

AD-A096 172

MITRE CORP BEDFORD MA

F/G 9/2

APPLICABILITY OF DSDS SIMULATION MODELING SYSTEM TO ESD SYSTEM --ETC(U)

FEB 81 J K FRYER

F19628-81-C-0001

UNCLASSIFIED

MTR-8187

ESD-TR-81-114

NL

100  
A  
0000

U

END

DATE

FORMED

4

DTIC



ESD-TR-81-114

**LEVEL**

12  
B.S.

MTR-8187

APPLICABILITY OF DSDS  
SIMULATION MODELING SYSTEM  
TO ESD SYSTEM ACQUISITION PROBLEMS

BY JEFFREY K. FRYER

FEBRUARY 1981

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS  
ELECTRONIC SYSTEMS DIVISION  
AIR FORCE SYSTEMS COMMAND  
UNITED STATES AIR FORCE  
Hanscom Air Force Base, Massachusetts



SDTIC  
MAR 10 1981  
C

AD A 096 172

DDC FILE COPY

Approved for public release;  
distribution unlimited.

Project No. 522M

Prepared by

THE MITRE CORPORATION  
Bedford, Massachusetts

Contract No. F19628-81-C-0001

81 3 10 008

When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

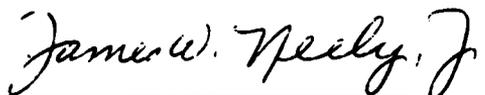
Do not return this copy. Retain or destroy.

#### REVIEW AND APPROVAL

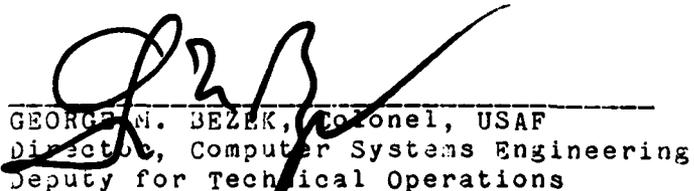
This technical report has been reviewed and is approved for publication.



BRADLEY W. UTZ, Capt, USAF  
Project Manager  
Computer Engineering  
Applications Division



JAMES W. NEELY, Jr., LTC, USAF  
Chief, Computer Engineering  
Applications Division



GEORGE M. BEZEK, Colonel, USAF  
Director, Computer Systems Engineering  
Deputy for Technical Operations

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER <b>18</b> ESD/TR-81-114	2. GOVT ACCESSION NO. AD-A094 172	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) APPLICABILITY OF DSDS SIMULATION MODELING SYSTEM TO ESD SYSTEM ACQUISITION PROBLEMS		5. TYPE OF REPORT & PERIOD COVERED <b>9</b> Technical rept.	
7. AUTHOR(s) <b>30</b> Jeffrey K./Fryer		14. PERFORMING ORG. REPORT NUMBER <b>14</b> MTR-8187	6. CONTRACT OR GRANT NUMBER(s) <b>15</b> F19628-81-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS The MITRE Corporation P. O. Box 208 Bedford, MA 01730		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS  Project No. 522M	
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Technical Operations Electronic Systems Division, AFSC Hanscom AFB, MA 01731		13. REPORT DATE <b>33</b> FEBRUARY 1981	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report)  UNCLASSIFIED	
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release; distribution unlimited.		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) DSDS DISCRETE EVENT MODELING MODELING SYSTEM SIMULATION			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) DSDS (Data Systems Dynamic Simulator) is a performance modeling tool developed for NASA by General Electric for the simulation of computer-communications systems. The system has been installed on the MITRE IBM 370. This paper presents the results of implementing five benchmark problems that were designed to be representative of problems confronting ESD/MITRE systems engineers. DSDS was found to be relatively easy to learn and use. However, the system must be made more efficient and undergo a thorough testing and validation effort before it can be considered a useful tool for (over)			

235000

lpg

**UNCLASSIFIED**

**SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)**

**20. ABSTRACT (Concluded)**

**ESD/MITRE use.**

**UNCLASSIFIED**

**SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)**



## TABLE OF CONTENTS

<u>Section</u>		<u>Page</u>
	LIST OF ILLUSTRATIONS	5
	LIST OF TABLES	5
1	INTRODUCTION	7
2	MODELING THE FIVE BENCHMARK PROBLEMS	8
	SYSTEM DIFFICULTIES	8
	System Errors	8
	System Limitations	11
	Model Debugging	12
	BENCHMARK PROBLEM IMPLEMENTATION	12
	Benchmark Problem I	13
	Benchmark Problem II	13
	Benchmark Problem IIa	14
	Benchmark Problem III	14
	Benchmark Problem IV	14
	Benchmark Problem V	15
3	RESULTS OF EVALUATION	16
	EVALUATION FACTORS	16
	System Performance Time	16
	System Learn Time	16
	Input Language	18
	Model Build Time	18
	Diagnostics	18
	Modifiability	18
	Output Reports	18

