Extending The MeshRouter Framework for Distributed Simulations

Thomas D. Gottschalk
Center for Advanced Computing Research, Caltech
Pasadena, California
tdg@cacr.caltech.edu

Philip Amburn
SAIC, PET FMS On-site
Wright Patterson AFB, Ohio
philip.amburn@wpafb.af.mil

ABSTRACT

The MeshRouter system provides a general framework for scalable, interest-limited communications among processors in large-scale distributed simulations, such as the SAF family. The architecture was initially developed and implemented within the specific context of the ModSAF application and has recently been implemented in the JSAF/JUO application, using standard RTI-s communications primitives. This work provides a more general analysis of the MeshRouter system, clarifying the application-specific requirements for use of the communications framework and presenting a number of communications performance studies (total message throughput) for a system of simple federates using RTI-s communications. The overall MeshRouter architecture is reviewed, emphasizing the application-independent overall structure and the modest additional work needed to adapt the framework to the specific case of RTI-s communications.

The RTI-s MeshRouter is then compared with a tree-based communications built from standard RTI-s routers, using pair-wise message exchanges among simple federates. It is shown that MeshRouter performance is compatible with tree performance for trivial (e.g., nearest-neighbor) communications, and, more importantly, the aggregate bandwidth supported by the MeshRouter is substantially higher for non-trivial communications patterns, as would be expected in any realistic simulation environment. The communications performance studies are presented versus a number of relevant variables, including message size, total number of participating federates, and nominal length of the communications path. Extensions of the basic mesh topology used within the performance study are noted, including both modifications to support fault tolerance and a simple Tree/Mesh hybrid that could be easily implemented within the context of ongoing JSAF/JUO operations. Finally, the extensions of the existing MeshRouter software needed to support the OneSAF/RTI-N application are discussed.

ABOUT THE AUTHORS

Thomas D. Gottschalk is a Member of the Professional Staff, Center for Advanced Computing Research (CACR) and Lecturer in Physics at the California Institute of Technology. He has worked at CACR for nearly a decade on the use of parallel computers for simulation. His instructional duties include Statistics for Physics Graduate students. He received a B.S. in Physics from Michigan State University and a Ph.D. in Theoretical Physics from the University of Wisconsin.

Philip Amburn received his Ph.D. degree in Computer Science from the University of North Carolina, Chapel Hill, his MSCS degree in from the Air Force Institute of Technology, and a BS degree in Physics from Kansas State Teachers College. He currently works for SAIC as the Programming Environment and Training on-site for Forces Modeling and Simulation at Wright-Patterson AFB OH. His research interests are constructive and virtual simulation, interactive 3D graphics, and visualization.

Extending The MeshRouter Framework for Distributed Simulations

Thomas D. Gottschalk
Center for Advanced Computing Research, Caltech
Pasadena, California
tdg@cacr.caltech.edu

Philip Amburn
SAIC, PET FMS On-site
Wright Patterson AFB, Ohio
philip.amburn@wpafb.af.mil

I. INTRODUCTION

Forces Modeling and Simulation (FMS) has become increasingly important in analysis, training and evaluations. Particularly in the context of asymmetric warfare and/or urban environments, the required simulations are necessarily very large in a number of dimensions, including entity count, entity and terrain fidelity, and data management and exploitation.

Large problems require large, effective computational models. Distributed computing, including assets at widely distributed sites and Scalable Parallel Processors (SPPs) - such as large Linux Clusters - are standard tools for addressing these large computational issues. The effectiveness of this approach has been demonstrated in the Joint Urban Operations (JUO) experiments done by the Joint Forces Command (JFCOM) Experimentation Directorate (J9). These simulations used the JSAF application suite and the RTI-s communications framework to scale to hundreds of thousands of simulated entities in over 300 simulation federates distributed across the continental United States and Hawaii.

Effective inter-processor communication systems are essential for very large-scale discrete event simulations such as JSAF/RTI-s. The MeshRouter communications framework was introduced in (Barrett 2004) as a means of providing effective inter-processor message exchange that scaled (i.e., approximately uniform message exchange times) as the scope of the underlying simulation problem increases. Preliminary results supporting the scalability claim were presented.

This paper provides a more thorough presentation of the MeshRouter framework for scalable, interest-limited communications. The "historical roots" of the MeshRouter lie in the SFExpress distributed implementation of ModSAF, summarized in Section II. The revised framework involves a careful abstraction of the essential SFExpress concepts, reformulated to carefully isolate application-specific details from the overall communications architecture. The generalities of this factorization are presented in Section III and the rather modest effort needed to incorporate RTI-s communications within the framework are noted in Section IV.

