Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments

Sharon Brunett, Dan Davis, Thomas Gottschalk, Paul Messina
Center for Advanced Computing Research
California Institute of Technology
Pasadena, California 91125

Carl Kesselman
University of Southern California
Information Sciences Institute
Marina del Rey, CA 90292

Abstract

A distributed, parallel implementation of the widely-used Modular Semi-Automated Forces (ModSAF) Distributed Interactive Simulation (DIS) is presented, with Scalable Parallel Processors (SPPs) used to simulate more than 50,000 individual vehicles. The single-SPP code is portable and has been used on a variety of different SPP architectures for simulations with up to 15,000 vehicles. A general metacomputing framework for DIS on multiple SPPs is discussed and results are presented for an initial system using explicit Gateway processes to manage communications among the SPPs. These 50K-vehicle simulations utilized 1,904 processors at six sites across seven time zones, including platforms from three manufacturers. Ongoing activities to both simplify and enhance the metacomputing system using Globus are described.

1 The Large-Scale DIS Problem

Over the past few years, Distributed Interactive Simulation (DIS) [1] has become an increasingly essential tool for training, system acquisition, test and evaluation within the Department of Defense. Key components of DIS include: high fidelity computer simulated individual entities (tanks, trucks, aircraft, ...); interactions among entities hosted on different computers through network messages; and support for Human In Loop (HIL) interactions. Using DIS, it is possible to create large scale virtual representations of real operational environments which are inexpensive enough to be used repeatedly.

ModSAF is a particularly important example of DIS which is routinely used for cost-effective training throughout the armed forces. It is generally run using an ensemble of workstations communicating over a network, typically a LAN. Each workstation is responsible for simulating some modest number (30–100) of entities. These computer generated Semi-Automated Forces (SAF) are intended to realistically mimic the behaviors of opposing or support forces within an exercise. The entity and environment models are accordingly quite detailed.

Individual simulators (workstations) interact through the exchange of data messages called PDUs (Protocol Data Units)[2]. These PDUs are used in ModSAF to describe the state of individual entities, weapons firing, detonations, environmental phenomenon, command and control orders, etc. In standard ModSAF, the PDUs are sent as UDP datagrams. Due to this unreliable message delivery mechanism, each entity state PDU typically contains a complete summary of the vehicle's current state, and PDUs are (re)transmitted at a frequent, regular 'heartbeat' intervals to compensate for dropped data packets.

Independent of the nature of the PDU communications mechanism, this simplest picture of ModSAF is not scalable in that it (implicitly) assumes each simulator receives and responds to all PDUs from all other simulators—a model which clearly fails as the number of simulators and simulated entities increases. Moreover, in many realistic large scale simulations, it is invariably the case that most system-wide PDU traffic is irrelevant for the limited set of entities hosted on an individual simulator (e.g., tanks separated by tens of kilometers generally don't interact).

The DIS community ran into these issues in their STOW-E exercise (Synthetic Theater of War-Europe [3]) and ED-1A Engineering Demonstration [4], where ModSAF was used to simulate 5,371 vehicles hosted at twelve separate sites in the USA and in Europe.

Increasing the simulated entity count could not be achieved by simply adding more workstations to the network. Addition of a PDU screening mechanism ('Interest Management') helped but did not eliminate all scaling hurdles.

This paper describes a new approach to truly large-scale DIS, using multiple Scalable Parallel Processors (SPPs) to solve the scaling problems observed in STOW-E. After a short summary of project goals and accomplishments in Sections 1.1 and 1.2, Section 2 presents the general method used to port ModSAF to run on an SPP. Sections 3-5 contain, respectively, a (long term) vision for an effective STOW metacomputing model, an analysis of initial multi-SPP ModSAF accomplishments, and an overview of ongoing activities to enhance and extend the existing software using elements from the Globus metacomputing toolkit [7][8].

1.1 The Synthetic Forces Express Project

The Synthetic Forces Express project (SF Express) [9] began in 1996 to explore the utility of Scalable Parallel Processors (SPPs) as a solution to the communications bottlenecks of conventional ModSAF. The SF Express team consists of personnel from the California Institute of Technology (Caltech), the Jet Propulsion Laboratory (JPL), and the Space and Naval Warfare Systems Center San Diego (SPAWARSYSCEN, formerly known as NRaD). The SF Express charter was to demonstrate a scalable communications architecture simulating 50K vehicles on multiple SPPs—an order of magnitude increase over the size the STOW-E simulation.

SPPs provide a natural, attractive alternative to networked workstations for large-scale ModSAF runs. Most of the processors on an SPP can be devoted to independent executions of 'SAFSim', the basic ModSAF simulator code. The reliable high-speed communications fabric between processors on an SPP provides significantly increased bandwidth over standard dataflows among networked workstations. Construction of a scalable communications scheme was done in three main steps:

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

Intra-SPP Communications: Within an individual SPP, a selectable number of processes were designated as message routers. These processes receive and store interest declarations from the simulator nodes and move simulation data packets according to the interest declarations.

Inter-SPP Communications: Additional interest-restricted data exchange procedures were developed to support SF Express execution across multiple SPPs.

The primary technical challenge in porting ModSAF to run efficiently on SPPs comes in the construction of a suitable network of message passing router nodes/processors. SF Express uses point-to-point SPP communications (implemented using the MPI Message Passing Interface [10]) to replace the UDP socket calls of standard ModSAF. The network of routers manages these messages, effecting interest-restricted communications among simulator nodes. There is considerable freedom available in the construction of the router node network. This paper describes a model based on fixed communication channels among specific subsets of processors within an SPP. This Router Network Architecture (RNA) was developed at Caltech [11],[12].