TABLE OF CONTENTS (Concluded)

<u>Section</u>	<u>Page</u>
User Documentation	19
Use Costs	19
ENHANCEMENTS	19
4 SUMMARY AND CONCLUSIONS	22
LIST OF REFERENCES	23
APPENDIX	24

LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
A-1	DSEM Diagram of Problem I Subnet Node 3	25

LIST OF TABLES

<u>Table</u>		<u>Page</u>
1	DSEMs Used for Benchmark Problems	9
2	DSDS Evaluation Factors	17
A-1	DSDS Output Statistics	27

## SECTION 1

### INTRODUCTION

The primary objective of the Performance Modeling and Simulation Project is to acquire computer-communications system performance modeling tools for use during the conceptual phase of C3 system acquisitions. These tools will assist the Electronic Systems Division Program Offices (ESD POs) and supporting MITRE personnel in performing conceptual-phase system engineering activities and analyses.

One type of performance modeling tool MITRE has been examining is termed a General Purpose Modeling System (GPMS). This type of system uses a high-level language to create a system model from a library of preprogrammed submodels. Four such modeling systems -- NASA's DSDS (Data Systems Dynamic Simulator), TRW's PERCAM (Performance and Cost Analysis Model), Hughes' DAS/DDPM (Design Analysis System/Distributed Data Processing Model), and Martin Marietta's ARES (Automated Requirements Engineering System) -- have been examined. All the systems except DSDS were either unavailable or undesirable for one reason or another. DAS/DDPM is written in a proprietary language and is therefore unavailable. ARES is not yet complete nor validated. PERCAM is primarily a system for modeling war games and is not suitable for modeling computer-communications systems. DSDS, on the other hand, is both available to MITRE and applicable to ESD/MITRE problems, so it was selected for hands-on testing. A brief description of DSDS is given in the appendix. For further information on the design and use of DSDS, see [GRAF79, HOOP78, NASA76, NASA79, and NASA80].

This report presents the results of implementing five benchmark problems using the batch version of DSDS and discusses how DSDS might be made more useful to ESD/MITRE. The benchmark problems were developed to assist in the evaluation of modeling systems. DSDS's applicability to these problems is presented in section 2, followed in section 3 by an overall evaluation of DSDS and a discussion of potential enhancements to the system. A brief summary of the report is given in section 4.

## SECTION 2

### MODELING THE FIVE BENCHMARK PROBLEMS

This section discusses the results of applying DSDS to the five benchmark problems. These problems represent a variety of systems engineering problems that are expected to confront ESD/MITRE systems engineers. The first part of this section discusses DSDS system difficulties that apply to all benchmark problems. The second part describes the implementation status of each benchmark problem and any specific difficulties encountered. Benchmark problem I was implemented by GE as part of the previous evaluation effort [NASA79]. Benchmark problems II through V were implemented for this study.

#### SYSTEM DIFFICULTIES

Many difficulties were encountered in modeling and testing the benchmark problems. Due to limitations in the DSDS documentation and the vagaries of certain DSDS system elements, some of the models had to be changed during testing and debugging. For example, the size of the model for benchmark problem II varied between 110 and 350 DSEMs (Data Systems Element Models) with final size about 150. System difficulties encountered in implementing the benchmark problems can be divided into system errors, system limitations, and model debugging. A description of each follows.

#### System Errors

A major problem was encountered with the DSEMs, the basic modeling elements of the system. Each DSEM is a preprogrammed, parameterized model of a system element. The user supplies parameter values and links the DSEMs together to create a system model. About 150 DSEMs are currently available to the user. However, only 23 DSEMs were needed to model the benchmark problems. These DSEMs are listed in table 1.

DSEMs that did not work as specified in the user's manual include DATSEN, USER02, DATGEN, DMANDQ, and RANROU. Both the DATSEN (Data Sensor) and USER02 (termination) DSEMs did not work for message transactions. As a result, all DATSEN blocks had to be replaced with two other DSEMs, DUPLIC (Duplicate) and SETSW (Set

Table 1

DSEMs Used for Benchmark Problems

DSEM Code	DSEM Name
ADDMUL	Add, Multiply
ANDAT5	ANDs Attribute 5
CHKPAR	Check Parameters
DATGEN	Data Generator
DELAY	Fixed Time Delay
DMANDQ	Demand Queue
DUPLIC	Duplicator
INVRT5	Inverts Attribute 5
ORATR5	ORs Attribute 5
PROCER	Processor
QC0002	Quality Control
RANATR	Random Attribute
RANROU	Random Routing Switch
SELSW	Selector Switch
SETVAL	Set Value
SETSW	Set Switch
SINCO2	Sink
STEPSW	Step Switch
THRUOT	Throughput Finish Time
THRUST	Throughput Starting Time
TIMU02	Timing Unit
UNITRY	Initialization Control Generator
USER02	User Terminator

Switch), to perform the same function. An option of the USER02 DSEM is to generate a response message. USER02 blocks used in this manner also had to be replaced by two other DSEMs -- RANATR (Random Attribute) and SETSW.