Section V presents a detailed comparison of mesh-based and tree-based router network performance for RTI-s communications, using the 128-node Linux Cluster at the Aeronautical Systems Center Major Shared Resource Center (ASC MSRC) at Wright-Patterson AFB OH. It is shown that message exchange times within the mesh are largely insensitive to separations of communications partners. Put differently, the mesh network provides more uniform message delivery performance than that measured for a tree-based router network.

Sections VI and VII note immediately applicable extensions of the MeshRouter framework, and Section VIII contains some brief concluding remarks.

II. HISTORICAL CONTEXT: SCALABLE MODSAF AND SFEXPRESS

The MeshRouter formalism has its roots in the "Synthetic Forces Express" (SFExpress) (Brunett 1998, Messina 1998) scalable implementation of ModSAF done through DARPA in 1995-1997. The SFExpress challenge involved simulation of 10^6 fully interacting entities (no light-weight "clutter" entities). Workstations at that time could typically simulate $O(10^2)$ entities, so that the target simulation would require coordinated communications among $O(10^4)$ individual simulators. In contrast, the largest network-based SAF simulation at the time (STOW 1996) involved only a few thousand vehicles.

The SFExpress implementation used Scalable Parallel Processors (SPPs) to provide the required computational horsepower and solved the message management problems of very large-scale simulations through a network of "interest aware" router processes, as illustrated in Fig.(1). Collections of SAF simulators exchange messages with specified Primary Router processors, and additional "Pop-Up" and "Pull-Down" router layers provided scalable, interest-limited message exchange among the various sets of simulators. Interest management is accomplished by way of explicit interest declaration messages from the SAFs, with the router network collecting and providing only those message categories requested by the simulators. The SFExpress architecture was used in a number of successful, very large-scale simulation demonstrations in 1997-1998.

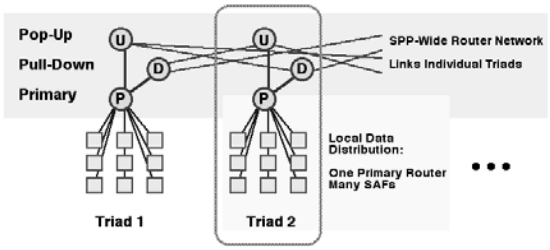

Figure 1: **Schematic illustration of the SFExpress communications architecture.**

While the SFExpress project succeeded in demonstrating a scalable framework for interest-limited message exchange in distributed simulations, the SFExpress capabilities were not incorporated into mainstream SAF simulations for a number of fairly obvious reasons associated with a "proof of concept" software demonstration. Among the more significant problems were the following:

1. Interprocessor communications within Fig.(1) were implemented using the standard Message Passing Interface (MPI) for SPPs. MPI is not particularly robust against individual processor failures.
2. The interest enumeration scheme implemented for SFExpress was somewhat ad hoc and restricted to geographical/proximity specifications.
3. Coordinated start-up for the wide-area "meta-computer" was tedious and operator intensive. The system did not support dynamic insertion/removal of participating processors.

The general MeshRouter communications formalism described in the next section resolves these and other issues, so that the demonstrated scalable communications capabilities of SFExpress can be incorporated into more general distributed simulations.

III. THE MESHROUTER FRAMEWORK

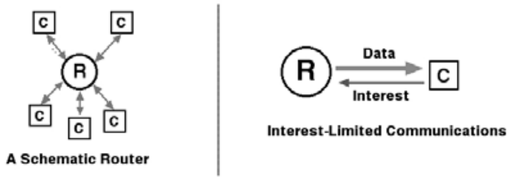

Figure2: **Router network schematics.**

Generalization of the specific SFExpress architecture of Fig.(1) begins best with brief considerations of the general interest-limited communications model shown in Fig.(2). It is assumed that

1. Individual client processes interact by message exchanges.
2. Messages are moved through explicit point-to-point communications with router processes.
3. Data messages can be characterized by "interest flags", specifying general categories of message content.
4. Client processes send "interest declarations", specifying interest categories needed by the client.
5. The router process screens messages sent to any client according to the interest declarations of that client.
6. The "clients' associated with the router in Fig.(2) could well be additional routers, thus supporting larger communications networks.

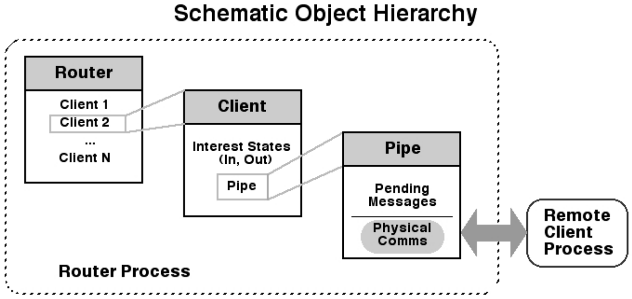

Figure 3: Fundamental objects in the MeshRouter system.

