

# An Architecture for Large ModSAF Simulations Using Scalable Parallel Processors

Sharon Brunett and Thomas Gottschalk  
Center for Advanced Computing Research  
California Institute of Technology  
Pasadena, CA 91125  
sharon@cacr.caltech.edu, tdg@cacr.caltech.edu  
626-395-6734, 626-395-6671

KEYWORDS: Parallel Processing, Scalable, Router Network, Interest Management, Communications Architecture

**ABSTRACT:** *An implementation of ModSAF for Scalable Parallel Processors (SPPs) is presented. This model exploits the large number of processing elements and fast interprocessor communications of SPPs to simulate many thousands of vehicles on a single SPP. The implementation uses a heterogeneous assignment of tasks to processors, with most processors running independent copies of the standard SAFSim code and additional processors used as either data servers or message routers. The key element of the architecture is the collection of router processors that move interest-restricted messages among the individual SAFSims at rates far higher than usually seen for ModSAF runs with networked workstations. The router architecture has been implemented using the standardized and portable Message Passing Interface (MPI) and has been run successfully on a variety of different platforms. The scaling behavior of the model is analyzed and measured performance results are presented for runs involving up to 16,000 simulated vehicles. Generalizations of the currently used interest management rules (based on ModSAF 2.1) to the more effective Data Distribution Management procedures developed for recent STOW exercises are discussed. Differences and similarities between this work and ongoing RTI-s and STOW communications network activities are also noted.*

## 1. Introduction

The Synthetic Forces Express (SF Express) project was begun in 1996 with the broad goal of investigating Scalable Parallel Processors (SPPs) as a means of supporting truly large Distributed Interactive Simulations (DIS), and the immediate objective of simulating a militarily realistic scenario with more than 50,000 vehicles. Members of the SF Express team include the California Institute of Technology (Caltech), the Jet Propulsion Laboratory (JPL), and the Naval Research and Development lab (NRaD). ModSAF 2.1 was selected as the initial underlying simulation engine for SF Express.

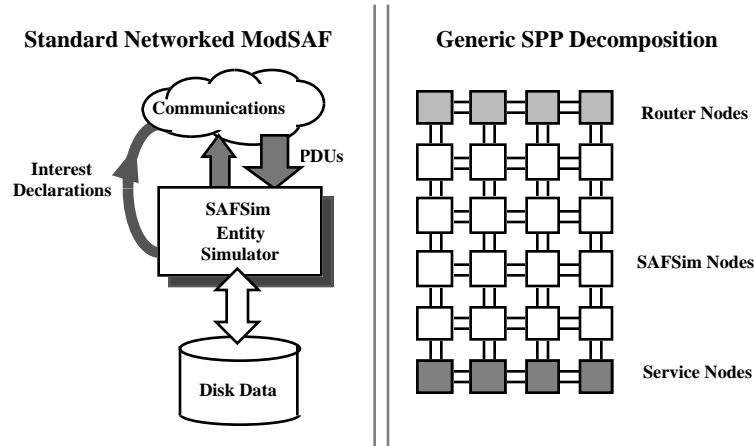
The first major SF Express milestone was achieved in November 1996, with the successful simulation of more than 10,000 vehicles on the Ft. Knox terrain, using the 1024-processor Intel Paragon at Oak Ridge National Laboratory (ORNL). In 1997, the initial single-SPP implementation was refined and extended to include multiple SPPs spanning diverse networks. The 50,000 vehicle objective was first achieved in August 1997 using 1,904 total processors on six SPPs (representing three popular architectures) at sites distributed across seven time zones. The 50K-vehicle milestone was subsequently matched using different collections of SPPs and different networking topologies during live demonstrations at the November 1997 High Performance Networking and Computing Conference (SC97).

This paper describes a scalable communications architecture for large ModSAF simulations on a single SPP. After some general remarks on mapping Distributed Interactive Simulations onto SPPs in section 2, sections 3-5 provide a description of the scalable and portable Router Network Architecture (RNA), including some representative performance results for large-scale runs on a 256-processor IBM SP2. (More thorough performance studies are given in Refs. [1,2].) Sections 6-9 examine various aspects of the RNA model, including discussions of interest state enumeration schemes, some ModSAF-specific porting impediments, and parallels between this work and ongoing STOW and HLA/RTI activities.