As the simulation problem size increases beyond the capabilities of any single SPP, additional interest-restricted communications procedures are needed to enable 'Metacomputed ModSAF' runs on multiple SPPs. A number of options were considered, and the first implementation used dedicated Gateway processors to manage inter-SPP communications.

1.2 50K+ Simulated Vehicles

On 11 August 1997, the SF Express project performed two separate simulation runs, each with more than 50,000 individually simulated vehicles. The runs used three different types of Scalable Parallel Processors (SPPs) at six separate sites spanning seven time zones, as shown in Fig.(1). These sites were linked by a variety of wide area networks. Specifics for each site are listed in Table 1. The majority of the SPPs used the RNA communications scheme, while NASA Ames and CEWES used an alternative approach developed at JPL [13].

The $N(P)$ entries in the table give the number of processors used at each site. The $N(V)_j$ columns indicate the number of locally simulated vehicles in each of the two runs. The 50K-vehicle simulation scenarios were created by the ExInit software [14] and featured immediate intense interactions among the simulated entities, causing high communications levels both within and among SPPs.

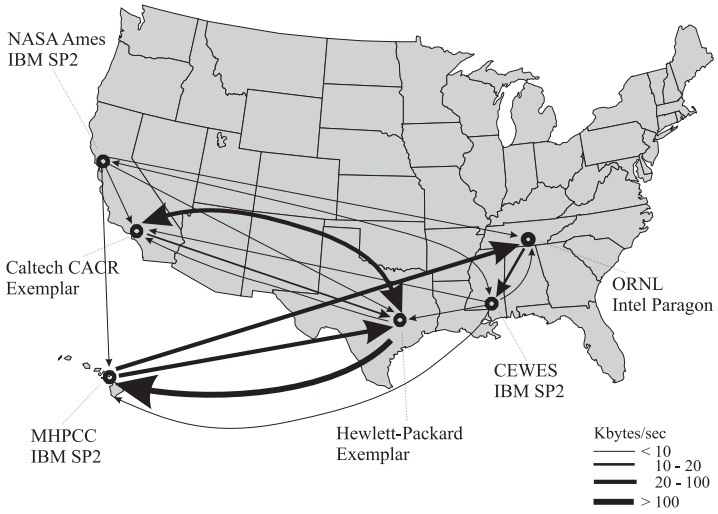

Figure 1: SPP Sites and Message Rates in the 50K SF Express Runs

Table 1: Participating Sites and Simulated Entity Counts for the 50,000 Vehicle SF Express Runs.

Site	Hardware	N(P)	$N(V)_1$	$N(V)_2$
Caltech, Pasadena CA	HP Exemplar	256	13,095	12,182
ORNL, Oak Ridge TN	Intel Paragon	1024	16,695	15,996
NASA Ames CA	IBM SP2	139	5,464	5,637
CEWES, Vicksburg MS	IBM SP2	229	9,739	9,607
MHPCC, Maui HI	IBM SP2	128	5,056	7,027
HP/Convex, Richardson TX	HP Exemplar	128	5,348	6,733
Total		1,904	55,397	57,182

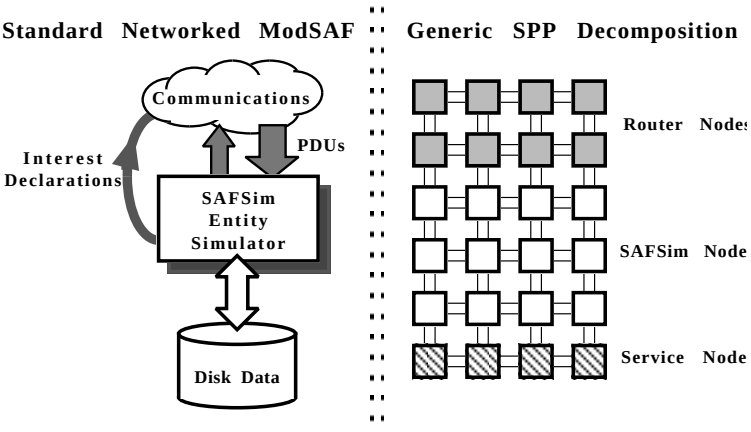

Figure 2: Schematic illustration of a networked ModSAF simulator and notional mapping of the simulator tasks onto an SPP.

2 Porting ModSAF to a Scalable Parallel Processor

The basic strategy used in porting ModSAF to an SPP is a heterogeneous assignment of tasks to processors, as illustrated in Fig.(2). The processors are divided into three classes:

Entity Simulators/SAFSims: Most of the SPP's processors execute a minimally modified version of 'SAFSim', the standard simulator code.

Simulation/Data Servers: A small number of nodes read and store the simulation data and forward it to the SAFSim nodes through SPP messages.

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

Neither side of Fig.(2) is scalable without the imposition of additional interest management logic, which limits the number of incoming data for an individual SAFSim. 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. RNA makes only two minimal assumptions in this regard: each PDU can be associated with an interest value (an "interest class"), and each SAFSim can compute its own interest state (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.

2.1 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.(3). There are only two essential modifications to the standard ModSAF code, as run of the SAFSim nodes of Fig.(3):

1. The usual (broadcast) network reads and writes in the ModSAF network 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.

The Primary Router in Fig.(3) 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.

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 determines whether a PDU matches a client's declared interest.

