A MULTI-PURPOSE (COMPUTER) SYSTEM SIMULATION (MPSS) MODEL AND ITS CALIBRATION AGAINST MEASUREMENTS OF THE MONITOR AND CONTROL DISPLAY SYSTEM (MACDS) AT THE NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (NASA) GODDARD SPACE FLIGHT CENTER (GSFC)*

Fred W. Mill, George N. Krebs, and
Erwin S. Strauss,

Computer Sciences Corporation,
Silver Spring, Maryland

ABSTRACT

A Multi-Purpose System Simulator (MPSS) model and an associated calibration analysis are described. MPSS models complex, interconnected computer systems. A system is defined by hardware, software, and data flow characteristics. Hardware entities include central processing units, extended memory banks, and peripheral devices. Elements within a system are linked together by communication lines. Data blocks processed through the system are defined by type, entry rate, block size, and routing through an assigned sequence of software modules. Software modules are described by size, priority, instruction count, and accesses to peripheral devices. Statistics are generated concerning the utilization of central processing units, peripheral devices, data block processing, and software execution.

To establish the credibility of model results, a series of controlled experiments was made on an existing computer ensemble. Analogous experiments were then simulated using MPSS. The model and prototypal results correlated highly. Average data block response times and CPU idle times over several trials had average correlation coefficients of 0.97 and 0.95 respectively. All significant discrepancies between model and prototypal results have been explained.

The model is not capable of simulating the non-linear behavior of the system imposed by data load backlog and does not simulate sequential I/O accesses to the same device in the same manner as the actual system. However, these isolated deficiencies did not significantly affect the overall system performance statistics as generated by MPSS. Accordingly, the calibration study results provide confidence in the fidelity of the model.

*The work described herein was performed by Computer Sciences Corporation in support of the National Aeronautics and Space Administration, Goddard Space Flight Center, under Contract NAS5-20640, Programming and Computing Services for Central Computing Support.

INTRODUCTION

A multi-purpose computer network simulation model, MPSS, has been developed to evaluate the performance of existing and hypothetical multi-computer systems. The model was developed in response to rapidly changing data load and computer network configurations at NASA. With the advent of the Tracking and Data Relay Satellite System (TDRSS) supporting future shuttle missions, an updated computer system is needed to handle the increasing data load. With a simultaneous shifting from large scale computers (e.g., IBM 360 systems) to networks of interactive minicomputers, coupled with the requirements of data-handling reliability in primary and backup systems, the need for performance evaluation and prediction became apparent. In developing a simulation tool to address these needs in an environment where system requirements and data loads are changing, and where the computer hardware and software state-of-the-art is being updated, a very general and flexible modeling capability was desired. The MPSS model described herein has been designed to satisfy these requirements.

To gain confidence in the accuracy of the model to simulate prototypal computer network systems, an extensive calibration study has been performed comparing model results against an existing NASA data acquisition and display system. The system performance measurements for the calibration study were obtained under carefully controlled experimental conditions for a variety of data block types. The calibration study correlations between system measurements and MPSS simulated results showed good overall agreement, while at the same time pinpointing specific limitations of the model.

MODEL OVERVIEW

The MPSS model was developed to evaluate the relative performance of proposed competitive computer systems and to indicate areas for enhancement in existing or proposed systems. Each computer system is composed of a hardware configuration with an associated software design for the computing units. The model is capable of simulating the operation of defined systems and of producing measures of performance for various data loads as results. These measures can then be compared among proposed competitive systems to isolate that system which best satisfies a given set of requirements.

Once a system is selected, the model can be used to monitor the projected performance of the evolving software and/or hardware structure. If areas of data throughput congestion are identified, alternative software redesigns can be tested to assure the alleviation of the congestion in the problem area and to ensure that the redesign does not cause a problem in another area. If a hardware component, rather than a software element, appears to be the problem, the model can be used to evaluate the performance expected by replacing the problem component with alternative substitutes. The performance parameters measured by the model include CPU processing and complementary idle times, data block backlogs, data block response times, individual software module response times, communication line usage, and statistics on peripheral device utilization and contention. These statistics are summarized in Table 1.