Simulations spanning multiple SPPs and incorporating other resources (e.g., displays) can be realized by generalizations of the RNA computational model. Discussions of such an environment and the use of this extended approach to simulate 50K vehicles are contained in a companion article in these proceedings [3].

## 2. DIS and SPPs

A Scalable Parallel Processor (SPP) is a large collection of processing elements (nodes) connected by a fast communications network. SPPs provide an attractive platform for large DIS simulations, due to both the large computational power of the aggregate collection of nodes and the fast



**Figure 1:** Functional components of ModSAF and mapping of these tasks onto processors within an SPP

interprocessor communications within an SPP. The basic strategy used in porting ModSAF to such machines is a heterogeneous assignment of tasks to processors, as illustrated in Fig.(1).

The left side of Fig.(1) is a schematic view of standard ModSAF, highlighting three key components: 1) the SAFSim entity simulator, 2) the communications network that moves messages (PDUs) among the simulators, and 3) the numerous ModSAF data files for terrain, vehicle definitions, and scenarios. The natural implementation of this system onto an SPP divides the nodes into three separate classes associated with these tasks, as shown on the right side of the diagram:

**SAFSim Nodes:** Most of the SPP's processors execute a minimally modified version of the standard SAFSim code.

**Data Service Nodes:** A small number of nodes read and store the simulation data and forward it to the SAFSim nodes through SPP messages. (It is not efficient or practical to have all SAFSim nodes in an SPP independently read the same sets of data files.)

**Router Nodes:** Data movement among the SAFSim nodes is managed by a number of dedicated router nodes. The broadcast or multicast network calls of standard ModSAF are replaced by point-to-point communications directed by this router network.

Neither side of Fig.(1) is scalable without the imposition of additional interest management logic, which limits the number of incoming data for an individual SAFSim (fast communications fabric alone cannot defeat an  $N^2$  algorithmic defect associated with broadcasts). Since interest management is an active research area, it is important that

the SPP implementation not depend on specifics of any one interest management scheme. For purposes of this work, the essential assumptions on the nature of the interest management procedures are as follows:

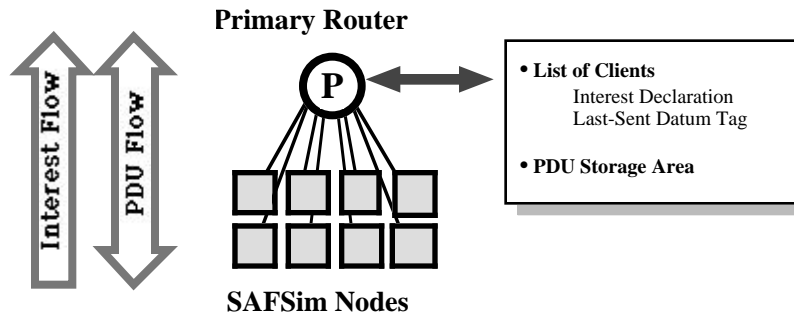
- Each PDU can be associated with an interest value (an "interest class").
- Each SAFSim can compute its own interest state, meaning the set of all relevant interest values for locally simulated vehicles.
- The communications network must deliver to the SAFSim only those PDUs which overlap the SAFSim's declared interest state.

The key aspect of SF Express is the network of router nodes that manage interest-filtered communications of PDUs among the SAFSim nodes. There is considerable freedom in the design of this network, and two different approaches were investigated within the SF Express project:

**Router Network Architecture (RNA):** Routers are associated with fixed collections of SAFSims (and an individual SAFSim communicates with only one router throughout the entire simulation).

**Interest Management Scheme (IMS):** The router nodes in Fig.(1) are associated with (static) sets of interest classes, and an individual SAFSim node communicates with all router nodes that overlap its current interest state.

The IMS scheme is described in Ref. [4]. The discussions below are restricted to the Router Network Architecture approach. As will be seen, there are a number of interesting parallels between RNA and ongoing STOW activities.



**Figure 2:** Schematic of a Primary Router with its associated set of SAFSim nodes

### 3. The Router Network Architecture

The basic building block of Router Network Architecture is a fixed set of SAFSim nodes communicating with single “Primary Router” node, as illustrated in Fig.(2).

