testing the Mesh Routers. "Koa" is a 128 node Linux cluster with two 3.06 GHz Intel Xeon processors and 4 gigabytes of memory per node. Nodes are interconnected 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 applicable to tree routers). The default configuration parameters were used for both RTI-s and the Mesh Router. Since the Mesh Router utilizes the RTI-s communication infrastructure, the authors 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, the authors show the Mesh Routers running in a tree configuration in addition to the standard RTI-s tree.

A number of tests ensured the Mesh Routers performed as required for JSAF experiments. The mesh infrastructure was used for an extended simulation using the JSAF suite. As expected for a small-scale simulation, the Mesh Router and RTI-s tree router were indistinguishable to the JSAF operator.

Latency values were measured on the "koa" cluster. The Mesh Router 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.

System Throughput

For testing the maximum throughput of the routing infrastructures, it was decided that pair-wise communication should be 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 13. For the test, 50 pairs were utilized, with 28 tree routers or 20 mesh routers creating the router infrastructure.

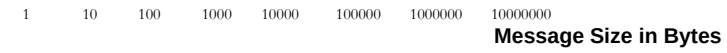

Figure 13: Realizable point-to-point bandwidth full communications load

As expected, Figure 13 shows that the maximum number of updates per second goes down as message size increases. The mesh router in a mesh configuration is able to move more traffic, and thereby cause more updates than either the RTI-s tree infrastructure or the Mesh Routers mapped into a tree topology. The RTI-s and Mesh Router 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.

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

Future Work

The mesh routers 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 Mesh Router architecture. Timestamp ordering requires all-to-all communication, placing enormous stress on the communication fabric. Previous experiments have shown abysmal scalability (Fujimoto, 1998) and the authors see no reason to expect any improvement using a mesh topology.

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 is suddenly 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 Mesh Router objects could accommodate these generalizations.

There are some uses leading to not-uncommon communication patterns for which the fully connected mesh is not well suited. One such pattern is a 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 a topology should be minimal, with most of the effort spent on reducing the work required to specify the topology.

Performance Testing Router Architectures

In a continuing effort to further quantify the relative utility of advanced router designs, a new set of performance runs were accomplished in the first quarter of calendar 2005. ASC-MSRC and the professional staff there supported these runs. They were monitored and assisted by the leadership of the PET organization from HPCMP. ASC-MSRC's Linux Cluster, Glenn, is basically the same configuration as Koa at MHPCC, but in the case of the cluster at Wright-Paterson, there are 60GB hard disks on each local node. This is a configuration subsequently installed in Kihei, Maui, on MHPCC's Koa.

Figure 14: RTI Performances Federates

The implemented simulation utilized for the Glenn performance studies again involved timed message exchanges between pairs of simple federates, as indicated in the schematic of Fig.(14). One processor within each pair initiates a sequence of fifty 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.

Figure 15: TreeRouter Test Configuration

This section presents the results that involve 96 total "rtiperf" federates (48 master-slave pairs). These are tested while they are communicating through router networks. Fig.(15) shows the TreeRouter network. 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 16: MeshRouter Test Configuration

Fig. (16), above, shows the corresponding MeshRouter connectivity/network. 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 Fig.(12). The entire triad is instanced on a single CPU, with software pipes ("MemoryPipes") that are seen as connecting Primary⇔Popup and Primary⇔PullDown within an individual triad. The shaded band in Fig.(16) represents the standard, full mesh connectivity among triads, as discussed above.

Next, the timing results are presented, based on three sets of runs for different separations of the Master-Slave "rtiperf" federates of Fig.(14). The three configurations are labeled "0 Hop", "2 Hop", and "3 Hop", according to 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 Figs.(15,16).

While it may be convenient to use the "n-Hop" labels, it may prove 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 Figs.(15,16).

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

Table 1: Number of distinct network links for "n-Hop" message exchanges in the network configurations of Figs.(13,14).

The TreeRouter almost invariably involves much longer communications paths than the MeshRouter.

A little analysis may be in order, before proceeding to measurement results:

1. The "Pipe" objects for Fig.(16) are "RTI-sPipes", 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 Figs.(15,16) are essentially identical, and the performance differences noted below are dominated by architectural differences.

Figure 17: Message Times versus Size for TreeRouters

Fig.(17) shows the basic data for the performance timing runs of TreeRouter configurations and the corresponding timing results for the MeshRouter configuration are shown in Fig.(18). The individual data points and error bars in these plots are evaluated for each message size/hop count as follows:

1. Fig.(17) sets forth the mean task times for the 48 contributing rtiperf pairs, which are sorted.
2. To minimize outlier-induced fluctuations, the two largest time values of each set are discarded,.
3. The data/error values included in the plots are the standard statistical means and standard deviations derived from the retained, 46-value samples.