A salient feature of MPSS is its ability to simulate a multi-CPU network involving external data entry as well as data transmission among the various CPUs, extended memory banks, and peripheral devices that comprise a system hardware configuration. The model input consists of CPU, memory bank, peripheral device, and device communication line hardware characteristics and linkages, simulated data block types and rates (constant, periodic, or Poisson probabilistic options), and associated executable sequences of software modules for each data block type. For each type of simulated software module, specifications of its number of effective instructions, priority, and detailed peripheral utilization are declared.

Table 1. MPSS Measures of Performance

PERFORMANCE MEASURE	DESCRIPTIVE PARAMETERS			
	TOTAL	PERCENTAGE	AVERAGE	STANDARD DEVIATION
CPU				
PROCESSING TIME	X			
OS TIME (IMPLICIT)	X			
IDLE TIME	X	X		
INTER-BLOCK LAG/LEAD TIME			X	X
PERIPHERAL DEVICES				
INPUT TIME			X	X
OUTPUT TIME			X	X
QUEUED DATA			X	X
QUEUE WAIT TIME			X	X
QUEUE >0		X		
COMMUNICATION BUSSES				
UTILIZATION	X			
CYCLE STEAL TIME	X			
COMMUNICATION LINES				
TIME BUSY	X			
DATA BLOCKS				
NUMBER ENTERED	X			
NUMBER PROCESSED	X			
NUMBER OF I/O ACCESSES	X			
PROCESS RESPONSE TIME			X	X
SOFTWARE MODULES				
PROCESS RESPONSE TIME			X	X
CALLS PER SECOND			X	

ROLE OF RANDOM NUMBERS

Although MPSS is primarily a deterministic model, random phenomena are introduced in four aspects: (1) sporadic arrival of external data blocks, (2) time displacement between block arrivals, (3) interrupt time between I/O accesses, and (4) selection of a particular peripheral device at each interrupt when multiple devices are assigned for I/O during execution of a software module. Random numbers between zero and one are selected sequentially from a uniform distribution. A particular sequence is specified by an input seed number.

MODEL ENTITIES, ATTRIBUTES, AND OPERATIONAL CHARACTERISTICS

Model hardware components are first discussed. Next, data block treatment is described, followed by a discussion of required software parameters. Lastly, details of the MPSS operational logic are delineated.

HARDWARE ATTRIBUTES

Central Processing Units (CPUs)

A multi-CPU configuration of diverse types may be structured. From one to ten CPUs of differing or identical types may be prescribed in a single scenario. Central processing units are characterized by processing power, in millions of instructions per second (MIPS); linkages to other CPUs, memory banks, or peripheral devices; executive control option; and input data acceptance method.

A CPU can be linked to other CPUs in either a load sharing or a master/slave arrangement. Intercomputer communication links allow data to be exchanged between the linked computers or data to be internally generated within a computer. Data are sent in accordance with the specifications of the software module currently under execution by the sending CPU. Load sharing is effected by allowing a linked idle computer to process software modules that may be executed in parallel to those being performed by the prime CPU. Such modules would have to reside at a commonly accessible location. Load sharing does not necessarily require the interchange of data blocks.

Extended Memory Banks

Extended memory banks (EMBs) provide additional core capacity for individual CPUs and a common memory repository for multiple CPUs that are ganged to it. Up to ten EMBs may be declared, each of which can be dedicated to a particular CPU or shared by several CPUs. EMBs are accessed in a non-interrupt manner in the model, i.e., a task does not relinquish the CPU when awaiting data or programs that are being transmitted from an EMB. However, the involved operating system (OS) and wait times are added to the CPU elapsed time clock. EMBs can be attached to CPUs to extend memory capacity and to share software modules and/or data block input. Inter-computer communication via an EMB causes a data block sent by a CPU to be chronologically merged with all other blocks scheduled for execution. A particular receiving computer would not be indicated to process the sent block; it would be processed by the first available computer attached to the EMB. By default, a receiving CPU is implied if the EMB is uniquely attached to it.