The MeshRouter architecture (Gottschalk 2004a) provides a general framework for interest-limited communications in which application-specific details are segregated into three broad categories:

- **Data Primitives:** General message characterizations (e.g., size) and, more

importantly, the interest specification/enumeration scheme.
- **Raw Communication:** Procedures for actual "bits on a wire" data exchanges along the links of Fig.(2).
- **Dynamic Client Management:** Procedures for recognizing and responding to entry and exit of individual client processes.

As is described in considerable detail in (Barrett 2004ab) and Gottschalk 2004ab), the MeshRouter framework supports this flexibility through a careful, object-oriented (C++) software design. The fundamental objects within the architecture are illustrated in Fig. (3).

Router Objects: These are essentially smart lists of objects associated with the individual external clients in Fig. (2). During normal, ongoing operations, the Router object simply loops over its associated clients, exercising basic data and interest manipulation methods. Routers also oversee dynamic client management tasks, including

1. Identification and removal of clients that have stopped communicating.
2. Initiation of communications links to specified (persistent) clients.
3. Client additions in response to communications requests from external clients.

Client Objects: There are internal representations of the external client processes of Fig.(2). Fundamental responsibilities include maintenance of interest declarations and queue management for incoming and outgoing message queues.

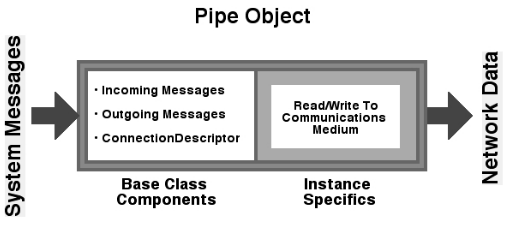

Figure 4: **General⇔Specific factorizations within the Pipe object.**

Pipe Objects: The essential interface between the internal data objects of the MeshRouter system and the real-world (bits on wire) communications procedures expected by the external applications. This "communications factorization" character of the Pipe is emphasized in Fig. (4).

As is emphasized in (Barrett 2004b), the overall software design within the MeshRouter system is flexible and intended to support a wide range of applications through the implementation of application specific instances of Data Primitives, Raw Communications, and Client Management tasks noted above. A brief overview of the required procedures for the JSAF/RTI-s application is provided in the next section.

IV: MESHROUTER SPECIFICS FOR JSAF/RTIS

The JSAF/RTI-s application (Ceranowicz 2002) provided the first concrete test of the general MeshRouter approach. Details can be found in (Barrett 2004ab, Gottschalk 2004ab). A brief overview of the implementation is as follows:

Data Primitives

"Messages" are regarded as little more than (possibly bundled) buckets of bytes, and the overall MessageInterpreter responsibilities described in (Gottschalk 2004b) are accomplished through interrogations of the standard RTI-s message headers. Interest is modeled using the bit-vector approach ("multi-cast emulate") within RTI-s. In particular, the "RtisInterest" daughter object needed within the MeshRouter framework is little more than a software wrapper around the RTI-s bitvector object.

Communications Primitives

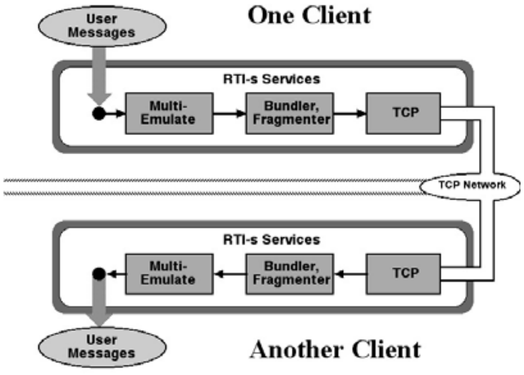

Figure 5: Construction of RtisPipe from RTI-s *dataflow_node* objects.

The "Instance Specific" component of Pipe communications in Fig.(4) is also constructed directly from standard objects within the RTI-s library. This is done rather easily using a multiple-inheritance (C++) approach in which the requited instance ("rtisPipe') is derived from both the MeshRouter Pipe base class and the RTI-s *dataflow_node* class. A schematic data flow within the hybrid object is illustrated in Fig. (5).

Of particular interest in the Fig. (5) schematic is the "Multi-Emulate" node, which implements interest management through enumerated interest state specifications within RTI-s. The standard RTI-s dataflow maintains client interest specifications whereas the overall MeshRouter framework requires that interest states be available at the level of the Client objects of Fig. (3). This was accomplished with very minor modifications of standard RTI-s operations within the dataflow paths.