Figure 18: Messages Times versus Size for MeshRouters

Figs.(17,18) contain several general features to be noted, which come from the results in:

1. The six distinct curves in the two plots 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 increased depth of the communications path.

Figure 19: Throughput Enhancement for MeshRouters versus TreeRouters (log)

Fig.(19) shows a more direct comparison of Mesh⇔Tree performance, where the ratios

$$R = [\text{Mean Tree Time}]/[\text{Mean Mesh Time}]$$

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 Figs.(15,16). 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 TCO/IP connection.

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

Figure 20: Throughput Enhancement for MeshRouters versus TreeRouters (linear)

Figure (20) presents the same information as in Fig.(19) using a linear vertical scale. The MeshRouter 2-Hop and 3-Hop message delivery times are typically 2-4 times faster in the 1Kbyte-10Kbyte message range - a range including most messages in typical JSAF applications.

Figure 21: Gaussian Cumulative Probability

Figs.(17-20) may imply the causes for the Mesh⇔Tree differences, but to further explore this, it is useful to look at the actual distributions in task times for the contributing "rtiperf" pairs. To set notation, Fig.(21) compares the usual probability distribution function (red curve) and cumulative probability function (blue curve) for a standard Gaussian distribution. The value of the blue curve at any point X is simply the probability that the unit Gaussian distribution will yield a value at or below X.

Fig.(22)'s left panel of presents approximate cumulative probability distributions for message times for 8K messages using the MeshRouter. The three curves are nearly coincident, and, in the sense of Fig.(21), indicate a rather narrow empirical distribution of message delivery times for the 48 contributing rtiperf pairs.

Figure 22: Distributions of 8KByte Message Size Delivery Times

The TreeRouter performance is qualitatively different with regard to the observed message delivery time distributions 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 results indicate a distinctly bi-modal nature in the distribution of message times, with about 80% of the rtiperf pairs completing the task in about 0.006 second while the remaining 20% take much longer (0.05 seconds).

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).

Figure 23: Average Point-to-Point Bandwidths

A final, useful representation of the basic performance results is shown in Fig.(23), where the (un-normalized) bandwidths

$$B = \{\text{Message Size}\}/[\text{Average Task Time}]$$

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.

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

Figs.(15,16) use networks that are actually on the small side of networks of interest for current and near terms JSAF applications. As the height of the TreeRouter network in Fig.(15) increases, the performance differences seen in, e.g., Fig.(23) will increase even more.

Performance and resource usage monitoring

Abstraction mechanisms found in many distributed programming systems enhance software reusability and interoperability by hiding the physical location of remote software processes. These abstraction mechanisms, which include HLA's concept of federates (Lightner, 1998) and CORBA's concept of components (Keahey, 1997), greatly reduce the complexity of accessing remote components. But, they come at the cost of reduced visibility, which hinders discovery of faults and impedes understanding of performance characteristics of the distributed system. This section describes a performance and resource usage monitoring tool Monitoring Remote Imaging (MRI) that aids developers in understanding the behavior of HLA simulations by displaying the monitoring data within the context of the execution of the distributed system. Similar specialized tools could easily be envisioned, designed and encoded for other simulations.

MRI displays monitoring data in the context of the federation connection topology. Figure 24 shows the screen dump of a MRI client's resource usage gauges displayed in the context of a three-level tree topology. The large oval pie chart at the top represents the root tree router. The set of rectangles underneath the root tree router represents sub-trees or router subgroups. Each subgroup has a tree router (medium-sized pie chart) connected to a set of federates (smaller pie charts). The first subgroup on the left as only one federate, and the other subgroups have eight federates.

Figure 24
Resource usage data of a JSAF federation displayed within the context of a tree connection topology.

In Figure 24 each CPU pie chart depicts the CPU usage breakdown for one compute node:

- Red for user-level CPU usage
- Blue for system-level CPU usage,
- Green for idle.

Each compute node has two CPUs, but the node is currently only running one process, so typically at most 50% of the CPU is used for non-thread applications like JSAF. At the snapshot when Figure 25 was taken the router nodes within the tree show substantial system-level CPU usage, which

indicates the routers are busy accessing kernel-level instructions to send/receive data. The federates in Figure 24 are only lightly loaded. Figure 25 shows alternative XY-plots for displaying time series data.