There are only two essential modifications to the standard ModSAF code, as run of the SAFSim nodes of Fig.(2):

1. The usual (broadcast) network *reads* and *writes* in the *libpktvalve* communications library are replaced by SPP communications with the router node.
2. Each SAFSim node periodically recomputes its collective interest state (union of interest states for all locally simulated vehicles) and sends this information to its router.

Note that the SAFSim nodes thus operate as though they were attached to a simple Broadcast IP ethernet. In particular, none of the Multicast IP interest management features within *libpktvalve* are needed or used. As is discussed below, this has important implications for the “granularity” of interest management rules.

The Primary Router in Fig.(2) receives and (temporarily) stores PDUs and interest declarations from the attached SAFSims and subsequently forwards those PDUs that match the SAFSim interest states. These tasks are implemented using three straightforward constructs:

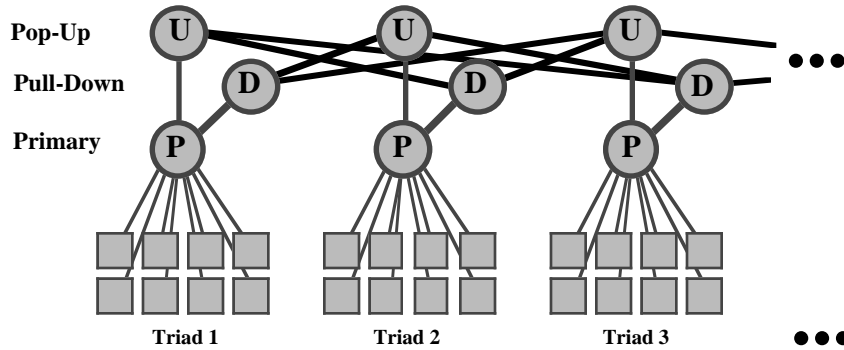
1. A large circular buffer to store active data elements (each of which contains a ModSAF PDU and some additional bookkeeping data).
2. A client list that maintains the current interest declaration of the individual attached SAFSims and pointers to the next outgoing PDU for each client.
3. A simple interest assessment function that compares a PDU to the interest state of the client and returns “yes” if the PDU should be sent to that client.

The details of the client list and interest assessment function depend, of course, on specifics of the interest enumeration rules. The overall RNA code is constructed so that these rules can be changed rather easily as more effective interest management schemes become available (for example, the RTI-s data distribution procedures from Ref. [5]).

The Primary Router in Fig.(2) is a pure data server that waits for and processes requests from the SAFSim clients (e.g., no data are sent from the router to a SAFSim until that SAFSim attempts a data read). For efficiency, the actual data messages exchanged between the SAFSims and Router are bundles of PDUs. A single RNA bundle generally includes PDUs from several different interest classes.

It has been found that a single Primary Router can comfortably manage the communications for a set of client SAFSims in Fig.(2) simulating up to 1K-2K total vehicles (the actual number of vehicles and SAFSims depends on characteristics of the SPP nodes, the nature of the simulated scenarios, ... ). Multiple replicas of the Primary Router Cluster are required once the overall simulation size exceeds this limit. In such cases, the basic unit of Fig.(2) is first augmented by the addition of two new Router nodes (referred to as “Pop-Up” and “Pull Down”). This enhanced routing “triad” is replicated, and additional communications links between Pop-Up and Pull-Down routers are enabled, giving rise to the full router network shown in Fig.(3).

Communications within the full architecture of Fig.(3) are straightforward. In addition to its normal communications with the SAFSim nodes, each Primary Router forwards all SAFSim PDUs to its associated Pop-Up Router and also sends its collective interest state (the union of the SAFSim interest states) to its Pull-Down Router. Each Pull-Down router subsequently collects interest-filtered PDUs from the full Pop-Up layer, and delivers these data to the Primary. Message passing within the router network follows a strict set of hierarchical rules. In particular, all data exchanges are flow-controlled, being initiated by small request packets sent from one node to a router in a higher layer within Fig.(3). This approach is used to prevent both communications deadlocks and the arrival of large unanticipated



**Figure 3:** Schematic illustration of the full Router Network Architecture communication links

messages that could exceed available system buffer space.