Another DSEM that caused difficulties was DATGEN (Data Generator). It was found, although it wasn't documented in the user's manual, that certain parameters could not be set to certain values. For example, the user could not generate a message transaction with a transmission time delay although the user's manual states that this is permissible. Trying to use the DATGEN in this way stopped the program and produced a large dump. If a Poisson distribution of interarrival times is specified in a DATGEN, the resulting distribution is close to uniform. On General Electric's advice (GE developed DSDS and maintains it for NASA), we replaced the Poisson distribution with an exponential distribution which functioned properly.

The DMANDQ (Demand Queue) DSEM set attribute 10 (a transaction variable) to 0. The DSEM was corrected so attribute 10 would remain unchanged. Another problem with the DMANDQ occurred late in the implementation of benchmark problem II. Starting with about the 70th message through the system, a number of extra submessages were generated in the model. Eventually this was traced to a DMANDQ DSEM. When this single DSEM was replaced with several other DSEMs that performed the same function, the problem disappeared. Further difficulties with the DMANDQ occurred in testing benchmark problem IIa. Messages would not go through the DMANDQ DSEM, so all DMANDQ's in the problem were eventually replaced.

During testing of benchmark problems II and III, it was discovered that the RANROU (Random Routing) DSEM did not work correctly. This DSEM would not route messages uniformly to their destination -- some destinations received many more messages than others. This fault was overcome in some places by using the RANATR DSEM which performs the same function in a slightly different manner. (This also required two DSEMs instead of one.)

Different system errors appeared with each problem tested. Support from GE was adequate but they rarely corrected the errors. Instead, they suggested ways to get around the problem by using different DSEMs. As a result, the following errors are still in the system:

1. DATSEN does not work for message transactions.
2. USER02 does not properly generate responses to message transactions.

- 3) DATGEN cannot generate a message transaction with a transmission-time delay.
- 4) DATGEN's Poisson distribution of interarrival times is incorrect.
- 5) DMANDQ does not work.
- 6) RANROU does not route messages randomly.

Some higher level of support would be required from GE to get these and other errors out of the system.

#### System Limitations

The major limitation imposed by DSDS is the high cost of running models. DSDS was installed on the MITRE IBM 370 and optimized as much as possible without actually changing it. However, due to its high use costs the system is run only at night and even then its costs are significant -- \$20 to \$50 per run for most benchmark problems. Running the system at night also necessitates a lengthy debugging process allowing only one user run per 24 hours.

Another important limitation of DSDS is the lack of attributes available to the user. Attributes are used to store information about a transaction. For example, it is often necessary to know the type, origin, and destination of a message. These parameters can be stored in attributes which are sent along with the message. The user processes and routes messages through the system based on the values in these attributes. DSDS has 11 attributes available, but attributes 1 through 8 are used by the system. Thus only three attributes, 9 through 11, are available for message parameters. This was a serious deficiency for benchmark problems II and V. In both cases four attributes were required, so one attribute had to be coded to take the place of two.

The statistics produced by DSDS impose several limitations:

- Statistics on use cannot be obtained directly for a resource made up of several DSEMs. This limitation was only encountered in benchmark problem V and was circumvented in a way that would make the problem very costly to run. DSDS has a set of DSEMs that could model such resources and automatically collect the required statistics, but they have never been debugged.

- DSDS does not produce minimum and maximum throughput time statistics. This affects several of the benchmark problems. There are cases where system performance requirements dictate maximum throughput times and it is therefore desirable to be able to collect such statistics.
- Statistics are rounded off to tens of milliseconds where accuracy to milliseconds may be needed. This affects those problems where system performance requirements, such as throughput, are specified in milliseconds.

### Model Debugging

Difficulties were encountered in debugging benchmark problems II and V. It is easy to determine the origin of problems that result in DSDS error messages, but logical errors that do not result in error messages are more difficult to trace and correct. DSDS provides trace statements called PULLATs. These allow the user to specify DSEMs at which the user wants to collect data. When these DSEMs are entered, the contents of attributes 1 through 11 are printed along with the time. Although some useful information can be obtained this way, it is often very difficult to trace a given message through the system. For example, attribute 6 contains the message number but each time a DUPLIC (Duplicate) DSEM is encountered, only the first message duplicated keeps the message number. Further problems are caused by the start- and end-time controls for the PULLATs. The times given by the user are rounded off to the nearest second. In problems dealing with milli- and microseconds, this means that if PULLATs are inserted the user will probably receive many more pages of output than needed.