Figure 25
Plotting router network I/O as a function of time

MRI provides a framework for monitoring the performance and resource usage of federations at both at the OS-level and at the application federate level. Performance metric from both levels allows developers to correlate resource usage with JSAF simulation behavior. MRI display clients subscribe to monitoring relay gateways, which periodically push out the monitoring data. This monitoring data is represented in XML for extensibility and flexibility. At the OS-level it monitors the CPU load (user, system, idle), memory usage (user, share, cached, free) and network traffic (packets in/out, bytes in/out). Currently, for such OS-level information MRI uses Ganglia, a cluster monitoring tool from Berkeley's Millennium Cluster Project. At the application level it currently monitors JSAF's internal load, heartbeats, and various types of entity counts (remote, local, ground vehicle).

MRI maintains a representation of the federation connection topology in order to generate the gauge displays. This does not violate HLA's information hiding principles of reusability and interoperability, since this topology information is still hidden from JSAF federates, and the federates still have to communicate with each other using HLA RTI's communication infrastructure. The difference is that the connection topology, which is always a vital part of the HLA RTI, is now explicitly represented. Software researchers have argued that explicit representation of software architectures and topologies facilitates better reasoning and understanding [Garlan and Shaw, 1993]. For example, in the authors' case within the context of a topology the authors can determine the relative importance of node failures/bottlenecks. In Figure 26, node hn068 is highlighted with a yellow background indicating that it failed to emit monitoring data in a timely manner. The failure of node hn068 would bring down the 361 local entities that it is simulating. However, if instead the router node hn084 had failed, then it would have disconnected an entire subtree affecting 6040 entities. If the head router hn207 had failed, then it would result in a forest of disconnect subtrees. Current development is directed at preventing such losses.

Initiation Issues and the "SimPrep Tool"

A major issue when using multiple and geographically distributed SPPs is the effective coordination of intitiation, operation, and termination. There is a large body of research and development literature on various approaches to this issue. (Foster, 1997) While using these existing utilities and tool-kits may perhaps be the smoothest path to an effective implementation, the authors believe that this is one of the cases where a new tool may be desirable. To illustrate the definition of a need and the implementation of a new tool to serve that need, the authors will discuss the JESPP "Simulation Preparation" (SimPrep) tool. The authors do not suggest that it has the broad functionality of a tool-kit like Globus, nor is it suggested that other groups will need or want to develop individual tool-kits in every circumstance.

The preliminary objective of the JESPP exercise is to enable scalable multi-user simulations of synthetic semi-automated battles across multiple SPPs. Accompanying the authors' mission are challenging problems that the authors must address:

1. Overcoming geographical separation that is inherently problematic in terms of latency, and this experiment is particularly interesting due to the requirements of transporting a large amount of data between the clusters.
2. Accommodating the variation of SPP operational policy, e.g. security policy, software and configuration, and network constraints.
3. Implementing interactive computation in a meta-computing environment. This is a new challenge, and requires a new way of doing business. the authors need to operate the SPPs in interactive mode, as oppose to their traditional batch-mode model.

Solving the challenges above was accomplished against a backdrop of constraints, which included but were not limited to:

1. Trying to juxtapose between ease of use and flexibility. the authors' GUI application had to be flexible as scripting language scripts. While this challenge is not new to software implementers, they were nonetheless challenges.
2. Having to deal with continuous and large dataset – this along with the need to conduct precise metric. Traditional batch operation on a single or multiple SPPs, while collecting data concurrent to simulations, postpone processing to the post simulation stage.
3. Data collection codes had to behave in a non-intrusive manner and act only as observers without disturbing either the simulation or the collection process.

The experiment process can be decomposed down to four, disjointed processes; along with accompanying software tools we've developed to facilitate each of the stages (Table 2):

Stage	**Applications**
Abstraction stage Designing the network and communication topology, and do simulation preparation.	SimPrep and MARCI collector and MARCI GUI
Implementation stage Deploying the authors' software tools and applications to the SPP compute nodes	MARCI application suite deployed and launched applications
Execution stage Conducting the actual experiment by *game players*	JSAF applications, including tree router, JSAF and ClutterSim.
Analysis stage Studying and analyzing the exercise and performance and effectiveness analysis	MRI and post processing and logger tools

Table 2: Experimental Processes with Accompanying Software Tools

During the abstraction stage, the authors planed and designed the network topology. The authors were primarily interested in how each of the SPPs would be configured and connected (internally) as well as the network connections (externally) between them. To facilitate this process, which was extremely tedious and error-prone, the authors developed a software program called *SimPrep* that read in as an extensible configuration (network topology specification) file that utilized PERL programming syntax.