The Pop-Up layer in Fig.(3) provides a distributed repository for all active data messages within the simulation (this makes the Pop-Ups a perfect place to attach data loggers for subsequent replays or statistics gathering). Note that the data collection activities of the Pull-Down routers occur in parallel with the Primary $\leftrightarrow$ SAFSim communications. This parallelism helps minimize the additional time delays for PDUs that must travel through the full router network. (In practice, the propagation delays through the upper network layers are typically less than 70 msec, but this value is sensitive to ancillary parameters, such as the time window during which PDU bundles are accumulated.)

#### 4. Portability of RNA

The Router Network Architecture software is written using the MPI (Message Passing Interface) [6] standard library of communications routines for SPPs (both distributed-memory and shared-memory models) to provide a platform-independent implementation of the communications architecture of Fig.(3). RNA currently runs on the Intel Paragon, the IBM SP2, the HP Exemplar, the SGI Origin and Onyx, and the “Beowulf” PC Cluster [7]. (The Beowulf system used in SF Express experiments is built from Pentium-Pro workstations and 100baseT ethernet switches, using freely available operating system software. As such, it provides an ideal opportunity for migration of the RNA approach to actual STOW exercises.)

By far, the hardest part of porting the SF Express functional components to any new SPP came in porting the ModSAF libraries to run on a single processor of that SPP. Compiler, endian, and operating system nuances had to be mastered. Various platforms were found to be more prone to a number of unpleasant, scenario-specific memory management problems deep within the bowels of ModSAF. Passing the standard benchmark test on all platforms was a confidence builder, but by itself did not ensure correct ModSAF behavior for the SF Express collection of

scenarios (featuring high vehicle densities and large local vehicle tables sizes).

#### 5. Scalability of RNA (Theory and Practice)

In a truly scalable implementation of ModSAF on SPPs, larger simulations would be accommodated by simply adding additional triads to the configuration in Fig.(3) with, at most, a  $\log N$  degradation in performance as the number “ $N$ ” of SAFSim nodes is increased. However, in order for this to happen, it is first necessary that the overall problem admits a scalable solution, which effectively means:

**Assumption:** *The number and rate of PDUs of interest to any individual SAFSim must be bounded.*

Without this assumption, the incoming data rates overwhelm the computational capabilities of the SAFSim, independent of any cleverness in the communications model.

This basic requirement is not satisfied by either ModSAF 2.1 or ModSAF 3.0, due to the existence of a number of broadcast classes within the PDU interest enumeration schemes. Approximate patches to this broadcast problem (as used in the large-scale SF Express experiments) are mentioned in section 6.

Assuming, for the sake of argument, the existence of an effective set of interest enumeration rules, the scaling behavior of the router network architecture is fairly easy to analyze:

**Scalability at the Primary Routers:** Given bounded incoming traffic for each SAFSim, the total communications burden at the Primary Router of Fig.(2) is also bounded, with the only questionable link being traffic from the full network of Fig.(3) through the Pull-Down. By the bounded interest assumption, the total traffic rate on this link is also bounded.

Scalability in the Upper Router Layers: In the large problem size limit, only a finite number of Pop-Up Routers will have relevant data for any given Pull-Down (again, by the bounded interest assumption), and the scaling behavior of the communications architecture is determined by the propagation of Pull-Down interest states up to the Pop-Up layer. As sketched in Fig.(3), this interest “broadcast” is not scalable since it is done with  $N_T-1$  separate messages (where  $N_T$  is the number of triads). This problem could be removed by inserting an additional interest-broadcast tree of nodes into the network. However, this solution would incur the usual cost that scaling of the number of communications processes at each node results in a logarithmically increasing total transit time for individual interest declaration messages. For simulation sizes of current interest (e.g., as many as 30 triads on the ORNL Paragon), the time spent in the “non-scaling” interest broadcasts is found to be inconsequential, and the (formally) non-scaling interest broadcast model is found to be adequate.

The preceding general expectations on the scaling behavior of RNA have been verified through extensive runs on a number of different SPP architectures, with up to 18,000 simulated vehicles [1,2]. Due to space limitations, detailed results are presented here for a single set of runs on the IBM SP2 at the Maui High Performance Computing Center (MHPCC).