The Primary Router in Fig.(3) is a pure data server that waits for and processes requests from the SAFSim clients. For efficiency, the actual data messages exchanged between the SAFSims and Router are bundles of PDUs.

It has been found that a single Primary Router can comfortably manage the communications for a set of client SAFSims in Fig.(3) simulating 1K-2K total vehicles. 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.(3) 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(4).

Communications within the full architecture of Fig(4) 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(4). This approach is used to prevent both communications deadlocks and the arrival of large unanticipated messages that could exceed available system buffer space.

The Pop-Up layer in Fig(4) provides a distributed repository for all active data messages within the sim-

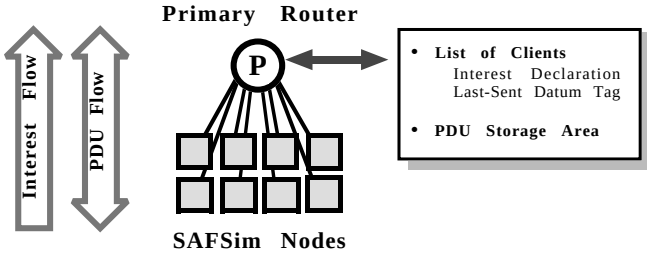

Figure 3: A Primary Router with its associated SAFSims.

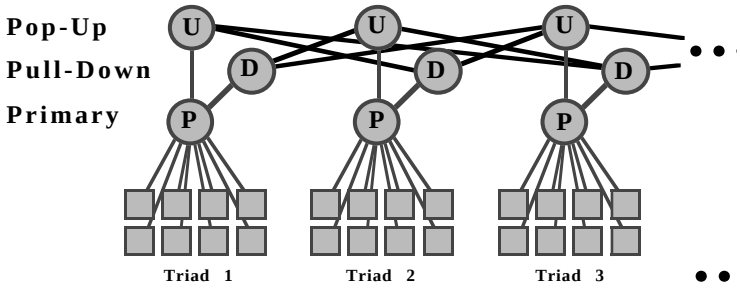

Figure 4: The full RNA router network

ulation (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 SAFSim communications. This parallelism minimizes the additional time delays for PDUs that must travel through the full router network.

2.2 Performance of the Single-SPP ModSAF Implementation

Detailed studies of the RNA model are contained in Refs.[11],[12]. Highlights of these analyses are as follows:

1. The RNA approach has been run successfully on a variety of SPP architectures, including the Intel Paragon, the IBM SP2, the HP Exemplar, the Silicon Graphics Origin 2000 and the "Beowulf" PC Cluster SPP [15].

2. These single-SPP runs have included simulations with up to 18,000 vehicles.

3. The scaling behavior of RNA with increasing problem size is well-understood, with 'theoretical' expectations validated by the measured performance results.

4. The effective inter-processor communications within an SPP lead to a significant reduction in PDU communication overhead for an individual SAFSim (relative to standard ModSAF performance on a LAN/WAN network).

3 Anatomy of a DIS Metacomputer

"Metacomputing" can be defined as the concurrent use of a multiple network-linked resources for solving very large computational problems. However, computing in networked environments has both advantages and drawbacks. The state and structure of networked resources are often dynamic and quite heterogeneous. Performance and portability can be compromised when trying to deal with either heterogeneity. Alternatively, linking large numbers of diverse resources allows access to processing power and unique capabilities beyond the resources at any one site. It also enables applications to be solved with a mix of systems, assigning appropriate and available assets to specific parts of the overall problem.

For many classes of large distributed applications, the aggregate computational power in a collection of SPPs is only part of the metacomputing solution. A full system should link computational engines, storage systems, scientific instruments, advanced display de-

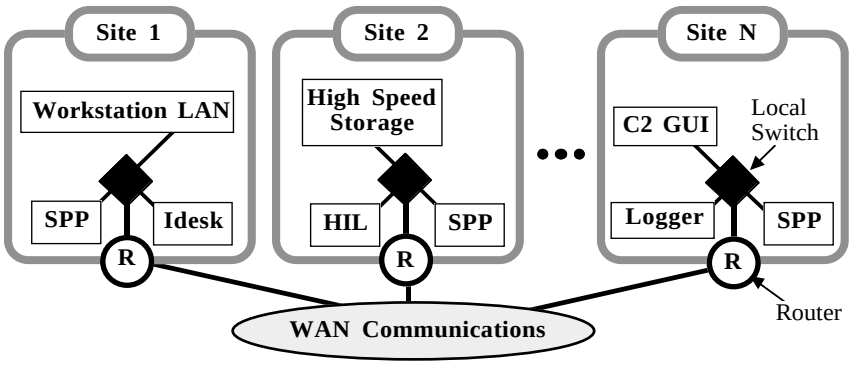

Figure 5: Schematic of a DIS Metacomputing Environment

vices, and human resources, as illustrated in Fig.(5), with "HIL" representing some sort of 'Human In Loop' interface and "Idesk" ("Immersa-desk") representing a typical advanced display device. Data could be gathered from a remote source (for example, a satellite downlink) and streamed into a collection of SPPs for real-time simulation processing. During the course of the simulation, mechanisms for logging, filtering or compressing data could be employed for subsequent post-processing (e.g., visualization, querying and persistent storage).

Distributed heterogeneous computing immediately implies diversity in terms of hardware architectures and performance, operating systems, administrative domains, network protocols, and so on. As the size and complexity of the distributed system increases, operational issues (resource scheduling, allocation, data staging, etc.) become increasingly important components of the metacomputing model.