Peripheral Devices

Up to thirty peripheral devices representing mass storage units, interface devices, and/or display devices may be connected to the CPUs via communication lines. Devices may be attached to one or more of the CPUs. Peripheral access is governed by software specification. Each software module specifies the normalized frequency that each device is accessed, whether input or output, and the associated amount of data to be transferred. The number of actual input/output accesses (if not exactly one) to peripheral devices induced by an executing module is probabilistically timed. A negative exponential distribution is used to determine the average time between interrupts. At each interrupt a peripheral access table is referenced in accordance with module specifications. The device to be accessed and the accompanying amount of data to be transmitted are selected from this table. If, in addition, at each randomly selected I/O interrupt, more than one device can be chosen for I/O, the device is randomly selected according to its normalized frequency in the table. Added to the data transfer time (number of words times transfer rate in words per second) is a delay parameter, called the latency, for each device access.

Communication Lines

Communication lines are modeled to serve as conduits for data between CPUs, memory banks, and peripheral devices. Lines are defined by their linkage (number of devices and/or CPUs sharing a line), flow direction (one way or two way), and average transmission time per request. A CPU may also communicate with its attached memory bank and peripheral devices via a communications bus, e.g., a Digital Equipment Corporation UNIBUS. In this instance, all data must be transmitted through the bus. During any time interval, only a data transfer between two devices (the CPU and EMB are considered as devices) can be accommodated. Thus contention (including CPU instruction cycle-stealing) for the communications bus can occur. Because of this potential for contention, the model dynamically computes the time required to transmit data between pairs of devices. Accordingly, each device is assigned a priority for transferring data on the bus. Transmission time is partitioned into two components: (1) the contention-free electrical transmission time, and (2) the time awaiting access when the bus is busy. An expected value algorithm was developed to compute the wait time for the bus. The values of the parameters in this algorithm are automatically modified during the course of a model run to reflect historical usage of the bus. Contention statistics reflect the frequency that old requests are awaiting processing when new requests are entered.

DATA BLOCK SPECIFICATIONS

Data Block Types

Up to 50 district data block types can be declared. They include both external types

with specified arrival rates or internally generated types. The latter are generated upon completion of specified software modules and are transmitted either to a different CPU or back to the sending CPU for subsequent processing. All data types have a specified software sequence uniquely assigned through which each data block must be processed.

External data can be simulated to arrive at the system from external sources in three distinct modes:

1. Constant Rate: A specified number of data blocks arrive uniformly each second.

2. Periodic Rate: data blocks arrive in recurring cycles of pulses. The number of blocks in each pulse is specified. A pseudo data type, whose sole function is to periodically stimulate other types of data, indirectly permits data of arbitrary periods to be simulated.

3. Sporadic Rate: data blocks arrive in probabilistic bursts. An average rate per burst is specified; the actual number of blocks in each burst is computed via the Poisson distribution, using a random number generator and the average rate specification.

Entry of a type of data block can be defined to occur only if stimulated by the entrance of another type. Under this condiction, the stimulated data type is dormant until energized for a specified time interval by the stimulating data type.

SOFTWARE SPECIFICATIONS

Software Modules

Up to 150 software modules can be declared. A module can represent a portion of a subroutine, a complete program, or any graduated element between these extremes. A module can simulate an OS service or an element of an applications program. Modules may be used in more than one software sequence and repeated within a sequence.

Among the attributes describing each module are:

1. An identifier of the CPU, extended memory, or peripheral device of residence

2. Average number of instructions executed per call

3. Disabled (non-interruptible) or enabled (interruptible) mode indicator

4. Serial or parallel execution indicator

5. Execution priority factor

6. Average time between I/O interrupts

7. Pointer to peripheral access table (see "Peripheral Devices")

8. Indicator for inter-computer communication.

Software Sequences

For each data block type (except pseudo-periodic), a unique set of up to thirty software modules to be executed sequentially is specified. It should be noted that a simulated software module can represent any logical operation and is not restricted to a one-to-one correspondence with an actual module.

