Advanced Message Routing for Scalable Distributed Simulations

Brian Barrett
University of Southern California
Marina del Rey, CA
bbarrett@isi.edu

Thomas Gottschalk
California Institute of Technology
Pasadena, CA
tdg@cacr.caltech.edu

ABSTRACT

The Joint Forces Command (JFCOM) Experimentation Directorate (J9)'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 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.

ABOUT THE AUTHORS

Brian Barrett is a programmer analyst on the JESPP project, Information Sciences Institute, University of Southern California. Brian's research has focused on communication issues for large-scale high performance computing systems. While at Indiana University, Brian was a lead developer of the LAM/MPI implementation of the Message Passing Interface (MPI) standard. He received a B.S. from the University of Notre Dame and an M.S. from Indiana University, both in Computer Science.

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 been with CACR for nearly a decade. Much of his research has been on the use of parallel computers to simulate various physical phenomena. His instructional duties include his upper division course on 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.

Advanced Message Routing for Scalable Distributed Simulations

Brian Barrett
University of Southern California
Marina del Rey, CA
bbarrett@isi.edu

Thomas Gottschalk
California Institute of Technology
Pasadena, CA
tdg@cacr.caltech.edu

Abstract

The Joint Forces Command (JFCOM) Experimentation Directorate (J9)'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 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 (JFCOM) 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 1: Software routing topology for the JUO exercise.

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

This paper presents an improved routing architecture for large-scale HLA environments, 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.

Scalable Parallel Processors

The JUO exercise requires a computational ability unavailable using traditional groups of workstations. Scalable Parallel Processors (SPPs) provide the required computational power, with modest increase in development and execution effort (Lucas, 2003). A SPP is a large collection of processing elements (nodes) connected by a fast communication network. Common SPPs include the IBM SP, SGI Origin, Cray X1, and Linux clusters. Traditionally, SPPs provide services not available in a group of workstations: high speed networks, massive disk arrays shared across the entire resource, and large per-CPU physical memory. In addition, SPPs generally have uniform environments across the entire machine and tools for scalable interactive control (starting processes across 100 nodes takes the same amount of time as it does across 10).

Linux clusters have recently become a suitable platform for the high performance computing community and are therefore readily available at Department of Defense Major Shared Resource 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.

RTI-s

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

RTI-s utilizes a flexible data path framework (an example of which is shown in Figure 2), 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 UDP communication. Kerberos authentication for data packets has been implemented for TCP communication.

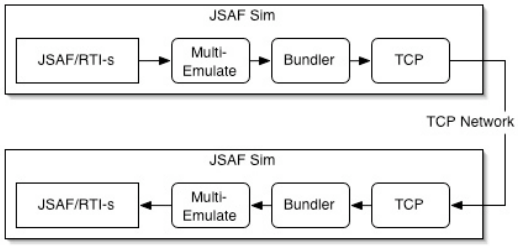

Figure 2: RTI-s data path architecture for TCP communication.

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

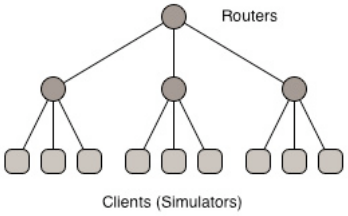

Figure 3: Tree topology used by RTI-s for point-to-point message traffic.

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.

In December 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 achieved 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 4. The basic building block for this

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

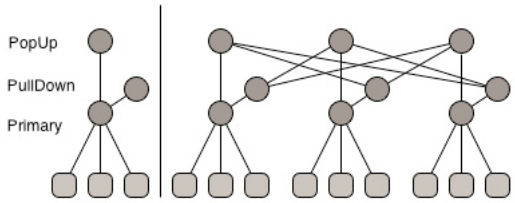

Figure 4: Basic building block of the SF Express routing network (left) and an example mesh topology (right).

While the SF Express project was quite successful, it had no life beyond a number of 50K-100K entity simulation demonstrations. This was expected, for a number of reasons. For example, the algorithms and software developed for that project were not compatible with ongoing SAF developments (e.g., the move to RTI). Finally, the MPI-based communications used within the SPPs did not tolerate the restarts and process failures found during a long running exercise.

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

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, we assume that the simulator limits interest declarations to guarantee bounded communication. In both the earlier SF Express and current JUO experiments, this assumption appears valid.

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

Figure 5: 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 5 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.

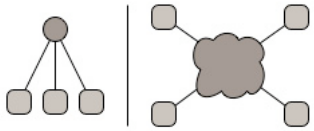

Figure 6: The basic building blocks for a Mesh Router topology: tree (left) and mesh (right).