The next two sections describe two initial steps towards the seamless metacomputing picture of Fig.(5). Section 4 presents the Gateway model used in the initial 50K-vehicle runs in Table 1. Section 5 describes subsequent multi-SPP experiments to integrate parts of the Globus metacomputing toolkit [7][8] in order to remove many operational difficulties encountered in the initial large simulations.

4 SF Express on Multiple SPPs using Explicit Gateways

For large runs on multiple SPPs, some portions of the entity state information from each SPP will, in general, be relevant for entities simulated on other SPPs. Extensions of the single-SPP architecture must effect interest-restricted PDU exchanges among the SPPs. Dedicated Gateway processors provide a straightforward mechanism for this task.

The Gateway processors are generalizations of the intra-SPP routers from Section 2.1, and can be viewed as communications servers for two distinct classes of clients:

Local Clients: Router nodes on the same SPP as the Gateway that hold the continually changing collective PDU and Interest State of the local SPP. These local clients send (internal) interest declarations and simulation data to the Gateway for subsequent delivery to remote resources.

External Clients: Processes on remote machines that receive and process interest declarations and PDU bundles from the local SPP. An external client could be a standard ModSAF workstation or GUI. For inter-SPP links, an External Client is essentially a mirror image of a Local Client which resides on the external SPP.

Gateways manage interest-selected data flow in two directions by way of four basic operations:

1. The collective interest state of the Local SPP is sent out to each of the external SPPs.

2. The corresponding interest declarations are received from the remote SPPs, defining standard client interests. The union of these external interest states defines the collective (external) gateway interest, which is sent up to the local attached routers.

3. The Gateway receives interest screened data from the local routers in the usual manner, and forwards these to the appropriate external hosts.

4. The Gateway receives data from the external SPPs and sends it the attached local routers for subsequent distribution within the local SPP.

Aside from the fact that a Gateway node has two important global interest states (the attached SPP and the external world), the overall operation of Gateways is extremely similar to that of the router nodes from Section 2.1.

4.1 Gateway Specifics for the Initial 50K-Vehicle Runs

The first metacomputing experiments within the SF Express project involved a number of simplifying assumptions and restrictions on the nature of the Gateway processes. In particular,

1. The communications network among the participating SPPs is implemented as a fully-connected set of links between pairs of SPPs, with each SPP dedicating a Gateway processor for each external SPP.

2. Messages between SPPs are sent as UDP/IP datagrams.

3. Interest declaration messages are retransmitted at regular intervals ("heartbeats"), to accommodate the unreliable nature of the UDP messages.

A schematic diagram of the multi-SPP environment is given in Fig.(6), illustrating the dedicated Gateway links.

The Gateways in Fig.(6) operate as pure communications servers, whose task it is to manage the flow of requested PDUs and interest states between SPPs. Details can be found in Ref. [12]. Timing results for Gateway operations in the 50K-vehicle runs are examined in Section 4.4.

The complete connectivity among Gateways in SF Express (as in Fig(6)) should be viewed as a provisional expediency on the road to a 50K-vehicle simulation. With one exception noted below, this model easily handled the inter-SPP traffic, at rates up to 1,000 PDUs/sec. However, this initial model does not scale well as the number of sites in Fig.(6) increases, and has the additional defect that Gateway processors associated with low activity links are a wasted resource. Movement towards an architecture linking individual SPPs by some form of multicasting (possibly ATM) network should be explored.

4.2 The 50K-Vehicle Scenarios

The scenarios used by SF Express involve Blue and Red forces laid down on the 300 km by 350 km SAKI (Saudi Arabia, Kuwait, Iraq) terrain database. The full complement of vehicles is organized into a number of opposing force groups. The relative populations of vehicles types (tanks, trucks, helicopters, ...) and the actual laydowns of units and vehicles were designed according to standard military doctrine [16] including, for example, a roughly 2:1 superiority in numbers for the attacking Blue forces.

Figure(7) presents a schematic of the force deployments in one of the two scenarios used in the 50K-vehicle runs. This 'Version 2.1' laydown has about 42K Blue Vehicles and 21K Red Vehicles. Most of the vehicles (about 85%) are trucks, as is realistic for many actual military campaigns.

The large boxed areas in Fig.(7) show the assignments of scenario elements to SPP platforms. The evolution of the scenario over time is fairly simple: all Blue forces move east and attack while the Red forces sit and defend. This gives rise to intense interactions along the dashed 'Front Line' in Fig.(7). For the given Force⇔SPP assignments, this yields significant data exchanges between the Ames and CEWES SP2's and among the four 64-processor components of the Caltech HP/Exemplar. Additional non-fighting interactions occur between some sets of adjacent Blue force groups.

4.3 Porting and Practical Issues

ModSAF was initially ported to the Intel Paragons at Caltech. Extensive single-node runs were required to understand and optimize the very large ModSAF code base, and to identify and assess the key communications libraries. Numerous problems were encountered (system call assumptions, inadequate bounds checking, ...). Solutions developed during the single-node Paragon work simplified subsequent ports to other platforms, although OS-specific assumptions, awkward build procedures, and occasional cross-compilation issues required case-by-case treatments.

Once the RNA code had matured to the point that simulations with 1K-10K vehicles were becoming routine, initial heterogeneous multi-SPP tests were begun. Coordination and synchronization of the simulation start was quickly identified as a key issue, along with management of the extensive scenario data files. A number of moderate runs, with 20K-30K simulated vehicles at two or three sites, were critical first steps.