MODEL OPERATION

The model is propagated on a "data block" basis. Blocks arriving from outside the system are scheduled in bursts for entry into the simulated system. The length of a burst is prescribed by an input value, nominally one second. The resultant stack of scheduled blocks awaiting processing can be either consolidated into a common stack accessible by all CPUs, or dispersed among the CPUs in individual stacks, or a combination of both. Dispatching of block processing among the CPUs depends on the executive control of each CPU and CPU accessibility to the data block stacks and task queues. Within a stack, blocks are queued in chronological order with respect to their scheduled arrival time. CPUs under the control of a single dispatcher and linked to a common block stack and task queue will be assigned work on a "first available" basis. That is, whichever CPU is idle or becomes idle first will be assigned to process the next block or task. CPUs may also operate independently, each with its own dispatcher. By the inter-CPU data exchange feature, a particular CPU may serve as a master and the others as slaves. Off-loading of work from one CPU to another is also permissible by proper input definitions.

Primary computer performance is measured by its capability to maintain real-time operation. Accordingly, a simulated clock is associated with each CPU. The clock is updated as a function of the MIPS value. The time (t_e) required to fetch and execute

an instruction equals (1/MIPS) if the instruction resides in main memory. If it resides in an extended memory bank, the time (t_e) includes additionally the time (t_m) required to extract the instruction and any accompanying data from the memory bank plus the times (t_L) and (t_u) to transfer them across the connecting communication line and the communications bus respectively, if one is involved. Mathematically,

$$t_e = 1/MIPS + t_m + t_L + t_u.$$

When a software module comprising X instructions is executed in a disabled mode, the clock is advanced by X * t_e seconds plus appropriate I/O wait times and OS times. If disabled, the CPU is locked onto the module at commencement of execution and is not released until processing of the module is complete. The CPU idles during any period of module induced I/O. If the module, alternatively, is executing in an enabled mode, the number of instructions executed in selected time, Δt, is $\Delta t/t_e$. The elapsed time Δt represents the processing time available until an I/O interrupt occurs. The CPU is released to process any other available tasks when the currently executing enabled task requires I/O.

A simulated task queue with three levels of priority is implicitly incorporated in each CPU. Task queue operation is modeled as follows: Each accepted data block stimulates sequential execution of an assigned set of software modules. Each software module designates whether it is to be executed in a disabled mode or an enabled mode. If enabled, an execution priority is also assigned. When an enabled mode module is to be processed, a task that corresponds to the data block-software module couple is set into the task queue of the appropriate priority. The priority of the task is that assigned to the software module. Within each priority queue, tasks are scheduled and taken in a First-In/First Out (FIFO) manner. Subsequent selection is always initiated in the highest (no.1) queue. If no tasks are scheduled for this queue, or all scheduled tasks are awaiting I/O, selection is initiated in queue 2. In like manner, selection eventually reaches queue 3.

A selected task is executed until one of three events occurs: (1) all software instructions have been executed, in which case the task is removed from the queue; (2) an I/O interrupt is caused by the task, in which case the task is placed at the end of the queue and its execution cannot be restarted until the I/O operation is completed; or (3) an interrupt is caused by the arrival of an internal or external data block, in which case the task is placed at the end of the queue and its execution restarted when it has advanced to the front of the queue. A new task is created whenever an enabled mode software module is encountered during the processing of a data block. A task is not formed for disabled mode software modules. When a disabled mode module is encountered, the CPU is locked onto that module until all instructions have been executed and any I/O operations have been completed. The model allows enabled and disabled mode modules to be interspersed in any manner within the software module sequence for each type of data block.

The model is capable of simulating a distributive network by allowing component CPUs to share the data load. Two types of load sharing are permissible. Arriving data blocks can be channeled to the least busy CPU in a "distributive" cluster and/or data processing can be shared among ganged CPUs when a software module can be executed in parallel with adjacent ones in the software sequence. In a distributive cluster the incoming data blocks arrive at a common memory bank and each is taken in turn by the least busy CPU in the cluster. Software modules can also reside in the extended memory bank so that data block processing can be shared.