GE informed us of another DSEM that can trace through the model -- DATCO. DATCO is supposed to take one message and list every DSEM it or its submessages enters. However, this DSEM has not been used since the system was developed and has never been documented. GE was not sure the DSEM worked and, as it turned out, it didn't. It does not trace when it is supposed to nor does it give a correct trace.

### BENCHMARK PROBLEM IMPLEMENTATION

All five of the benchmark problems have been modeled using DSDS. Benchmark problems I, II, III, and V have also been implemented and tested to differing degrees. The paragraphs that follow describe the implementation status of each problem and any unique difficulties or limitations found during implementation.

### Benchmark Problem I

Benchmark problem I specifies a model of a communications network described by its topology. GE implemented this problem using DSDS. To simplify the modeling task, two new DSEMs were created. One of these, PROCER, is used for modeling different types of queueing and processing functions. Without this DSEM, modeling any of the benchmark problems would be much more difficult. The other new DSEM, ADDMUL, is used to change the value of an attribute by multiplication or addition. This DSEM was used for benchmark problem I to simulate data compression by multiplying attribute 8 (which stores the number of characters in a message transaction) by .5.

GE used the interactive version of DSDS to implement problem I. (The DSDS system installed at MITRE is the batch version.) The major benefit of the interactive version is its curve plotting feature, which GE used to plot the number of messages in the system versus time. If this value increased continuously, then either the model was incorrect or the system was unstable.

GE implemented all seven system configurations specified in the problem. However, a major limitation was imposed on the implementation of problem I. DSDS was not able to perform message segmentation.

### Benchmark Problem II

Benchmark problem II is the most complex of the benchmark problems that have been implemented using DSDS. It requires modeling a bus communications system consisting of a control bus, a data bus, varying numbers of hosts, and a Network Control Element (NCE). The actual modeling effort took place from December through March. Testing began in early April, but the problem was not actually running correctly until mid-July. A total of 44 man-days (9 weeks) was spent on this problem -- about half the time in modeling and re-modeling the problem and the other half in debugging it.

Benchmark problem II was tested using the first configuration specified in the problem -- four hosts generating 10 messages per second. Changing the number of messages generated per second simply involves changing a variable in the DATGEN DSEM. Increasing the number of hosts requires an increase of 23 DSEMs per host added. Thus a 7-host network requires about 220 DSEMs while a 10-host network requires about 290 DSEMs.

Some features of the problem not modeled because of the amount of extra code required were slotted buses and the collection of certain statistics.

#### Benchmark Problem IIa

A slightly different version of benchmark problem II was also modeled and an attempt made to implement it. This version required modeling a bus communications system without an NCE and without a control bus. Before sending a message on the bus, a host checks to see if the bus is available. If so, a message is sent and an acknowledgement returned. If two hosts send messages out on the bus at the same time and a collision results, an acknowledgement is never received and the message is retransmitted.

The approach taken to modeling this problem differed from that taken in modeling problem II in one respect -- the use of sets. DSEMs were put in four different sets to generate messages at the four hosts. Linking these sets proved to be more difficult than expected and is not covered in the user's manual. A number of telephone calls to NASA/GE were required to solve this difficulty. The problem was still being debugged when testing was halted. Six man-days were spent in implementing this problem.

#### Benchmark Problem III

Benchmark problem III compares three loop architectures -- the Pierce Loop, the Newhall Loop, and the Distributed Loop Computer Network (DLCN). The Pierce Loop was the first problem modeled with DSDS. This simple loop was an ideal initial problem because it required only a short DSDS program (29 DSEMs). Modeling and running the program took only about seven days and several different runs were made. The Newhall Loop and DLCN were both modeled but not tested.

#### Benchmark Problem IV

Benchmark problem IV requires the modeling of a local computer network employing a loop as the communications subsystem. The only difficulty in modeling the problem was its size. Modeling 6 hosts with 50 terminals attached to each host would have required an extremely large DSDS program. Thus the size of the system was cut to three hosts with three terminals each. Even modeling this abbreviated version required about 160 DSEMs. Each added host (with 3 terminals attached) would require 50 more DSEMs and each added terminal would require about 10 more DSEMs. Thus 6 hosts with 50

terminals each would require over 3000 DSEMs. Some of this code could probably be combined, but the program would still be very large and costly to run.

#### Benchmark Problem V

This problem differs from the other benchmark problems in that it requires detailed modeling of a computer system. DSDS has a special set of DSEMs capable of this level of modeling detail and the problem was first modeled using these so-called third-level DSEMs. However, there were two reasons for serious doubts as to whether this model would work: the DSDS user's manual description of the use of these DSEMs is very unclear, and GE informed us that some of the DSEMs were not well tested and might not work. As it turned out, it was necessary to model benchmark problem V without using third-level DSEMs.