Dynamic Client Management

The RTI-s provides extensive procedures/objects for "listeners" responding to new connection requests. The associated MeshRouter ConnectionManager objects are simply built from these existing RTI-s primitives.

It should be noted that both the communications primitives of Fig.(5) and the entire notion of dynamic client management assume access to a considerable amount of run-time system specification data. This is done using a standard data file ("RID") within RTI-s. The MeshRouter framework provides a simple mechanism for passing such applications specific data to the component objects of Fig.(3). Additional segments of the RID file were developed to support run-time specifications of mesh communications networks within standard JSAF/RTI-s applications.

The implementation of RTI-s instances for the general MeshRouter framework is described in considerably more detail in (Barrett 20094b). It should be noted that this particular implementation makes minimal extensions or modifications to standard RTI-s objects, with the effect of a number of potential inefficiencies (e.g., multiple memory copies for data messages). More efficient implementations are possible within the joint context of both the RTI-s and MeshRouter software packages.

V: Case Study: Tree Versus Mesh Communications for RTI-s

Performance of Mesh-based and Tree-based communications within RTI-s has been studied using a number of simulation studies on "Glenn", a 128 node Linux cluster with two 3.06 GHz Intel Xenon processors, 4 Gbytes local memory and a 60 Gbyte disk on each node. Glenn is located at the Aeronautical Systems Center Major Shared Resource Center (ASC MSRC) at Wright-Patterson AFB OH. The results reported in this section were supported by the staff there as well as the PET organization from HPCMPO.

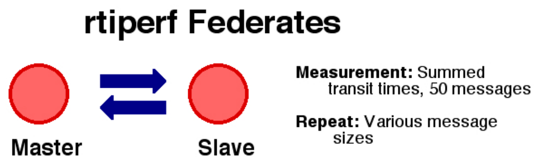
rtiperf Federates
Master
Slave
Measurement: Summed transit times, 50 messages
Repeat: Various message sizes

Figure 6: **Simple communications federates for the performance study.**

The underlying simulation for the performance studies on Glenn involved timed message exchanges between pairs of simple federates, as indicated by the schematic in Fig.(6). A test harness, originally developed by Brian Barrett, was modified to run on Glenn in batch mode, using the LSF batch scheduling system. The LSF script used the nodes assigned by the batch scheduling system and created the configuration files for the tree router and mesh router runs. For each LSF script execution, a tree router run was followed by a mesh router run. The configuration files partition the assigned Glenn nodes into pairs, with one master and one slave per pair. The master initiates a set of fifty round-trip, fixed size message exchanges with the slave, reporting the total wall clock time for the communications. This basic timed message exchange sequence is repeated for a variety of message sizes. Times for the message exchanges of various sizes were recorded in files for the analysis.

The runs reported here involve 96 total rtiperf federates (48 master-slave pairs) running and communicating simultaneously. Note that the test scenario involves essentially no real computations but is, instead, an almost pure "communications thrasher"

The 48-pair communications timing study was done for two different router networks: a tree-based network shown in Fig.(7) and a fully connected mesh network shown in Fig.(8).

In each configuration, the 96 simple federates are organized into groups of six associated with individual lowest-level routers. The lowest level routers for the Tree configuration are connected through a set of layered, higher level routers, as shown in Fig.(7). The timing results presented below are based on three different separations of the master and slave federates, labeled as “0 Hop”, “2 Hop” or “3 Hop” according to the depth of the TreeRouter communications path (ignoring the lowest level routers). Representative master-slave pairings for the three configurations are shown in Fig.(7). The actual router processes used in this configuration are the standard routers available within the RTI-s implementation.

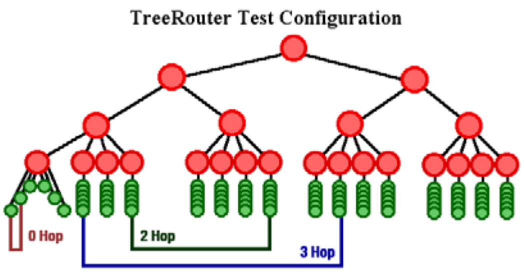

Figure 7: Communications network for the TreeRouter test runs.