The large, 50K-vehicle runs with six SPPs spread across the country involved substantial coordination issues. Various sites had different disk policies, accounting mechanisms, usage models and schedulers. Ultimately, the success of the large runs resulted from moderate to significant system administration interventions, competent system support personnel, and numerous phone calls. While this was acceptable for a demonstration, it is clearly inadequate for a production model. Many of the initial Globus activities

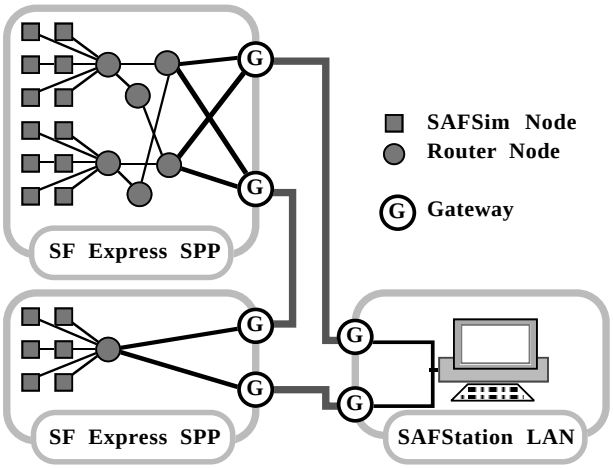

Figure 6: Schematic multi-SPP SF Express using explicit Gateways.

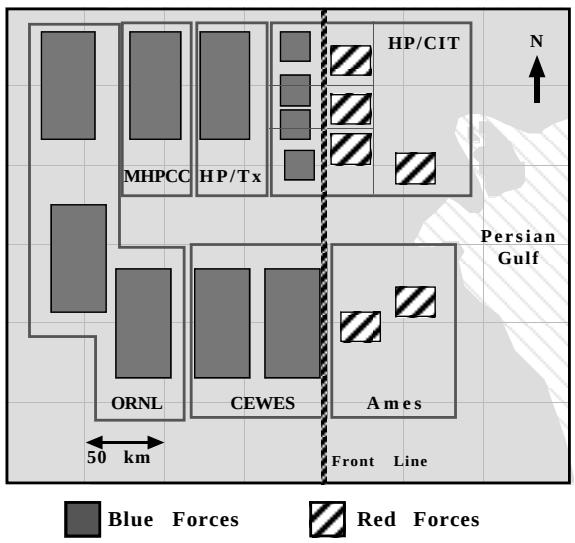

Figure 7: The Version 2.1 Scenario and assignments of force groups to specific SPPs

described in Section 5 focus on these operational issues.

4.4 Inter-SPP Highlights Of The 50K Runs

The performance issues for the metacomputing model of Fig.(6) center on data movement through the Gateway nodes. The results presented in this section will show that communications levels were easily managed with one, essentially expected Paragon exception.

A word on the configuration of the HP Exemplar machines is in order here. The Caltech machine was available only as four independent 64-processor machines (labelled 'HP/Cj' below) when the 50K runs were done (it is now a single 256-processor system). In contrast, the 128-processor Exemplar at the Convex site ('HP/Tx') was configured as a single system image.

4.4.1 Results From The Version 2.1 Scenarios

Table 2 summarizes inter-SPP data rates for the V2.1 scenario run. The rows and columns are labelled by SPP site, and entries are in Kbytes/sec. Links with less than 0.5 Kbytes/sec are left blank.

Many of the RNA⇔RNA communication links have no appreciable activity. This is due to the geographic separation of the Force groups in Fig.(7) and additional restrictions on broadcast PDUs, as discussed in Refs.[11],[12]. The communication model running on Ames and CEWES retains a significant level of simulation-wide broadcast PDUs, giving rise to the constant "background" data rates evident in the bottom two rows of Table 2. The values in Table 2 give the rates at which data are sent from the 'Row SPP' to the 'Column SPP'. Due to dropped packets, these are not the same as the rate at which data are received by the Column SPPs, but they are generally close. The exceptions involved links to ORNL, where packet loss was often severe. In the worst case,

MHPCC	Sends	63.9 Kbytes/sec To ORNL
ORNL	Receives	7.9 Kbytes/sec From MHPCC

With the exception of communications to ORNL, the number of dropped UDP packets within the Version 2.1 runs is small, and well within the tolerable range for ModSAF.

Table 3 contains a detailed look at three of the more active inter-SPP links from Table 2:

HP/Tx ⇔ MHPCC: Successful, moderately high bandwidth communications between machines on a Wide Area Network.

MHPCC ⇔ ORNL: Saturated/Failed communications between machines on a Wide Area Network.

HP/C1 ⇔ HP/C2: Successful communications between machines on a Local Area Network.

The 'PDU Busy' rows list the fraction of (wall clock) time spent in PDU communications within the SPP and through the Gateway to the remote SPP. The last two rows give the mean times for PDU bundle communications across the network. The UDP-ethernet reads and writes on the ORNL Paragon are about 30 times slower than on the other platforms, leading to an overwhelmed Gateway and the significant data losses noted above.

It should be stressed that no attempts were made to optimize network communications in these initial 50K runs. Networks used included ESnet, LosNettos, NREN, DREN, ANSnet, and commodity providers. Fig.(8) shows a partial network map of communications links to the Caltech site, with shaded ellipses for the various network domains. A message from Caltech to MHPCC visits 14 routers, while a return message travels through 12. Clearly, the SF Express 50K-vehicle runs did not use an overly optimized network.

The results in Tables 2,3 indicate that something more aggressive than simple UDP/IP ethernet will be needed to successfully use the ORNL Paragon in a large scale, distributed simulation. As was noted in Section 4.1, the RNA Gateway strategy can accommodate various transport mechanisms.