The main difficulty in developing this model was in collecting statistics. For example, modeling the activity of a display console without using a third-level DSEM requires several DSEMs. With DSDS it is not directly possible to compute one utilization for the console when the console is made up of several DSEMs, so an area in storage available to the user and defined as the array SETGET (I) (I=1, ... 100) was used. For example, SETGET (1) was set to 1 when the console was busy and to 0 when it was not busy. At regular, brief intervals the value in SETGET (1) was checked to get the required utilization. CPU and storage utilizations were obtained the same way. This procedure increased the size of the DSDS program, the possibility of errors, and the cost of a computer run.

As with benchmark problems II and IV, the size of the problem was cut back severely. Ten to twelve terminals are specified in the problem but the model used only 3 terminals and still contained 139 DSEMs. Each additional terminal would have required 14 additional DSEMs. Thus a 10-terminal system model would have required 237 DSEMs.

When testing was terminated, the model was exceeding the specified time limit and routing messages incorrectly. The debugging aids in DSDS were of little help in tracing such problems. Twenty-one days were expended modeling the problem and another four attempting to run it. Most of the modeling time was spent trying to understand a rather complex problem and then figuring out how to use the third-level DSEMs.

## SECTION 3

### RESULTS OF EVALUATION

The purposes of the study were to evaluate DSDS via actual use and at the same time develop a set of recommended enhancements to make DSDS more useful for modeling ESD/MITRE system problems. Implementing the benchmark problems provided the basis for a more detailed evaluation than previously possible.

#### EVALUATION FACTORS

A number of evaluation factors were developed in a preliminary study. Table 2 lists these factors together with the ratings given in that study and the ratings given as a result of this evaluation.

#### System Performance Time

DSDS efficiency is poor. DSDS required about 60 CPU seconds for every second of simulation time to run benchmark problem II. By way of comparison, a GASP IV program simulating benchmark problem II required less than 2 CPU seconds for every second of simulated time. To run the Pierce Loop (benchmark problem III) also required a great deal of CPU time considering the small size of the problem. Generating 1.7 messages per second per host for six hosts required almost 2 seconds of CPU time for every second of simulated time.

#### System Learn Time

DSDS is relatively easy to learn compared to a standard simulation language. Although the system provides over 150 DSEMs, only about 25 should be necessary to model most ESD/MITRE problems. Most users could get a good understanding of DSDS in a week. However, to facilitate learning and use, the user's manual needs substantial revision and expansion. For example, it needs an expanded section on the operation of the batch system, an expanded section on the DSEMs most applicable to ESD/MITRE problems and better sample problems.

Table 2  
DSDS Evaluation Factors

Evaluation Item	Result*	
	Preliminary Study	After Testing
System Performance	3	2
Useability (by Analyst)		
System learn time	4	4
Input language	4	4
Model build time	4	4
Diagnostics	4	3
Model modifiability	5	3
Output reports	4	3
User Documentation	4	3
Use Costs	2-3	1

\*Rating: 0-Not available, 1-Very poor, 2-Poor, 3-Fair, 4-Good, 5-Excellent.

### Input Language

The input language earns a rating of good. It requires from three to six lines of code per DSEM to reflect information from the model diagram including coded parameters for each DSEM. However, the diagram from which the DSEMs are generated is, if kept up-to-date, a more readable and more understandable record of the model than the list of DSEMs.

### Model Build Time

A DSDS model is easy to build. Most models were constructed in a few man-days once both the problem and the use of the DSEMs were understood. However, original model designs usually went through a number of changes as more was discovered about DSDS. Another factor affecting model build time is that the system is too costly to use during the day. This means that only one debug run can be made every 24 hours versus 2, 3, or more on a system that can be run during working hours. (See Use Costs below). This results in a very lengthy debugging process.

### Diagnostics

DSDS is very good at pointing out programming errors, but it is less helpful in detecting logical errors. PULLATs are available to trace transactions through the system, but they have certain limitations that are discussed in section 2, Model Debugging. Thus DSDS diagnostics are only rated fair. If the user could get an easy-to-read trace of a message through the system, DSDS's diagnostic rating could be changed to very good. (This capability is in fact available on the interactive version of DSDS.)

### Modifiability

The modifiability of a DSDS program can be rated fair. It is easy to change the rate at which messages are generated, but changing message routing is more difficult. This is especially true when one attribute is serving more than one purpose, as in problem II or V. If DSDS allowed the user to use more attributes, DSDS's rating for modifiability could be increased to good.

