Proceedings of the 1984
Winter Simulation Conference
S. Sheppard, U. Pooch, D. Pegden (eds.)

A LANGUAGE-DIRECTED DISTRIBUTED DISCRETE SIMULATION SYSTEM

An Extended Abstract

Dana L. Wyatt
Dept. of Computer Science
Texas Christian University
Ft. Worth, TX 76129

Sallie Sheppard
Dept. of Computer Science
Texas A&M University
College Station, TX 77843

1. INTRODUCTION

In recent years, there has been a noticable increase of interest into the feasibility and utility of distributed discrete simulation. This has been prompted by an improvement in distributed algorithms and distributed computer systems, as well as the need to economically manage the increasingly complex simulation models found in today's world.

Researchers have centered their efforts around two distinctly different approaches to the design and implementation of distributed discrete simulation systems, with the differences based on what types of tasks are distributed to the individual processors. The more common approach researchers have taken is to develop a system in which the model functions (events) are distributed among the processors (1,3,7,8). The alternative approach distributes simulation support functions (e.g. random number generation) to the available processors (4,9).

A basic philosophical difference separates the two approaches. In the distributed discrete simulation via model function (DDS/MF) systems, designers have chosen to deal with the complex issues of deadlock detection and avoidance and resource management in situations where the event set is larger than the number of processors available in order to maximize the potential parallelism of the model. However, they generally increase speed while sacrificing ease of programming. In the distributed discrete simulation via support function (DDS/SF) systems, designers have chosen to approach the issue from an ease of programming standpoint. They have sacrificed fully utilizing the inherent parallelism of simulation applications in favor of adopting an approach where the architecture of the system is "transparent" to the user.

2. A LANGUAGE-DIRECTED DDS/SF SYSTEM

A research project currently being completed at Texas A&M University involves the development of a multitasked implementation of a microprocessor-based distributed discrete simulation system in which the simulation support functions would be distributed to the available processors. Two prototypes were constructed based on different existing simulation languages: SIMPAS and GASP IV. The information presented here concerns the prototype based on SIMPAS. The GASP IV prototype is discussed in a paper by Krishnamurthi and Young, also found in these proceedings.

The objectives of the SIMPAS phase of this project were (1) to develop a translator to produce a distributed Pascal program from a sequential SIMPAS program, (2) to examine the communication activities which occur between the various components of the DDS/SF architecture in order to determine if the hierarchical design constructed was efficient, and (3) to determine the feasibility and utility of this type of simulation.

2.1 The SIMPAS Language

SIMPAS is a strongly typed, event-oriented, discrete simulation language based on Pascal (2). The extensions to Pascal which SIMPAS incorporates for managing events are similar to those of SIMSCRIPT. Statements such as SCHEDULE, CANCEL, DESTROY, and RESCHEDULE are used for manipulating event notices on the event list; INSERT and REMOVE statements are used for manipulating entities in queues; and event routines are simply Pascal procedures with the keyword "EVENT" replacing "PROCEDURE".

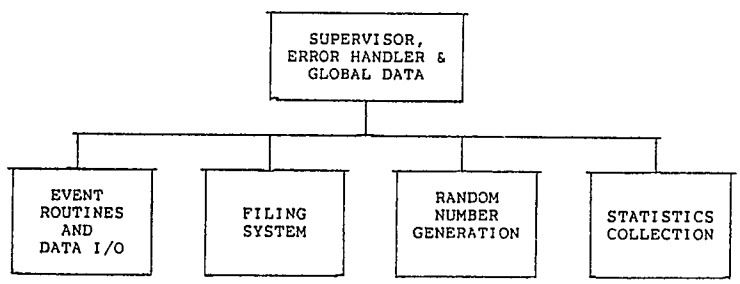

Figure 1: A SIMPAS-based Distributed Discrete Simulation System

This research was supported in part by the National Science Foundation under Grant No. ECS-8215550.

The language translator is implemented as a preprocessor which accepts as input a SIMPAS program and produces as output a Pascal program. This Pascal program must then be compiled on the target computer using that system's Pascal compiler. It is this feature, coupled with it's structured facilities, which led to the selection of SIMPAS as the base language for the DDS/SF system.