The corresponding communications network for the MeshRouter is shown in Fig. (7). The ovals in this figure represent complete (Primary,Pop-Up,Pull-Down) triads in the sense of Fig. (1). Each triad is instanced as a single process (single processor) involving three distinct router objects in the sense of Section III. The Primary⇔Pop-Up and Primary⇔Pull-Down communications within a given triad are implemented using simple “MemoryPipe” objects for Fig. (4). As described in (Barrett 2004b), MemoryPipes

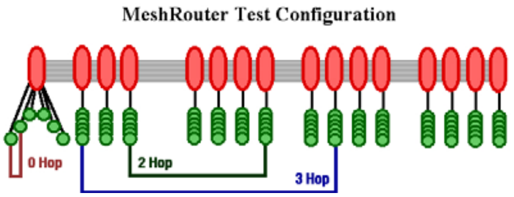

Figure 8: Communications network for the MeshRouter test runs.

are little more that coordinated message lists, involving essentially no communications overheads. The shaded band in Fig.(8) represents fully connected communications among the individual triads (full Pop-Up⇔Pull-Down connectivity). The inter-triad communications for Fig.(8) are implemented using the rtisPipe objects described above. That is, the lowest level “bits on a wire” communications implementations are the same for all links in Figs(7,9), and the only essential differences are the different point-to-point communications paths.

he “n-Hop” labels in Figs.(7,8) are convenient, but perhaps a bit misleading. Table 1 lists the number of distinct inter-processor (network) communications lengths for the actual message paths used in the study.

Table 1: Numbers of distinct network links for “n-Hop” message exchanges.

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

Finally, it should be noted that the standard RTI-s message bundling mechanisms have been disabled for this study - as needed in order to assess timing results for a variety of message sizes.

Basic Timing Results

The basic timing data for the TreeRouter configuration are shown in Fig.(9), with the corresponding results for the MeshRouter given in Fig.(10). The individual data points and error bars in these plots are evaluated for each hop/size configuration as follows:

1. The mean task times for the 48 contributing master-slave pairs are sorted.
2. The two largest values from each sorted set are discarded.
3. The data/error values in the plots are the statistical means and standard deviations for the retained 46-value samples.

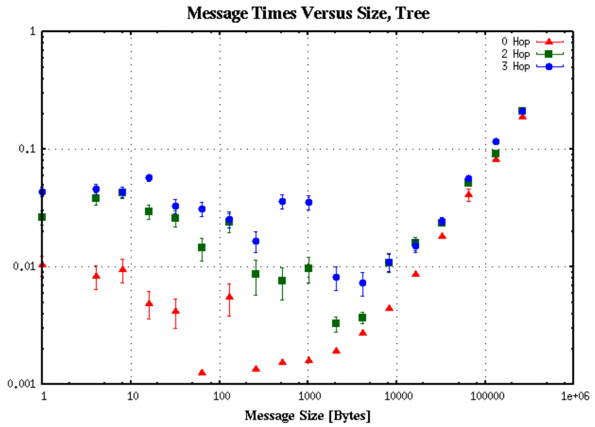

Figure 9: Basic timing results for TreeRouter Communications

The pruning of the two lowest times per data point reduces some sporadic random fluctuations, giving "tidier" plots without altering qualitative results.

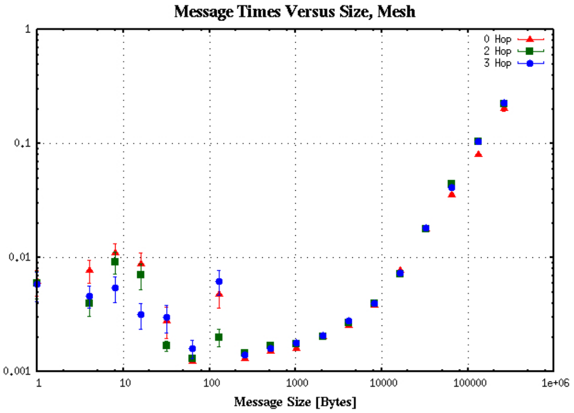

Figure 10: **Basic timing results for MeshRouter communications.**

There are a number of general features that can be noted from the results shown in Figs.(9,10):

1) The six distinct data sets in these two plots approach a common curve for very large message size. This is reasonable, and the rates for large messages are bandwidth limited (and the lowest-level RTI-s primitives provide efficient pipelining).

2) The 0-Hop, 2-Hop and 3-Hop results for the MeshRouter are remarkably similar. This is expected for the 2-Hop and 3-Hop cases, given the identical numbers of actual network messages in Table 1. The similarities of multi-hop and "0-Hop" times indicate that the MeshRouter procedure provides efficient, concurrent message distribution.

3) Except at the bandwidth-limited, large message size tails of the curves, the TreeRouter results show significant performance degradations (larger message transit times) for increased depth of the communications path.