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 6, left) and a fully connected mesh (Figure 6, right). The two building blocks can be combined to form meshes of meshes, trees of meshes, meshes of trees, etc. (Figure 7). 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.

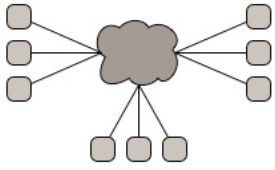

Figure 7: Combinations of the basic building blocks used to generate advanced routing topologies.

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.

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 that "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).

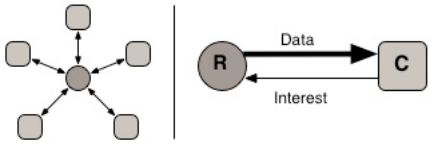

Figure 8: High level schematic of a router process (left) and dataflow of router/client connection (right).

These four flags are sufficient to reproduce the specific communications model of Figure 4 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 9 and has three layers:

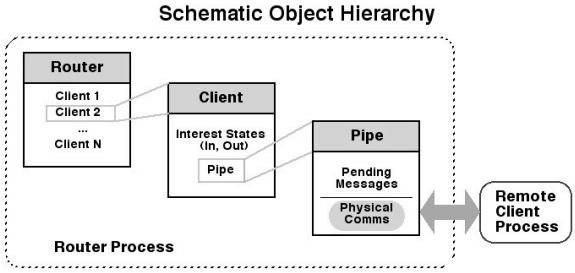

Figure 9: Schematic design of the Mesh Router application.

Router Objects: These are little more that smart lists of objects associated with the clients in Figure 9. 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:

- RtisPipe: Message exchange using the RTI-s framework. (Indeed, this object has been built entirely from objects and methods in the RTI-s library).
- MemoryPipe: Message "exchange" within a single process on a single CPU. This is used when two or more router processes in the sense of Figure 8 and Figure 9 are instanced as distinct objects within a single management process on a single CPU.

The 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 10.

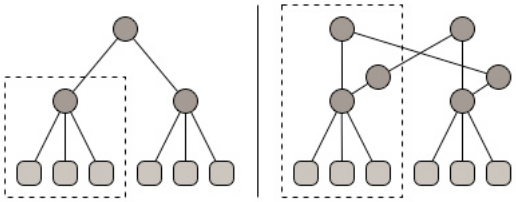

Figure 10: Basic topologies available using the Mesh Router.

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 RtisPipe 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 4 are instanced as distinct Router objects on a single CPU, with MemoryPipe connections are used for the Primary ⇔ PopUp and Primary ⇔ PullDown links within a single triad. All other links in Figure 10 use the RtisPipe, with the cross-triad PullDown ⇒ Primary links persistent.

As noted, the current RtisPipe implementation is based entirely on objects and method calls within the current RTI-s library. This is important for demonstrating "ease of insertion" of the 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 10. 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 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, we believe that any parameter tuning done to one system would apply equally well to the other system. To highlight the importance of topology in routing infrastructure, we show the 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 measurements were taken 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, pair-wise communication was used. Attribute updates were sent between process pairs as fast as possible, with loose synchronization to ensure multiple pairs were always communicating. The average per-pair throughput, specified in number of `reflectAttributeValues()` calls per second for a given message size, is shown in Figure 11. For the test, 50 pairs were utilized, with 28 tree routers or 20 mesh routers creating the router infrastructure.

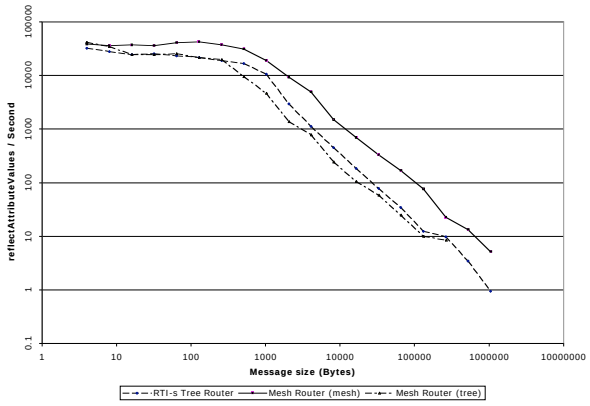

Figure 11: Realizable point-to-point bandwidth full communications load

As expected, Figure 11 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. We 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 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.

Conclusion

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

We would like to thank the Open Systems Laboratory at Indiana University, Aeronautical Systems Center Major Shared Resource Center, and the Maui High Performance Computing Center for the use of computer resources for performance measurement. We would also like to thank Bill Helfenstein for advice on integrating the mesh router code with the RTI-s code base.

REFERENCES

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

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

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.

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.

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

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

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

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

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

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