2.2 The DDS/SF System Design

The hardware selected for the implementation of this project was a Texas Instruments 990/12 minicomputer running the DX10 operating system. The TI 990/12 is a 16-bit minicomputer which supports multitasking applications with ease. The system allows up to 8 64K byte tasks to be running simultaneously, with communication between tasks accomplished either via messages or a common data area associated with multiple tasks.

An initial analysis of the SIMPAS language yielded three basic types of support functions it provides: random number generation, statistics collection, and event and queue maintanence. Based on this, the system was designed with the hierarchical architecture shown in Figure 1. A supervisor module was added to maintain control over the sub-modules and to handle any errors which might occur.

2.3 Implementation Constraints

Due to the method in which the TI 990/12 handles tasks and data access, some modifications were made to the SIMPAS language. Most noticable of these is the method in which space is allocated for temporary entities. SIMPAS traditionally makes use of the dynamic storage allocation facilities of Pascal when creating new event notices and entities, however, the TI 990/12 uses a relative addressing scheme for mapping a pointer to the actual data area it references. The pointer is treated as an offset from heap storage allocated to a task, and since each task is allocated it's own heap, pointers in one task are not valid in another. Therefore, the multitasked implementation of SIMPAS uses arrays for it's event notices and entities, similar to that of GASP IV.

2.4 Monitoring Communications Between Modules

One of the objectives of this research project was to analyze the communication activities which occurred between the various modules in the system. This would show some indication of the size and volume of message traffic which would be found on the communication lines between the processors which would hold these tasks. Since each message is of a prescribed format, only the number and type of messages is recorded. This communication monitor was implemented not as an independent module, but included in the invocation of each message and is thus distributed throughout the programs.

As of this writing, three sample programs have been tested and communication measurements collected. They include a simple M/M/1 queueing system, the traditional harbor simulation problem, and an inventory simulation problem. As expected, initial analysis indicates a very heavy traffic flow between events and filing routines, moderate flow between events and statistics and filing and statistics, and a lighter flow between events and random number routines. However, a further analysis of messages which require an immediate reply and postpone execution of the sender until such a reply is received (e.g. request for information from a queue) is expected to yield a method of placing priorities on message communications.

3. CURRENT STATUS

This project is currently in it's final phase of testing, lacking only the implementation of several more varieties of simulation applications in order to provide a larger base of communication measurements from which to draw conclusions concerning the functional breakdown of the DDS/SF system presented here. The implementation and execution of this prototype system based on SIMPAS does illustrate that the DDS/SF approach to distributed discrete simulation is a viable alternative to traditional sequential simulation. It provides the user with the ease of programming in a language that is familiar to him while still allowing some of the advantages of parallel processing to be gained. The user has no need to concern himself with the actual organization of the distributed system. In this manner, not only would the system gain the speed-up of executing the events in parallel, but would also gain some speed-up by allowing the time needed for support functions to be reduced.

REFERENCES

1. Bryant, R.E., "Simulation on a Distributed System," 1979 Distributed Computing Systems Conf., pp. 544-552.

2. Bryant, R.M., SIMPAS Users Manual, Dept. of Computer Science and Academic Computing Center, University of Wisconsin - Madison, Madison, WI, 1981.

3. Chandy, K.M. and Misra, J., "Asynchronous Distributed Simulation Via a Sequence of Parallel Computations," Comm. ACM, 24, 4, pp. 198-206, 1981.

4. Comfort, J.C., "The Design of a Multi-Microprocessor Based Simulation Computer-I," Proc. of 15th Annual Simulation Symp., pp. 45-53, 1982.

5. Enslow, P.H., "Multiprocessor Organization - A Survey," Computing Surveys, 9, 1, pp. 103-129, 1977.

6. Notkin, D.S., "An Experience with Parallelism in Ada," SIGPLAN Notices, 15, 11, pp. 9-15, 1980.

7. Peacock, J.K., Wong, J.W. and Manning, E.G., "Distributed Simulation Using a Network of Processors," Computer Networks, 3, 1, pp. 44-56, 1979.

8. Reynolds, P.F., "Active Logical Processes and Distributed Simulation: An Analysis," Proc. 1983 Winter Simulation Conf., Vol. 1, pp. 263-264.

9. Wyatt, D.L. , Sheppard, S. and Young, R.E., "An Experiment in Microprocessor-Based Distributed Digital Simulation," Proc. 1983 Winter Simulation Conf., Vol. 1, pp. 271-277.