5 An Integrated Metacomputing Environment Using Globus

The Globus Project [7][8] is developing a basic software infrastructure to support applications which need and/or are capable of using geographically distributed computational and information resources. A key element of Globus is the design and implementation of a distributed supercomputing infrastructure toolkit that provides an integrated set of services in five key areas:

1. **Communications:** The Nexus communications library provides message delivery services for a variety of communications models in a manner that is cognizant of network quality of service parameters.

2. **Information:** The Metacomputing Directory Service (MDS) provides a uniform method for obtaining real-time information on system status and structure.

Table 2: Inter-SPP communications rates for the V2.1 scenario large-scale metacomputing runs.

	HP/C0	HP/C1	HP/C2	HP/C3	ORNL	MHPCC	HP/Tx
HP/C0	-	20.7					35.1
HP/C1	23.9	-	15.9	14.8			16.1
HP/C2		64.2	-	15.3			
HP/C3		18.4	5.8	-			
ORNL					-	63.9	
MHPCC					22.2	-	67.2
HP/Tx	3.7	21.0				169.0	-
AMES	3.3	3.3	3.3	3.3	3.3	3.3	3.3
CEWES	6.3	6.3	6.3	6.3	17.6	6.3	6.2

Table 3: Details of Gateway performance on three busy links of the V2.1 SF Express run.

Local SPP Remote SPP	HP/Tx MHPCC	MHPCC HP/Tx	ORNL MHPCC	MHPCC ORNL	HP/C1 HP/C2	HP/C2 HP/C1
Local PDU Busy	0.168	0.074	0.041	0.050	0.023	0.014
Remote PDU Busy	0.147	0.080	0.926	0.032	0.031	0.030
Read Time [msec]	0.60	0.63	22.26	0.77	0.61	0.60
Write Time [msec]	0.97	0.28	27.52	0.26	0.45	0.34

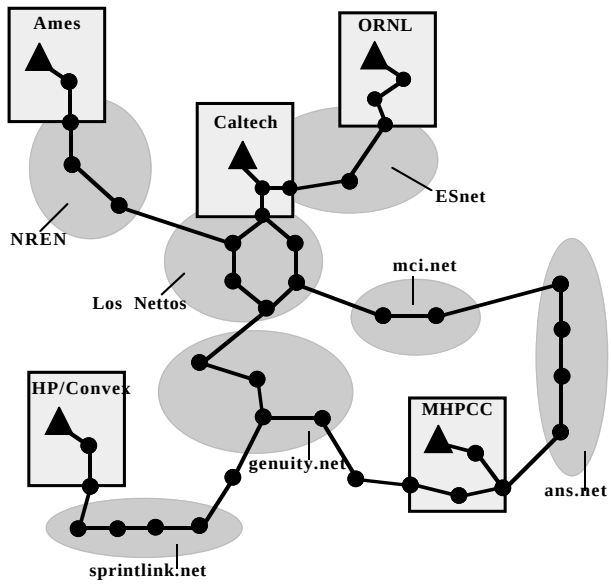

Figure 8: Partial network connectivity map for the 50K-vehicle simulations

3. **Resource Location/Allocation:** The Global Resource Allocation Manager (GRAM) provides mechanisms for declaring application resource requirements, identifying and scheduling appropriate resources, initiating and managing the application on these resources (a low-level scheduler API).

4. **Security:** The Globus system provides a number of basic security services (e.g., authentication and authorization), enabling sophisticated application specific security mechanisms and single sign-on functionality.

5. **Data Access:** Mechanisms are provided for high-speed remote access to persistent storage.

5.1 Benefits of the Globus Toolkit

The modules within the Globus toolkit directly address a number of problems uncovered during the initial, manually operated SF Express metacomputing runs.

The Nexus library provides a "resource aware" implementation of communication tasks (e.g., data exchanges between the Gateway nodes of Fig(6)), using the best available communications mechanism (UDP over IP, HiPPI, ATM, etc.). Simple automatic selection rules or user-guided directives determine the appropriate communications method, with selections dynamically dependent on the status of the available network services. These features are particularly useful for the communications links between Gateway processors, in order to avoid bandwidth saturation, as was observed in Section 3 for the ORNL⇔MHPCC link. The communications layer provides efficient implementations of native communication methods, including message passing, multicast, distributed shared memory, remote procedure calls, etc. The selected method must be aware of Quality of Service (QoS) parameters, like reliability, bandwidth and latency. Intelligent, performance based, application configuration choices can be made to match the "currently available" execution environment. This enables the user to better utilize shared resources, and attain higher throughput.

The MDS and GRAM elements of the Globus toolkit address the broad problem of resource identification, allocation, and task execution within the grid of available assets. The MDS provides an automated, "information-rich" approach to system configuration, enabling intelligent automated resource allocations. MDS includes a data model to represent dynamically changing capabilities of various parallel computers and networks, so that tools and applications do not have to rely on stale or programmer-supplied knowledge (e.g., use a dedicated HiPPI or ATM node on an Intel Paragon instead of garden variety UDP/IP).

Once the desired distributed assets have been identified, GRAM provides a simple, uniform interface to local resource allocations. In essence, GRAM enables the coordinated startup of a metacomputing run by a single "go" script that drives the participating SPPs, attached displays, etc. This represents a significant improvement over the existing environment, in which the non-static differences in operating systems and resource schedulers on various platforms are coordinated by hand-crafted scripts (and prayers) based on detailed knowledge of resource-specific usage models. Globus services also provides periodic health and status information for each job instantiation and allows application-specific tools to hook into generic health and status monitor services. This capability would be an improvement over the existing SF Express method using separate monitoring tools on each SPP.