### Output Reports

DSDS output reports receive a rating of fair. DSDS computes a lot of statistics, but a few are missing. Missing statistics include minimum and maximum throughput times from DATGEN to USER02, from THRUST (Throughput Start Time) to THRUOT (Throughput Finish

Time), and from individual PROCER DSEMs. Mean throughput statistics are given rounded to hundredths of seconds. This is inadequate for some problems.

The user also does not have enough control over the output. At times it would be useful to either print out intermediate results or to reset the statistics and start computing them from the present point in the program. Neither can presently be done using DSDDS. In addition, the batch version of DSDDS does not have the capability to produce graphs and histograms.

#### User Documentation

The user documentation earns another rating of fair. The latest user's manual is an improvement over previous manuals, but it still has some deficiencies. It contains too much information on the interactive version of DSDDS and not enough on the batch version, and it has inadequate descriptions for the use of some DSEMs. The DSDDS user's guide also needs updating. The principal need is to change the examples from NASA-oriented problems to ESD/MITRE oriented problems.

#### Use Costs

As previously mentioned, DSDDS uses a lot of CPU time. However, this is not the only cost that is high. To run DSDDS requires either a large number of disk accesses (up to 33,000) or a 7300K byte region size. Both approaches are prohibitively expensive except at night. At MITRE there is a 40 percent discount for deferred (night) jobs, and the maximum region the user is charged for is 1000K. Thus a 5-second run for benchmark problem II that would cost about \$225 during the day costs about \$65 at night. This is still very expensive for just one short run -- 5 seconds may not be long enough to tell the user anything meaningful about the system modeled. If a number of sensitivity analyses need to be made using DSDDS, the costs could quickly become very high.

#### ENHANCEMENTS

It is clear from using DSDDS to model the benchmark problems that a number of enhancements are necessary to make DSDDS a useful tool for modeling ESD/MITRE problems. Some of the things that can be done are:

- Improve efficiency
- Correct system errors
- Increase the number of attributes

- Improve tracing facilities
- Improve SETGET capabilities
- Add throughput statistics
- Provide run-time statistics
- Provide more significant digits
- Produce graphs and histograms
- Allow use of subroutines
- Provide user access to random number seeds

The most important improvement would be a reduction in the high computer costs of using DSDS. Improved efficiency would not only reduce costs: sufficient improvement would allow the system to be used during the day, thereby also reducing model build time.

A number of system errors were discovered during implementation of the benchmark problems (see section 2). These errors need to be corrected and the system thoroughly tested to eliminate as many other errors as possible.

Another important enhancement is to increase the number of attributes. Eleven attributes are not enough if eight of them are used by the DSDS system. GE is planning changes that will make fourteen more attributes available to the user on the interactive version of DSDS. This change could also be implemented on the batch system.

A better means of tracing messages through a DSDS model is also necessary. The DATCO DSEM would help if it were documented and if it worked. Another solution is for the user to number each message by setting one attribute to a message number. However, this requires yet another attribute when DSDS already has too few.

Another enhancement, that might be considered is the ability to route on values in the SETGET array and collect statistics on those values (average, minimum, maximum, etc.). For example, if SETGET (1) contains a value indicating the status of a terminal (0=idle, 1=busy) the routing capability would allow the user to route messages elsewhere if the terminal is busy. If statistics were gathered, the utilization of the terminal would automatically be computed. This would greatly simplify the implementation of problem V.

It would also be useful to allow multiple THRUST/THRUOT (throughput statistics) pairs. Currently if two THRUST statements are encountered a message only remembers the second. Thus throughput statistics are only collected between the last THRUST statement and a THRUOT statement. It is often useful to get throughput times from more than one point in the system, but right

now this can only be done by making separate runs. Minimum and maximum throughput times for THRUST/THRUOT pairs, DATGEN/USERO2 pairs and individual PROCER blocks are not available, but these are necessary particularly where maximum throughput requirements are specified.

The user ought to be able to print out intermediate results or reset the statistics and start computing from a point other than the initial start time. This feature is partially implemented in the interactive DSDS (a run can be interrupted and the model changed), but queues are not reset when execution resumes.

Since many communication problems deal in milli-, micro-, even nanoseconds, DSDS must provide statistics that have more accuracy. The system presently rounds output to hundredths of seconds.

Graphs and histograms of various queue values and utilizations should be provided as optional output reports. The system does not provide this statistical information to the user, so such graphs cannot even be prepared manually.

Two other enhancements are worth considering. One is to allow the use of subroutines in DSDS -- this might make a DSDS program shorter and easier to read. The other is to allow the user to alter random number seeds in order to make model replications which otherwise would always be identical.

## SECTION 4

### SUMMARY AND CONCLUSIONS

This report presented the results of an evaluation of a general purpose modeling system, DSDS, using five benchmark problems representative of C3 systems. All problems were modeled and required from 1 to 9 man-weeks of effort per problem. Based on these study results, DSDS in its present form is not recommended for general use by ESD/MITRE systems engineers. Before such a recommendation could be made, a number of improvements are necessary.