During the implementation stage, the authors used the MARCI applications to query the clusters for resources. Using the resource information and the configuration file defined (designed) in the abstraction stage, *SimPrep* performed resource allocation and map concrete actual compute nodes to abstract network layout.
There were two output files:
(1) the RID, a flattened connectivity file
(2) a mass launch file.

The RID file was in a LISP dialect and required to be manually stitched into a larger RID file and is understood by the JSAF, clutter, and router applications. The mass launch file was a MARCI specific instruction file on how to launch applications for a specific SPP. Note that the rules for different SPP are specified in the *SimPrep* configuration file.

Once the implementation stage was done, the exercise began. At this point the MARCI application took over. MARCI was responsible for starting and stopping applications – and specifically MARCI along with SimPrep served as the tool with which operators can interface and managed applications on an SPP interactively. This fact contrasted the authors' way of using the SPP with the traditional batch-processing model. The communications between the MARCI GUI and the MARCI collector is a socket-based communication on top of the SSL Layer and it used public/private key for message encryption.

The option to use Globus is limited to resource scheduling and resource discovery. As the experiment policy is still being shaped and defined, the authors felt that Globus would be better used when the authors' way of doing business is solidified. The authors also feel that Globus does not address the conduct of experiment, instead it serves to facilitate the experiment once the rules of engagement have been defined. For future experiments, the authors feel the Globus will play an important role – especially in the resource scheduling and discovery stage.

Accomplishments and Future Directions

In December of 2002, the JESPP team ran a successful prototype event using a partition of the USC IBM Linux cluster, consisting of some 240 IBM 335 servers, with 2 GHz Xeons, 1 GByte of RAM and both GigE and Myrinet mesh communications (see Figure 26). Both the scientists at ISI in California and the operators at JFCOM in Virginia jointly shared control. More than 1,000,000 civilian entities were successfully simulated. They showed appropriate behavior and were stable,

even when scanned by the SLAMEM program, emulating two GlobalHawk platforms. To ensure usability and operational validity, about 1,100 warfighting entities were also simulated and controlled in a manner consistent with normal J9 experimentation. Stability and appropriate response to control commands were evident throughout. Several runs were conducted over the course of a week and performance was characterized.

Figure 26
Conceptual diagram of December Prototype Event.

Following the December event, it was decided to show the utility of the DoD's SPP assets by using two Linux clusters, at two High Performance Computing Modernization Program sites. Two centers agreed to support this activity, the Aeronautical Systems Center (ASC) in Ohio and the Maui High Performance Computing Center (MHPCC) in Hawai'i. Maui had the larger resource in this case, a several hundred node IBM Linux cluster with Pentium III processors running at 933 MHz and with 1 GByte RAM per node. ASC's cluster was smaller, but exhibited similar processing parameters. With assistance from the HPCMP PET program, the Defense Research and Engineering Network (DREN) was used to interconnect MHPCC, ASC, and the Joint Forces Command in Virginia. Scalability and stability were recorded. Initiation and system configuration issues were studied and addressed.

The High Performance Modernization Program (HPCMP) established a new Distributed Center (DC) to provide computing assets for this work. The authors' experience in scoping the hardware needs for continued progress, the process of developing the joint DCEE Distributed Center project with HPCMP, and the current status of the system are all germane. The senior staff of the HPCMP were most helpful in discussions relating to the appropriate hardware for the task at hand. As an illuminating example, they raised the issue as to the potential benefits of installing 64 bit AMD or Intel processors, but were very sensitive to the ultimate customer's desire that the leap to high-end computing and parallel processing on Linux clusters remain as closely compatible with the existing Linux workstation configurations, all of which were 32 bit Intel processors. The system was installed in the spring of 2004 and became operational almost immediately. The installation contractor, Linux Networx, quickly worked to resolve a few minor communications and file systems issues.