Not all start-up and job management concerns are addressed with the use of a single script which starts program execution on all resources. Determining and staging required datasets is another concern. Staging of data automatically and efficiently just prior to simulation time avoids a number of headaches associated with site-specific disk usage policies. For example, the 50K scenario datasets could not permanently reside on the file systems of the SPPs used in the SF Express runs, due to various quota limits on assorted high-speed file systems. This situation necessitated tedious (and somewhat error prone) hand staging prior to the large runs.

Simulations to date have involved static assignments of scenarios to SPPs, such that configuration file preparation and data staging could occur prior to SPP resource allocation. This approach typically wastes disk space and does not allow the application to take best advantage of the resources available. The Data Access services (remote I/O calls) within Globus allow high-speed remote access to persistent storage, such as simulation scenarios and behavior files, potentially saving vast amounts of disk space and frequent user-intervention required to move large data sets both before and after runs (possibly scheduled arbitrarily). The more resource-aware an application can become, the larger the window for adaptive and optimal choices.

5.2 Initial Experiments With Globus

The coordinated start-up capabilities of Globus were successfully tested during two live demonstrations at the November 1997 High Performance Networking and Computing Conference (SC97) in San Jose. These experiments involved 824 processors on

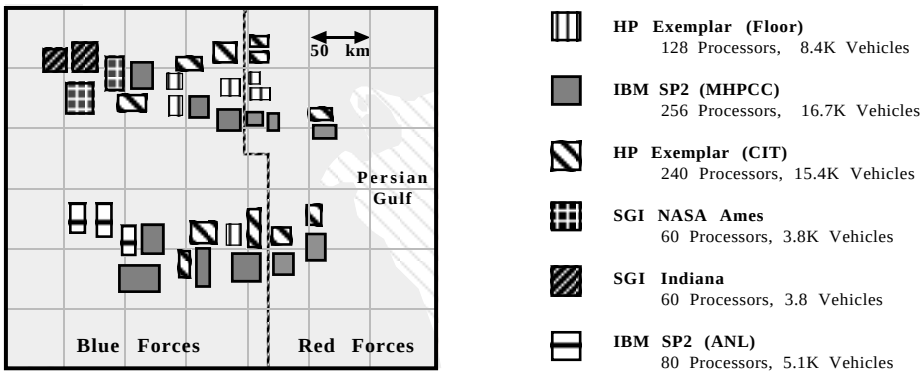

Figure 9: The Version 2.1 Scenario element assignments for the initial tests with Globus.

SPPs at six sites, as shown in Fig.(9), simultaneously displaying parts of the simulation on an Immersa-desk in the Argonne National Laboratory booth on the conference floor. Unlike the fairly conservative force group assignments of the initial 50K-vehicle simulations, these runs involved a more "interleaved" assignment of scenario files to SPPs, as shown in Fig.(9). This was done in order to provide more stressing tests of inter-SPP communications. The overall simulation involved about 40K vehicles (about 50K ModSAF entities). The important new aspects using GRAM specifications to drive the simulation were successfully demonstrated.

6 Accomplishments and Future Directions

The multi-SPP runs in August 1997 surpassed the project goal of a 50,000 vehicle simulation on a heterogeneous collection of SPPs and validated the overall SF Express concept. The ExInit team generated a collection of sound military scenarios, featuring intense, quick interactions (and fighting) within the one-hour time frame of the runs.

Problem areas in the single-SPP SF Express code seemed to center, not surprisingly, on the ModSAF simulation engine itself. Of the hundreds of thousands of lines of ModSAF source code, less than 5 percent of libraries were modified to accommodate RNA. The core simulation engine was purposely left alone, due not only to project scope, but to decouple performance of the communications architecture from the driving simulation engine. Among other issues, simple profiling determined that ModSAF vehicle table manipulations consumed a substantial fraction of total CPU time. Possible solutions for expensive ordered list operations are noted in Ref.[17].

Problems in the multi-SPP runs of Sec.4 were largely operational, arising from the differing environments at the six SPP sites. The Globus experiments described in Section 5 can be viewed as the first steps towards a more user-friendly robust system.

An attractive near-term direction involves a greater exploitation of the unified resource information services, resource location and allocation services, and data access modules within Globus to eliminate much of the configuration file mechanisms within SF Express and optimize runtime parameters. Using the currently deployed Globus services, initialization and execution of a large simulation would proceed roughly as follows:

1. The user specifies the location of the simulation data and the desired simulation size from a single place (e.g., console or file).

2. MDS evaluates the request and locates appropriate resources (with the MDS databases augmented to understand information on the inherent simulation capabilities of the individual platforms).

3. Once the appropriate computational assets are allocated, GRAM is used to start the distributed simulation and to exchange run-time system configuration information among the participants.

4. Using the system configuration information from GRAM, each SPP takes responsibility for a specific subset of the simulation scenario files, retrieving these data automatically from the staging area using the Globus Data Access services.

In this model, user input is largely restricted to the high-level specification of the problem itself (i.e.,

the simulation scenarios), and Globus manages all pragmatic issues of resource allocation, data staging, job management, and network connectivity needed in order to meet the user specified requirements (which could well include additional constraints, such as required network bandwidths).