4) The instances of large statistical error bars for some of the data points in Figs.(9,10) are in fact associated with communications contentions. Particularly for small message sizes, the start-up latency can be dominated by delays associated with busy network links.

A more direct comparison of Mesh⇔Tree performance is shown in Fig.(11), where the ratios

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

are plotted versus message size for the various n-Hop configurations.

Note that the 0-Hop ratio is essentially one throughout the entire range of message sizes. This is reasonable, since the 0-Hop messages never move beyond the lowest-level routers in Figs.(7,8), and the different higher-level communications topologies are irrelevant.

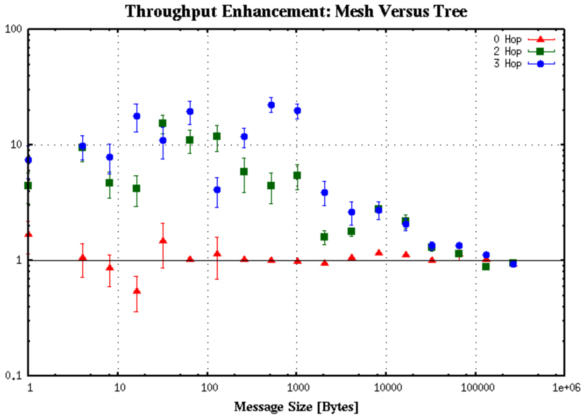

Figure 11: **Ratios (Tree/Mesh) of mean communications times versus message sizes.**

The observation that this empirical ratio is consistent with unity provides evidence that the MeshRouter framework of Section III introduces no significant operational overheads. The 0-Hop timing results in both Figs.(7,8) are dominated with operations of the TCP/IP connection class of RTI-s. The increased message delivery time for very small message sizes (latency/contention limited) and very large message size (bandwidth limited) are, in fact, as expected.

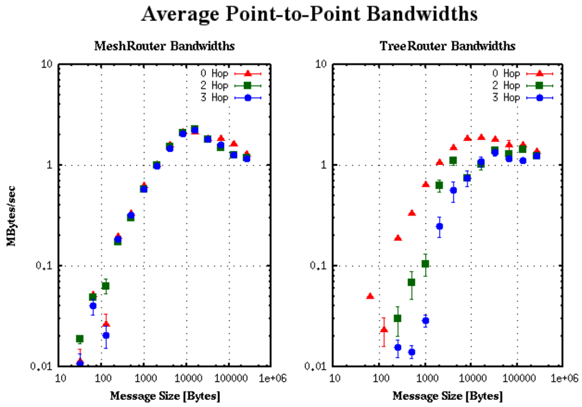

Figure 12: Bandwidth comparisons.

A final, perhaps more useful comparison of TreeRouter and MeshRouter performances is provided in Fig.(12), comparing the (unnormalized) bandwidths

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

These results reinforce the same general conclusions:

1. MeshRouter performance is remarkably insensitive to separations of the two participating processors.
2. TreeRouter performance degrades as the overall problem size (and typical underlying message path) increases.

It should also be noted that the networks/problem sizes in Figs.(7,8) are actually rather small in terms of current and near-term actual JSAF applications. As the height of the TreeRouter network in Fig.(7) increases, the performance differences seen, e.g. in Fig(12) will be exacerbated.

VI: EXTENSIONS/APPLICATIONS FOR JSAF/RTI-S

The procedures already in place for using the MeshRouter architecture with the standard JSAF/RTI-s application are, in fact, essentially complete with only a few formalities remaining to be addressed before the MeshRouter could be used within operational JSAF experiments.

The essential issues have to do with start-up procedures for large-scale distributed simulations. As noted above, JSAF/RTI-s executions are defined by a number of system-level data files - in particular the RID file. (Barrett 2004b) introduces simple extensions to standard RID file syntax to allow specifications of the fully-interconnected mesh topology of Fig.(8), and the runs reported in this work were in fact done using this simple generalization of standard JSAF/RTI-s procedures.

The remaining ‘nit’ has to do with automated start-up and/or configuration file generations for very large exercises. This is currently done using a separate procedure (“MARCI”). Generalizations of MARCI to spawn mesh triad processes would appear to be the largest remaining hurdle to full-scale incorporation of the MeshRouter within ongoing JSAF experiments.

It is suggested that the MeshRouter provides at least two aspects of “low-lying fruit” that would be of immediate benefit to JSAF/RTI-s.

The current JUO network topology is a tree of software routers spread across the country, as sketched in Fig.(13). In terms of the “worker nodes” at the primary computational sites (MHPCC and ASC), the height of the total tree network is rather greater than that of the performance study configuration of Fig.(7), so that bandwidth degradations greater than those shown in Fig.(12) are to be expected.