The scenarios used in these runs were selected from the 60K vehicle “Version 3” scenario set created by the ExInit project [8]. This involves seven separate conflict areas between Blue and Red forces distributed over the Saudi Arabia, Kuwait, Iraq (SAKI) terrain database. The vehicle laydowns within each force group were done according to validated military doctrine. The scenarios feature intense Red↔Blue interactions within 20 minutes of the simulation’s start. The ExInit scenario files were sorted for the scaling runs so that both the local vehicle densities and the level of local Red↔Blue interactions remained approximately constant as the overall problem size increased.

**Table 1:** Router and SAFSim Performance Measures for Runs on the MHPCC IBM SP2

Performance Measure	Run Size (Number of Nodes)		
	81	161	238
Number of Router Triads	3	6	9
Number of SAFSim Nodes	60	120	180
Number of Simulated Vehicles	4,327	8,529	12,915
Primary Busy Fraction	0.188±0.038	0.189±0.018	0.207±0.035
Pop-Up Busy Fraction	0.025±0.015	0.025±0.007	0.027±0.014
Pull-Down Busy Fraction	0.030±0.022	0.026±0.018	0.031±0.016
Primary Receive Time (msec)	0.560±0.115	0.537±0.057	0.587±0.089
SAFSim Comms. Fraction	0.023±0.021	0.024±0.011	0.030±0.037
SAFSim Receive Time (msec)	1.191±2.200	0.978±0.912	1.526±2.652

Table 1 presents a number of results from three scaling runs on the MHPCC SP2. Each Primary Router in these runs controlled 20 SAFSim nodes (intentionally conservative), and each SAFSim simulated approximately 75 vehicles. Due to the fairly intense/localized nature of the scenarios, the remote vehicle tables on the SAFSims were quite large (up to 2,000 remote vehicles, with an average of about 750). A typical SAFSim node sends out 20-40 PDU/sec and receives 100-200 PDU/sec from the communications network (a typical PDU has a size of 200 bytes). Nodes not explicitly counted in the table (e.g., 12 nodes in the 81-node run) were used for either CTDB terrain servers or initialization data servers.

The primary performance measure listed in Table 1 is the router busy fraction, defined by

$$f_{\text{BUSY}} = \frac{\text{Time Spent In Routing Activities}}{\text{Total Wall Clock Time}}$$

The RNA software monitors and reports this statistic periodically for each router (typically once every 30 seconds).

The values listed in Table 1 are measured means and standard deviations for representative routers, averaged over the time interval

$$1000 \text{ s} \leq t - t_{\text{UF}} \leq 2000 \text{ s},$$

where  $t_{\text{UF}}$  (“UnFreeze”) marks the start of the actual simulation.

As expected, the busiest nodes in the Router Network Architecture are the Primary Routers, as they service the data needs of the SAFSims. The sampled values in the table correspond to a total message rate of about 4,000 PDUs/sec (about 1 Mbytes/sec) through each primary router. The Primary Routers in the MHPCC runs handle this communications load easily, spending 80% of their time simply waiting for a new incoming request. (This suggests that the assignment of only 20 SAFSim nodes to each triad was too conservative.) More importantly, the busy fractions for the Primary Routers remains approximately constant as the problem size increases, consistent with the general scaling behavior discussions given above.

The busy fraction values for the upper layer routers in Table 1 are much smaller and are also almost constant with increasing problem size. The first result is expected (e.g., a Pop-Up has far fewer attached clients than a Primary) while the second result is scenario-dependent and a bit better than expected. The more extensive sets of runs discussed in Refs. [1,2] confirm the claims made above that:

1. A “non-scaling” increase in upper-layer router activity is eventually observed, but
2. Even at the largest SPP runs done to date, the upper-level routers are idle more than 75% of the time.

The small non-scaling communications component in RNA is easily accommodated in all analyses done to date. In fact, the upper layer router activity levels are sufficiently low that it is possible and even tempting to combine the functionality of these two routers onto a single processor. This option has not yet been explored, since the resource costs of plausible additional tasks for these routers (e.g., PDU loggers attached to the Pop-Ups) are not yet known.