The construction of this Globus-directed metacomputing model is a realistic near-term goal. Modifications within the existing RNA codebase of Ref.[12] would largely involve generalizations of the Gateway communications procedures to use portable Nexus routines in place of socket calls. Additional new logic would be needed within the single-SPP initialization sequence to support run-time assignments of scenarios to SPPs, based on configuration data from GRAM. (Neither of these tasks is seen as being particularly difficult.) This system would become the next-generation SF Express proof-of-concept demonstration, with intelligent resource allocation, simulation startup and data management all done in a simple, user-friendly manner.

Acknowledgements

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 various computational facilities and significant system support were essential for this work. The 1024 node Intel Paragon was made available by the Oak Ridge Center for Computational Sciences. The smaller Intel Paragon and 256 processor HP Exemplar were made available by Caltech/CACR. The IBM SP2s were provided by the Maui High Performance Computing Center, U.S. Army Corps of Engineers Waterways Experiment Station Information Technology Laboratory, and the Numerical Aerodynamic Simulation Systems Division at NASA Ames Research Center. Indiana University and NASA Ames provided access to the Silicon Graphics machines. A 128 CPU HP Exemplar was provided by HP/Convex Division Headquarters. We thank the system administrators and support staff at all these sites.

Globus experiments and integration were made possible by the dedicated team at Argonne National Laboratory and USC Information Science Institute, including Ian Foster, Karl Czajkowski, Mei-Hui Sue, and Marcus Thiebaux.

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.

Dan Davis is Assistant Director, CACR, has a B.A. in Psychology and a J.D., both from the University of Colorado. With a background in Naval cryptology and intelligence, he and Dr. Gottschalk are also pursuing technology for K-12 education.

Thomas Gottschalk is a CACR Senior Research Scientist, Member of the Caltech Professional Staff, and Lecturer in Theoretical Physics. 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 and for physical design validation of VLSI chips.

Paul Messina is Assistant Vice President for Scientific Computing at Caltech, Faculty Associate in Scientific Computing, Director of Caltech's Center for Advanced Computing Research, and serves on the executive committee of the Center for Research on Parallel Computation. His recent interests focus on advanced computer architectures, especially their application to large-scale computations in science and engineering. He also is interested in high-speed networks and computer performance evaluation. He heads the Scalable I/O Initiative, a multi-institution, multi-agency project aimed at making I/O scalable for high-performance computing environments. Messina has a joint appointment at the Jet Propulsion Laboratory as Manager of High-Performance Computing. Messina received his PhD in mathematics in 1972 and his MS in applied mathematics in 1967, both from the University of Cincinnati, and his BA in mathematics in 1965 from the College of Wooster.

Carl Kesselman is a Project Leader at the Information Sciences Institute and a Research Associate Professor in Computer Science at the University of Southern California. He received a Ph.D. in Computer Science at the University of California at Los Angeles. He co-leads the Globus project, along with Ian Foster, at Argonne National Laboratory. Dr. Kesselman's research interests include high-performance distributed computing, parallel computing and parallel programming languages.

References

[1] J. S. Dahmann and D. C. Wood, "Scanning the Special Issue on Distributed Interactive Simu-

lations", *Proceedings of the IEEE*, Volume 83 (1995) 1111, and references therein.

[2] R. C. Hofer and M. L. Loper, "DIS Today", *Proceedings of the IEEE*, Volume 83 (1995) 1124.

[3] C. M. Keune and D. Coppock, "Synthetic Theater of War-Europe (STOW-E) Technical Analysis", NRaD Technical Report 1703 (1995).

[4] K. Boner, C. Keune, and J. Carlson, "Synthetic Theater of War (STOW) Engineering Demonstration-1A (ED-1A) Analysis Report", NRaD Report, May, 1996.

[5] S. Rak, M. Salisbury, and R. MacDonald, "HLA/RTI Data Distribution Management in the Synthetic Theater of War", Proceedings of the Fall 1997 DIS Workshop on Simulation Standards (97F-SIW-119).

[6] R. Cole and B. Root, "Network Technology for Stow 97", Naval Research Laboratory briefing, and private communication.

[7] The documents link on the Globus Project WWW site (http://www.globus.org) points to a number of technical papers and interface specifications.

[8] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", to be published in *International Journal of Supercomputer Applications*.

[9] P. Messina em et al., "Distributed Interactive Simulation for Synthetic Forces", Proceedings of the International Parallel Processing Symposium, Geneva (1997).

[10] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, "MPI, The Complete Reference", MIT Press (1996).

[11] S. Brunett and T. Gottschalk, *Pathfinder: A Scalable Implementation Of ModSAF on SPPs*, CACR Report in preparation.

[12] S. Brunett and T. Gottschalk, *Large-Scale Metacomputing Framework for ModSAF*, Technical Report CACR-152, January 1998.

[13] L. Craymer and C. Lawson, "A Scalable, RTI-Compatible Interest Manager for Parallel Processors", Proceedings of the Spring 1997 DIS Workshop on Simulation Standards.

[14] J. Kohler, S. Narasimhan, J. Pittman, A. Whitlock. "ExGen Software Design Document." Naval Command, Control and Ocean Surveillance Center (NCCOSC), RDT&E DIV, June 18, 1996.

[15] http://cesdis.gsfc.nasa.gov/beowulf, the Beowulf WWW site, and references/links therein.

[16] L. Mengel, "Scenario Modifications for SF Express 50,000 Vehicle Scenario", NCCOSC RDTE DIV Report N66001-97-M-1531 (1997).

[17] L. Craymer and L. Ekroot, "Characterization and Scalability", Proceedings of the 1998 Spring Simulation Interoperability Workshop.