Issues including the location of the Linux cluster, government security, networking bandwidth and latency, and cluster capabilities were surfaced and resolved. The desire for both redundancy and the desirability of further proving distributed high performance computing militated in favor of splitting the 256 nodes (two processors, 4GB RAM per node) into two machines, one at the MHPCC in Hawai'i and one at the ASC MSRC in Ohio. Both of these goals have been met and the original analysis has been proven prudent many times, when one site was able to carry the entire load while the other was faced with some type of outage, administrative encumbrance (*e.g.* Center power system maintenance) or network isolation. Government security was quite a sensitive area. The standing Government security was predicated on batch operations and did not easily accept the issues of real-time, interactive computing. Something on the order of an FTE (Full Time Equivalent staff person) was required just to evaluate, document, design, and socialize the solutions to these problems. Networking turned out to be a small factor and evidenced itself in unexpected ways. The underlying simulation program, one of the SAF family, tolerates latencies quite well. Even latencies up to 500

milliseconds (0.5 seconds) are tolerable. However, the speed-of-light-latency alone from Virginia to Maui is on the order of 100 milliseconds, and that was disorienting for the operators, not in the performance of the simulation, but in graphical user interface (GUI) issues like the delay in a drawn circle responding to mouse/cursor movements. Fully defining and delineating cluster capabilities was somewhat problematical, but the technical expertise and the professional good will of the operations staff quickly obviated that problem.

The appropriate teaming and staffing for this endeavor are critical and the projects success in overcoming hurdles. The multi-disciplinary force, supported by an effective management structure ensured making optimal use of new opportunities created by implementing large-scale simulations on Linux clusters were produced. The authors are completely confident in their vision of the future of scalable computing, especially large-scale Linux clusters, for military simulations. There can be a future expansion into grid and distributed high performance computing and they offer this work to support their view that the military simulation community would be well advised to pursue this path.

The group contributing to the JESPP project has made several noteworthy advances in high performance computing. the authors note the two-router designs, both of which merit further testing and use. Also, a fresh look at performance monitoring on heterogeneous and geographically dispersed SPPs has yielded a robust and useful tool that both generates data and presents status information in a visual manner that is useful for both the parallel processing experts and the simulation professionals. Some unique initiation problems have resulted in a new approach to complex synchronization issues not adequately addressed by either the SAF family software or by more general meta-computing tools.

Open issues for future work:

There is much to be done, of course, in terms of instrumenting and analyzing the existing system, contrasting performance with that from communications options within the current RTI-s baseline. The more interesting studies here will involve comparisons of new qualitative features of the underlying simulations. An example here is the difference between "reduced capability" and "self-aware" clutter (i.e., do clutter objects interact).

Many of the more interesting near-term development paths can be characterized in terms of "special purpose gateways" (now supportable in view of the reformulated Gateway models). Examples include:

- Translation Gateways: Processors to interpret and convert interest declarations among simulations (federates) that do not use a common interest-enumeration protocol.
- Visualization Gateways: Processors (quite possibly multi-processor collections) to request, collect, process and simplify (e.g., iconify) visualization data within very large simulations. (Current model does most of this work within the visualization workstations, giving rise to ample opportunity for death by communications overload.)
- Input Gateways: The "Collect, Preprocess, Summarize" objectives of the Visualization Gateway could be extended to other processes interested in large subsets of the simulation entities. An important example here is SLAMEM.

This was not merely a translation of existing (i.e., RTI-s) communications procedures. It was the first of a number of steps to achieve the qualitatively new capabilities that follow from:

1. The scalable communications capabilities of the basic architecture.
2. The additional capabilities of the "intelligent gateways" supportable within this architecture.

Conclusion

While the techniques proposed in this paper may not be a universal panacea, the authors maintain those techniques will increase the opportunity for the analysts to discover new and dangerous threats and work out the best way to defend against the destruction they carry. Considering the huge losses that might result if the nation is ill-prepared, the efforts required in implementing the described techniques pale in comparison.

The mesh router 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 tree routers. Within a local area network, the mesh routers provide a scalable communication architecture capable of supporting hundreds of federates.

Acknowledgements

This work was directed and funded by the Joint Experimentation Group at the Joint Forces Command and the authors wish to thank Major General Woods, Anthony Cerri, Jim Blank, and the entire J9 staff. The authors especially would like to acknowledge the direction, support and counsel of Rae Dehncke who has been the program manger and guiding light for this project. The authors would like to acknowledge the excellent technical support provided by the members of the Computational Sciences Division of the Information Sciences Institute, Gene Wagenbreth and John Tran, as well as the guidance and assistance from members of the Scalable Systems Division there, the Dr.s Ke-Thia Yao and Robert Neches. None of this could be accomplished without the help of the JSAF team Dr. Andy Ceranowicz, Mark Torpey, Bill Hellfinstine and many others. This material is based on research sponsored by the Air Force Research Laboratory under agreement number F30602-02-C-0213. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

References

Barrnett, T.P.M., (2004) *The Pentagon's New Map: War and Peace in the Twenty-First Century*, New York, New York, Putnam Company

Ben-Ari, E. (1998). Mastering Soldiers: Conflict, Emotions and the Enemy in an Israeli Military Unit, *New Directions in Anthropology*, V. 10. Oxford: Berghahn Books.