The “Primary Receive Time” row in Table 1 gives the mean time spent by the router nodes to receive and store a PDU bundle from a SAFSim node. These values are seen to be approximately constant, as expected. Similar behavior is found for runs on other SPP architectures, with hardware-dependent variations in the actual receive time value (e.g., about 0.8 msec/request for the older Intel Paragon architecture and about 1.5 msec/request for the 100baseT communications within a Beowulf).

The final two rows of Table 1 examine communications performance for SAFSim nodes, with the “Comms Time Fraction” row listing the fraction of wall clock time spent in communications with the router network. These values are approximately constant and, more importantly, they are quite small. The fast internal communications fabric within an SPP reduces the communications burden on the SAFSims, and almost all time can be devoted to the actual entity simulations.

A number of other system performance measures are explored in Refs. [1,2] including distributions in the total transit times of PDUs within the full communications network of Fig.(3). The network transit time is clearly an important characterization of the router network. Unfortunately, it is also difficult to assess, since straightforward timing measurements are found to be distorted by occasional delays in read attempts from the SAFSims (sluggish scheduler performance). The results in Ref.[1] suggest that the total time delays associated with the Router Network are less than 70 msec (at about the 95% confidence level).

## 6. Remarks on Broadcast PDUs

As has been noted above, true scalability requires that there be no broadcast PDUs within the system - a requirement that is not strictly satisfied by any existing version of ModSAF. For the ModSAF Version 2.1 code used in the initial SF Express runs, a straightforward variant of the enumeration rules from Ref. [9] was used for all PDUs with natural position values, leaving radio messages and emissions as the primary broadcast traffic. Since ModSAF 2.1 does not support radio filtering through frequency, a simple patch was implemented such that radio PDUs were broadcast within the set of SAFSims associated with a single Primary Router, but not sent from the Primary Router into the upper layers of the full routing network. This strategy appears to give a reasonable level of radio traffic (e.g., about 10 incoming radio PDUs/sec at each SAFSim). This same approach was used for emission PDUs.

This (artificially) reduced broadcast rate was chosen as a conservative safeguard for the initial 50K-vehicle SF Express runs involving multiple SPPs. However, earlier versions of the single-SPP code included numerous true broadcast classes. In one instance, with 16,000 simulated vehicles on the 1024-node Intel Paragon at Oak Ridge National Laboratory, 60% of the total incoming traffic at a typical SAFSim came from broadcast PDUs, while the steady state traffic rate through the Primary Routers was about 20K PDUs/sec/router (about 4Mbytes/sec/router).

The Router Network communications architecture of Fig.(3) can easily accommodate a significant level of broadcast traffic. The real problem with broadcast PDUs comes in processing the messages once they arrive at the SAFSim (the SAFSims in the large ORNL run were completely overwhelmed). Broadcast PDUs are a simulation engine problem long before they are a problem for the Router Network Architecture.

## 7. ModSAF and Truly Large Simulations

The design, implementation, and testing of the Router Network Architecture communications model was a long but fairly straightforward task. Ultimately, many of the major obstacles were due to the underlying ModSAF simulation code itself. In addition to straightforward porting and memory management issues, some of these problems were more interesting and are worth noting.

The first example concerns initialization of the terrain database. The SAFSims in SF Express use cached terrain data, with the cache data maintained in a number of dedicated server nodes. Construction of the terrain “Route Map” during initialization has every SAFSim interrogating every patch of cached data. In order to complete this initialization in a reasonably short amount of time, an unfortunately large number of replicas of the terrain data servers are required (this accounts for most of the difference between total nodes and router+SAFSim nodes in Table

1). The correct solution to this problem would be a parallelization of the Route Map construction code. (Unfortunately, the Route Map appears to be quite a rat's nest of pointers, and parallelization will require some care.)

Another problem encountered with early SF Express runs involved loading the scenario files. The standard ModSAF procedures in *libcreate* are perhaps best described as approximate and almost adiabatic, relying on vehicle migration to balance slowly the computational load across multiple SAFSims. This approach is not reasonable for large-scale runs on SPPs, and the entire scenario loading procedure was rewritten in a purely deterministic and much more efficient fashion.