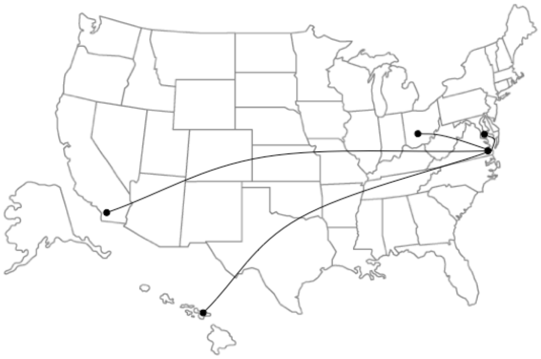

Figure 13: **Software routing topology for the JUO exercise.**

The physical networks available for JUO include a number of additional links between sites not involved in the Fig.(13) map. Adoption of a full MeshRouter approach could utilize these links in a transparent manner. There is, moreover, a somewhat simpler “partial mesh” implementation that could minimize the large-distance communications issues of Fig.(13) without modifying operations at the individual computational sites.

This involves a hybrid “TransContinental Mesh” architecture, as illustrated in Fig.(14). The essential elements of this architecture are as follows:

1. The root nodes in the conventional tree router networks at the individual sites are replaced by full triads.
2. Nodes below the root function/communicate as usual.
3. The root triads are fully interconnected, thus exploiting/utilizing all available real communications links.

Since the “non-standard” triad processors are limited to one at each site, the generalizations of start-up procedures to accommodate the mesh triad processes would be reduced.

A second immediate benefit of MeshRouter communications has to do with robustness against router failures. In either Fig.(7) or Fig.(8), failure of a router process severs communications for processes below the router, with the loss of the root node particularly catastrophic in the case of the TreeRouter. In contrast, the effects of router failures are far more localized for the MeshRouter configuration.

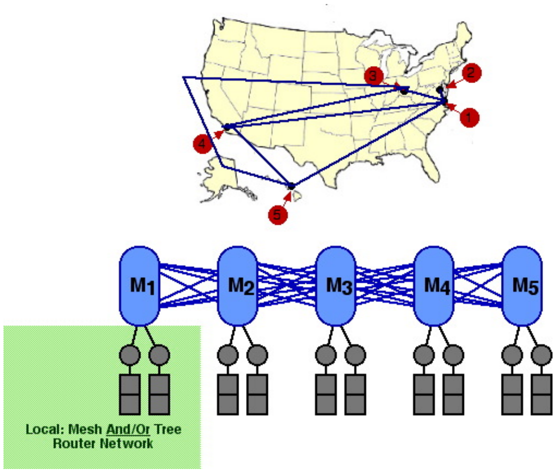

Figure 14: Schematic illustration of a hybrid "TransContinental Mesh" communications framework.

Robust systems for either the Tree or Mesh configurations would require some collections of "hot backup" routers and dynamically reconfigurable routing information. The essential issues involve specifications of the alternative routes - issues that would be the same for either mesh or tree configurations.

VII: EXTENSIONS/APPLICABILITY FOR ONESAF/RTI-N

JSAF/RTI-s provided both the impetus for the original MeshRouter development (in the context of the JUO project) and the first concrete implementation test case. OneSAF Objective System (OOS) is the US Army's next-generation constructive simulation, and is scheduled for initial release in October 2005. OneSAF has been ported and tested on HPC resources, and the experience of US JFCOM with JSAF and the mesh router technology can be provided within OneSAF. Initial investigations with a pre-release version of OneSAF have been successfully completed with behaviors for urban non-combatants. Given the lessons learned from this initial implementation, extensions of the MeshRouter to OneSAF/RTI-N are expected to be fairly straightforward.

As was the case with RTI-s, the ease of porting to RTI-N will rely on the careful factorization of Framework⇔Application components highlighted in Section III and described in more detail in (Barrett 2004b). The implementation strategy/schedule for an RTI-N implementation would accordingly involve a number of simple steps:

- Identification of interest enumeration schemes and construction of an appropriate RtiNInterest object within the MeshRouter framework.
- Construction of a specific "RtiNPipe" class - presumably built from RTI-N analogues of the dataflow nodes utilized in Fig.(7) - to incorporate low-level RTI-N communications procedures within concrete MeshRouter communications Pipes.
- Construction of RTI-N-specific "ConnectionManagers" to listen for and respond to new communications requests directed at the routers.
- Generalizations of existing OneSAF/RTI-N configuration files and start-up procedures to include more general communications networks.

There will undoubtedly be a number of "annoying complications", depending on the nature of RTI-N interest specifications and availability of these internal interest states to outside processes. However, given the common heritage of RTI-s and RTI-N, it seems unlikely that any of these complications will prove to be substantial.