Brunett, S., Davis, D., Gottschalk, T., and Messina, P., (1998), Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments, *Seventh Heterogeneous Computing Workshop*, Orlando, Florida

Brunett, S., & Gottschalk, T., (1997), Scalable ModSAF Simulations with more than 50,000 Vehicles using Multiple Scalable Parallel Processors, Technical Report CACR-156, 1997, Caltech, presented at *Spring Simulation Interoperability Workshop*, 1998, Orlando, Florida

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

Cebrowski, A.K., & Garstka, J.J., (1998), Network Centric Warfare: Its Origin and Future, *Naval Institute Proceedings*, 124/1, 28-35.

Ceranowicz, A., Torpey, M., Hellfinstine, W., Evans, J. & Hines, J., (2002), Reflections on Building the Joint Experimental Federation, *Proceedings of the 2002 I/ITSEC Conference*, Orlando, Florida

Ceranowicz, A. & Torpey, M., (2004), Modeling Human Behaviors in an Urban Setting, *Proceedings of the 2004 I/ITSEC Conference*, Orlando, Florida

Davis, D., Baer, G., & Gottschalk, T., (2004), 21st Century Simulation: Exploiting High Performance Parallel Computing and Advanced Data Analysis, *Proceedings of the 2004 I/ITSEC Conference*, Orlando, Florida

Defense Modeling and Simulation Office, (1998), *High Level Architecture Interface Specification*, v1.3, 1998.

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

Foster, I. & Kesselman C., (1997), Globus: A Metacomputing Infra-structure Toolkit, *Intl Journal Supercomputer Applications*, 11(2): 115 –128

Fujimoto, R. & Hoare, P., (1998) HLA RTI Performance in High Speed LAN Environments. *Fall Simulation Interoperability Workshop*, September, 1998.

Garlan, D. and Shaw. M., (1993), An Introduction to Software Architecture: *Advances in Software Engineering and Knowledge Engineering*, volume I. World Scientific Publishing,.

Hill, R. W., Gratch, J., & Rosenbloom, P.S., (June, 2000). Flexible Group Behavior; Virtual Commanders for Synthetic Battlespaces. *Proceedings of the Fourth International Conference on Autonomous Agents*, Barcelona, Spain.

Kaufman, W. & Smarr, L., (1993), *Supercomputing and the Transformation of Science,* New York, Scientific American Library

Keahey, K. & Gannon, D., (1997), and PARDIS: A parallel approach to CORBA, Proceedings. The *Sixth IEEE International Symposium on High Performance Distributed Computing*, pp: 31-39, Portland, Oregon

Lightner, G.; Zeswitz, S.; & Graffagnini, J., (1998), Practical insights into the process of extending a federation-a review of the High Level Architecture Command and Control Experiment, Proceedings, *2nd International Workshop on Distributed Interactive Simulation and Real-Time Applications*, pp: 41-51, Montreal, Quebec

Lucas, R.. F. & Davis, D. M., Joint Experimentation on Scalable Parallel Processors. In *Interservice/Industry Training, Simulation, and Education Conference*, 2003

Messina, P. C., Brunett, S., Davis, D. M., Gottschalk, T. D., (1997, April) Distributed Interactive Simulation for Synthetic Forces, In J. Antonio, (Chair), Mapping and Scheduling Systems, International Parallel Processing Symposium, Geneva, Switzerland.

MPI Forum, (1993), MPI: A Message Passing Interface. In *Proceedings of 1993 Supercomputing Conference*, Portland, Washington, November 1993

Rak, S., Salisbury, M., & MacDonald, R., (1997), HLA/RTI Data Distribution Management in the Synthetic Theater of War, *Proceedings of the Fall 1997 DIS Workshop on Simulation Standards*, Orlando, Florida

Sanne, J. (1999). *Creating Safety in Air Traffic Control*. Unpublished doctoral dissertation, Institute of Tema Research, Linköping University, S-581 83 Linköping, Sweden.

van Lent, M. & Laird, K., (1998). Learning by Observation in a Complex Domain. *Proceedings of the Knowledge Acquisition Workshop*, Banff, Canada.