The combined effects of the re-worked scenario loader and the "redundant" terrain servers on initialization times for large ModSAF runs were quite dramatic. The worst case total initialization time (a run using all 1024 nodes of the ORNL paragon) was only about 25 minutes. On Caltech's 256 processor HP Exemplar, initialization of a 12,000 vehicle simulation can be done in less than 10 minutes.

A final, more fundamental problem has to do with the implementation of ModSAF vehicle tables. It was observed early in this project that scenario files with high vehicle densities often resulted in poor scheduler performance (i.e., a large amount of SAFSim "GASping"). This problem limits the number of vehicles that can be hosted on a SAFSim to values well below expectations based on the standard ModSAF benchmark criteria. Profiles of large SPP runs indicated that simple manipulations of the vehicle tables consumed an inordinately large fraction of the total CPU time. This problem has been investigated in some detail by the members of the SF Express team at JPL, and the reformulation of the position-based vehicle table (*libpbt*) described in Ref. [10] shows promise for improving ModSAF performance in cases of very large local vehicle densities.

## 8. Project Status and Brief Comparisons With STOW

In fewer than 18 months, the SF Express project has evolved to a point where single-SPP ModSAF runs involving more than 10,000 simulated vehicles have become almost routine. The RNA communications model provides efficient PDU exchange among the SAFSim in a manner that is demonstrably scalable, up to the well-understood problems associated with broadcast PDUs. Within the initial context of ModSAF 2.1, the unsolved performance problems are largely related to ModSAF itself (The sluggish manipulations of large vehicle tables noted above is one example. A more interesting issue beyond the scope of the initial SF Express objectives is a fast-moving vehicle with a large field of regard – the so-called "high flier problem.")

During the latter portion of 1997, the Router Network Architecture was extended to include coordinated multi-SPP runs with more than 50,000 simulated vehicles, as is described in a companion paper [3]. Ongoing activities in this area include improvements in multi-SPP program startup through the Globus metacomputing framework [11] and incorporation of conventional SAFStation LANs as players in the large runs (meaning, among other things, rudimentary command and control capabilities through PO directives.)

An obvious next step for the SF Express project is an upgrade of the underlying SAFSim code to a newer version of ModSAF (such as STOW/RTI-s). Since the separation between ModSAF simulation engine and the Router Network code is fairly clean, this upgrade should be largely straightforward. The more interesting parallel design and programming issues will likely center on efficient distributed implementations of the more elaborate terrain database and the enhanced synthetic environment capabilities that have been added since ModSAF 2.1. Longer term projects could include work to adapt RNA as an HLA-compliant federate, most likely using the "big fat federate" model of Ref. [12]. This possibly could be implemented through some variant of the dedicated gateway processors now used for SF Express runs on multiple SPPs [3].

There are interesting similarities between parts of the Router Network Architecture and some recent STOW and HLA/RTI activities. The functionality of the Primary Router in Fig.(2) is similar to that of the custom routing hardware constructed for the STOW 97 communications network[13], in that each approach efficiently sends only interest-screened PDUs to a fixed set of associated client SAFSims. Similarities between the Primary Router and the dedicated routing workstations used in Ref. [14] to implement reliable message delivery within HLA/RTI are even more striking, extending to similar "bail out" contingencies in the event of storage buffer overflows. (It is worth noting that the MPI communications libraries used in SF Express provide guaranteed message delivery.)

The most striking comparison between RNA and STOW/RTI-s is, however, a significant difference related to "interest enumeration capacity." The HLA/RTI-s data distribution management procedures from Ref. [5] continue the use of Multicast IP addresses for interest enumeration, as was originally done in Ref. [9], imposing a rather significant limit (about 3,000) on the number of interest enumeration classes that are available. Due to this limitation, the RTI-s interest enumeration model is forced to use a non-uniform gridding of the full battle field in order to achieve useful spatial resolution in "interesting" regions of the SAKI terrain. In the RNA approach, on the other hand, interest enumeration is done by bit fields in an internal buffer, and the position-based interest grid used in the large runs on the SAKI terrain typically involved more than 40,000 grid cells. This fine position grid is particularly

important in the vicinity of the “motor pool” elements of the SF Express scenarios, with up to 1,200 trucks in a 3 km by 3 km square.