The DOD HPCMPO Programming Environment and Training has a project scheduled through May 2006 to provide elements of the Army Constructive Training Federation (ACTF) on HPC resources. The primary thrust of this project is the porting and optimization of WARSIM and OneSAF, two of the major components of the ACTF, to HPC-class machines. Included in this effort is incorporating the mesh router technology in OneSAF. This effort will be done on a Linux cluster called Powell at the ARL MSRC, which has recently been made available for interactive use, such as the human-in-the-loop simulations. Including mesh router technology in OneSAF positions it to better participate in urban simulations with large entity counts.

VIII: CONCLUDING REMARKS

This work has provided a thorough discussion and analysis of the MeshRouter scalable communications architecture introduced in (Barrett 2004a). The primary conclusions of this work can be stated as follows:

- The MeshRouter model provides a general procedure for interest-limited message exchange in large-scale distributed simulations.
- The package provides an effective factorization of application-dependent details within a generally applicable framework.
- Compared to a straightforward tree-based router network, the MeshRouter provides message distributions at rates that are more uniform.

The temptation exists to restate the last conclusion as "MeshRouters scale better than TreeRouters". This is actually a bit misleading, as "scaling" in large distributed simulations ultimately depends on a number of factors outside of communications *per se* (e.g., the average interest level of individual entities and the attendant CPU demands on the hosting processors).

The more important and defensible conclusion based on the analyses of Section V is the statement that

> **MeshRouter Message exchange overhead remains constant as the number of participating simulators increases.**

This property is not characteristic of tree-based router networks. This more limited sense of "size independence" is a clear requirement for scalability. The MeshRouter formalism provides this "pre-scaling" communication capability for a variety of general simulations used within Forces Modeling and Simulation (FMS) activities.

The Tree-versus-Mesh results presented in Section V are hardly surprising. Instead, as discussed in Section V, the better MeshRouter performance is expected given the fixed lengths of communications paths, as noted in Table 1. The better performance is ultimately a result of maintaining and managing many concurrent, fixed-length message paths in place of the hierarchical, longer message paths in the tree network.

The multiple-path character of the MeshRouter has another important advantage: robustness against router failures and restarts during the course of a long simulation. In both the TreeRouter and MeshRouter configurations of Figs.(7,8), the failure of a router process severs communications to all simulators lying below that router. All router processes in that Mesh framework have limited "underlings", and the effects of individual router failures/restarts are accordingly localized. In contrast, failures of upper layer routers in the tree architecture have widespread consequences.

The better message delivery times and limited router failure effects make the MeshRouter framework a natural choice for very large-scale message-based simulations. As noted in Sections VI and VII, a number of specific extensions and applications of the framework are currently under investigation.

ACKNOWLEDGEMENTS

We than the Joint Forces Command (JFCOM) and Joint Experimentation Directorate (J9) for support and encouragement during this work. The detailed studies presented here were done using facilities at the Aeronautical Systems Center Shared Major Resource Center. In particular, we thank Tammy Brown of the ASC MSRC staff for assistance in performing most of the runs analyzed in this work.

REFERENCES

Barrett, B. & Gottschalk, T. (2004a), Advanced Message Routing for Scalable Distributed Simulations, Proceedings of the 2004 I/ITSEC Conference, Orlando Florida.

Barrett, B. and Gottschalk, T. (2004b), Pipes and Connections, Caltech/CACR Technical Report 2004.213

Brunett, S. & Gottschalk, T. (1998) A Large-Scale Metacomputing Framework for the ModSAF Real-time Simulation, *Parallel Computing* 24.

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

Gottschalk, T. (2004a), The MeshRouter Architecture, Caltech/CACR Technical Report 2004.212.

Gottschalk, T. (2004b), MeshRouter Primitives: Messages, Interest, and Interpreters, Caltech/CACR Technical Report 2004.214.

Graebner, R., Rafuse, G., Miller, R. & Yao, K.-T. (2003) Proceedings of the 2003 I/ITSEC Conference, Orlando Florida.

Lucas R., & Davis, D. Joint Experimentation on Scalable Parallel Processors (JESPP), Proceedings of the 2003 I/ITSEC Conference, Orlando Florida.

Messina, P., Brunett, S., Davis, D., & Gottschalk, T. (1998), Distributed Interactive Simulations for Synthetic Forces, Proceedings of the International Parallel Processing Symposium, Geneva, Switzerland.

Synthetic Theater of War (STOW 1996) Engineering Document 1A (ED-1A) Analysis Report, NRaD Report, 1996