The most important improvement is to make the system more efficient. This would reduce user costs (the major system limitation) and allow the system to be used during the day. Being able to test a model several times a day would also significantly reduce model development time.

The other major improvement needed is to make the system more error free. A number of system errors were encountered during the implementation of the benchmark problems. These indicate that a comprehensive testing, validation, and maintenance effort would be required.

It has been proposed that MITRE examine the interactive version of DSDS to see how it compares with batch version. The interactive system is presently hosted on a VAX time sharing system at NASA/Goddard and on a dedicated PRIME at NASA/Marshall.

## LIST OF REFERENCES

- GRAF79 H. A. Graf, "Training Manual for Data Systems Dynamic Simulator," Doc. 78HV091, General Electric Company, Huntsville, Alabama, 1 September 1979.
- HOOP78 J. W. Hooper and D. W. Rowe, "Data Systems Dynamic Simulator -- A Total System for Data System Design Assessments and Trade Studies," The Eleventh Annual Simulation Symposium, Tampa, Florida, 15-17 March 1978.
- NASA76 National Aeronautics and Space Administration, "Study to Establish Models and Simulation for Data Systems, Final Report, Volume 2: User's Manual," Marshall Space Flight Center, Huntsville, Alabama, 17 June 1976.
- NASA79 National Aeronautics and Space Administration, "Data Systems Dynamic Simulation (DSDS), Simulation Analysis Report, Communications Network Design Problem," Marshall Space Flight Center, Huntsville, Alabama, October 1979.
- NASA80 National Aeronautics and Space Administration, "Interactive Data Systems Dynamic Simulation (DSDS) User's Manual," Marshall Space Flight Center, Huntsville, Alabama, January 1980.

## APPENDIX

### DSDS SYSTEM DESCRIPTION

The NASA Data System Dynamic Simulator (DSDS) was developed for NASA by GE. DSDS, designed to simulate large data processing and communications systems, has been used to support trade off studies of NASA's data system needs in the 1985 to 1990 time frame. It is designed to perform comprehensive design analyses and trade off studies of alternate system configurations, and to reduce the time and cost of performing simulations by using preprogrammed parameterized models of system elements. DSDS simulates timing, control and sizing characteristics, and has features to estimate the cost and personnel support characteristics of the system simulated. DSDS uses a building block approach: just as hardware and software elements are pieced together to form a system, so are models of the elements pieced together for system simulation.

DSDS uses modular entities called Data System Element Models (DSEMs), which are parameterized models of system elements, such as a multiplexer, tape unit, software task, central processor, terminal, etc. The equations and protocol that duplicate the functions of these elements are provided in the form of subroutines. Functionally, a DSEM subroutine processes data entering the element according to user-specified performance parameters. This processing produces element throughput characteristics and generates demand for computer resources and personnel resources.

DSEMs consist of mathematical and logical relationships, all preprogrammed. Only characteristic parameters are specified to tailor the DSEM to the user's requirements; no programming skill is required. Modelers are provided block diagrams of each DSEM with labeled inputs and outputs. Elements are interconnected to form complete system models by directing outputs from one DSEM to inputs of another, in the same manner that hardware is interconnected. Figure A-1 shows the DSEMs required to represent a subnet node in benchmark problem I.

The DSDS model library contains DSEMs modeled to three levels of detail -- Global, Intermediate, and Subsystem -- Global models being the least, and Subsystem models the most detailed. The different levels of DSEMs may be intermixed within a single system model. Thus, once a portion of a system has been satisfactorily designed, modeled and "shaken down," it may be replaced by a higher-level DSEM