Based on experiences to date with the large entity counts and high vehicle densities of the SF Express runs, it would seem that the much finer interest enumeration capabilities supported by the RNA software are essential for any truly large ModSAF simulation.

## 9. Acknowledgments

Support for this research was provided by the Information Technology Office, DARPA, with contract and technical monitoring via Naval Research and Development Laboratory (NRaD).

Access to well managed and supported supercomputing facilities as well as the unprecedented, large scenario data were instrumental for this research. Special thanks to Betsy Riley (ORNL), Heidi Lorenz-Wirzba (Caltech/CACR), Lyn Gerner and Margaret Lewis (MHPCC) and Adam Whitlock (NOSC).

## 10. References

- [1] S. Brunett and T. Gottschalk, *Pathfinder: A Scalable Implementation of ModSAF on SPPS*, Center for Advanced Computing Research (CACR) technical report, in preparation.
- [2] S. Brunett and T. Gottschalk, *Large-Scale Meta-computing Framework for ModSAF*, Technical Report CACR-152, California Institute of Technology, January 1998.
- [3] S. Brunett and T. Gottschalk, “Scalable ModSAF Simulations with More Than 50,000 Vehicles Using Multiple Scalable Parallel Processors,” 98S-SIW-181, 1998 Spring Simulation Interoperability Workshop.
- [4] L. Craymer and C. Lawson, “A Scalable, RTI-Compatible Interest Manager for Parallel Processors,” 97S-SIW-152, 1997 Spring Simulation Interoperability Workshop.
- [5] S. Rak, M. Salisbury, and R. MacDonald, “HLA/RTI Data Distribution Management in the Synthetic Theater of War,” 97F-SIW-119, 1997 Fall Simulation Interoperability Workshop.
- [6] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, MPI, *The Complete Reference*, MIT Press (1996).
- [7] D. Ridge, D. Becker, P. Merkey and T. Sterling, “Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs,” Proceedings, IEEE Aerospace, 1997. (See also the Beowulf WWW site at <http://cesdis.gsfc.nasa.gov/beowulf>.)
- [8] J. Kohler, S. Narasimhan, J. Pittman and A. Whitlock, “ExGen Software Design Document,” Naval Command, Control and Ocean Surveillance Center (NCCOSC), RDT&E Division, June 1996.
- [9] J. E. Smith, K. L. Russo, and L. C. Schuette, “Prototype Multicast IP Implementation in ModSAF,” Proceedings of the 12th Workshop on Standards for the Interoperability of Distributed Simulations.
- [10] L. Craymer and L. Ekroot, “Characterization and Scalability,” 98S-SIW-190, 1998 Spring Simulation Interoperability Workshop.
- [11] The *documents* link at the Globus Project WWW site (<http://www.globus.org>) has links to a number of technical papers and interface specifications.
- [12] J. E. Smith, “Adapting ModSAF Applications to Comply with the HLA,” Proceedings of the 15th Workshop on Standards for the Interoperability of Distributed Simulations.
- [13] R. Cole and B. Root, “Network Technology for STOW 97,” Naval Research Laboratory briefing and private communication.
- [14] H. Wolfson, S. Boswell, D. Van Hook, and S. McGarry, “Reliable Multicast in the STOW RTI Prototype,” 97S-SIW-119, 1997 Spring Simulation Interoperability Workshop.

## Author Biographies

**SHARON BRUNETT** is a computing analyst for Caltech’s Center for Advanced Computing Research. She received her B.S. in Computer Science and Applied Mathematics from the University of California, Riverside in 1983. Current interests include scalable I/O methodologies for SPPs, characterizations of performance on MTA architectures, and integration of multidisciplinary applications.

**THOMAS GOTTSCHALK** is a Lecturer in Theoretical Physics, Member of the Caltech Professional Staff, and CACR Senior Research Scientist. He received a B.S. in Astrophysics from Michigan State University in 1974 and a Ph.D. in Theoretical Physics from the University of Wisconsin in 1978, and spent several years designing simulation models for High Energy Physics interactions. He has written large parallel codes for multi-target tracking (missile and air defense applications) and for physical design validation of very large VLSI chips. He is active with K-12 outreach projects dealing with computer and Internet usage.