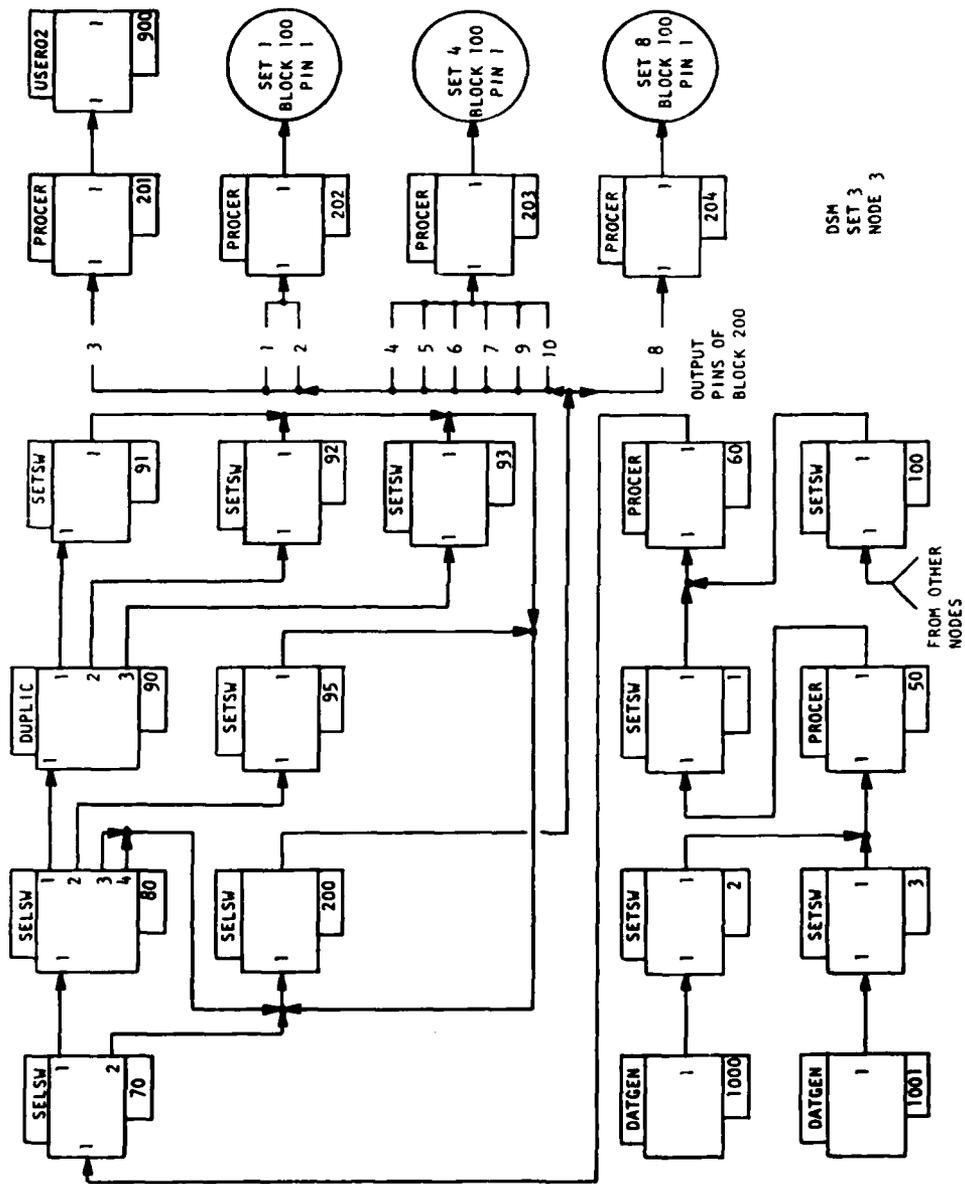


Figure A-1. DSEM Diagram of Problem I Subnet Node 3

(or DSEMs) while other portions of the system are studied in detail. Use of a higher-level DSEM saves computer time by eliminating unnecessary model detail.

Other DSDS system components include a computer resource scheduler, a personnel resource scheduler and an executive. Computer resources requested by a DSEM are scheduled and provided as they become available. A record is kept of the demand for these resources. Similarly, personnel resources are requested by DSEMs and scheduled by the personnel resource scheduler. The executive connects the DSEMs and controls the flow of the simulation.

The system produces nine different throughput and resource utilization statistics reports, and five different operational reports. Statistics (table A-1) include performance statistics on messages, utilization statistics for system elements, and statistics regarding the need for operational support.

DSDS is owned by the government (NASA) and is currently available for installation on other computer systems. There are two versions of the system -- batch and interactive. The batch system is installed on an IBM 360/75 at Huntsville. The interactive version is installed on a PRIME computer at Huntsville and on a DEC VAX system at NASA-Goddard. GASP IV, a FORTRAN-based simulation language, was used to develop the DSDS system software and FORTRAN was used to implement the individual DSEMs.

Table A-1

DSDS Output Statistics

Message Statistics

- Transit time - mean and standard deviation
- Number of messages by origin-destination pairs
- Number of bits by origin-destination pairs

System Element Statistics

- Throughput - mean
- Utilization - percent
- Transmit time - mean

Storage Statistics (per device)

- Utilization - percent
- Messages in storage - mean, standard deviation, maximum, minimum
- Time in storage - mean, standard deviation, maximum, minimum

Queue Statistics (per element)

- Messages in queue - mean, standard deviation, maximum, minimum
- Time in queue - mean, standard deviation, maximum, minimum

CPU Statistics (per device)

- Utilization - mean, percent
- Queueing - maximum

Operations Statistics

- Manpower per skill code - hours, percentage
- Personnel per skill code - available, maximum demand, average demand
- Queueing per skill code - mean, standard deviation, minimum, maximum

Cost Statistics

- Per element
- Per facility

EN

DAT

FILME

4

DTIC