If load sharing is allowed and a parallel (processing) module is encountered, the CPU that received the data block attempts to off-load execution of the parallel module onto a ganged cooperative CPU that is less utilized or under-utilized. Such off-loading implies the availability of the module to the cooperative CPU. When one or more parallel modules is off-loaded, the primary CPU commences execution of remaining modules in the sequence. The least busy CPU is one that is idle or, if none in the cluster is idle, the one that will become idle soonest.

The operating system of a CPU can be treated explicitly by prescribed software modules, implicitly via fixed time values, or by a combination of both methods. If explicit modules are defined, they are placed in the software sequence along with applications modules. Alternatively, the following generic OS services can be considered via input time parameters:

1. Accept an incoming data block
2. Access main memory per request
3. Access extended memory per request
4. Access a peripheral device per request
5. Set up a disabled mode task
6. Open and close the task queue per task, and
7. Handle I/O.

The OS service time is added to the CPU clock each time the particular service is invoked.

MODEL SYNOPSIS

MPSS simulates the dynamic operation of a synthesized computer network. Elemental features of the model include:

1. A multi-CPU configuration of diverse types. From one to ten CPUs may be prescribed in a single scenario. Each modeled CPU may represent a distinct type as produced by various vendors, e.g., IBM 360 series, Univac 642, DEC PDP-11 series, etc. Two or more CPUs of the same type are also permissible.

2. Extended memory banks of the same or distinct types. A given memory bank can be a dedicated adjunct assigned to a particular CPU, or the bank may be shared by several CPUs.

3. Modeled peripheral devices may represent mass storage units and interface and display devices. Any device can be attached to one or several CPUs.

4. Component systems can be represented as associated units attached to a common communications bus, such as the PDP-11 series UNIBUS. Direct dialogue between any two units attached to the same bus is permissible.

5. Up to 50 distinct data block types can be declared. Types can be specified as being generated by a source outside of the modeled system (exogenous) or by a source within the modeled system (endogenous). Exogenous types each have a distinct rate of arriving at the modeled system. All types have a specified software sequence uniquely assigned through which each data block must be processed.

6. Up to 150 software modules can be declared. A module can represent a portion of a subroutine, a complete program, or any graduated element between these extremes. A module can simulate an OS service or an element of an applications program. Modules may be used in more than one software sequence and repeated within a sequence.

7. Data load distribution. Arriving external and generated internal data blocks may be distributed among specified CPUs as a function of individual CPU busyness.

8. Capability to trace each data block through the system.

9. Simulation of task queue operation and task dispatching.

MODEL CALIBRATION

INTRODUCTION

A calibration study was pursued to verify the model results against independently determined performance measurements on an existing operational system. With the satisfactory agreement (within experimental and statistical uncertainties) obtained, increased confidence in model fidelity was achieved. The Monitor and Control Display System (MACDS) at the NASA Goddard Space Flight Center was chosen for the calibration study for the following reasons:

1. It is an operational interactive system incorporating a variety of hardware and software resources, and it handles a variety of data traffic in normal operations.

2. Performance measures could be straightforwardly obtained, as the software designers and system operators were in-house and could create a controlled experimental environment to measure performance parameters compatible with the model output.

3. It was a system where current interest existed in monitoring performance versus anticipated data load variations, and in evaluating the expected performance of a MACDS backup system with reconfigured hardware.

MACDS consists of a PDP-11/40 switching computer and a linked PDP-11/40 display computer, to which 46 digital television displays and associated keyboards are currently connected. The display data are transmitted to the screens via a pair of Data Disc vector and character generators (VCGs) with associated refresh disks. An additional RK11 disk used for data base displays (normally multi-paged) is included in the data system. A schematic diagram of the system is shown in Figure 1.

Input from a UNIVAC 642B and two IBM 360 hosts is routed via the switching computer to the display computer for processing, and subsequently via peripherals for display or to data base storage. In addition, a variety of operator input is possible via typed keyboard commands from each of the 46 keyboards.

EXPERIMENT DETAILS

The calibration study concentrated on three types of data block processing which tested the basic capabilities of the model and were amenable to straightforward experimental control.

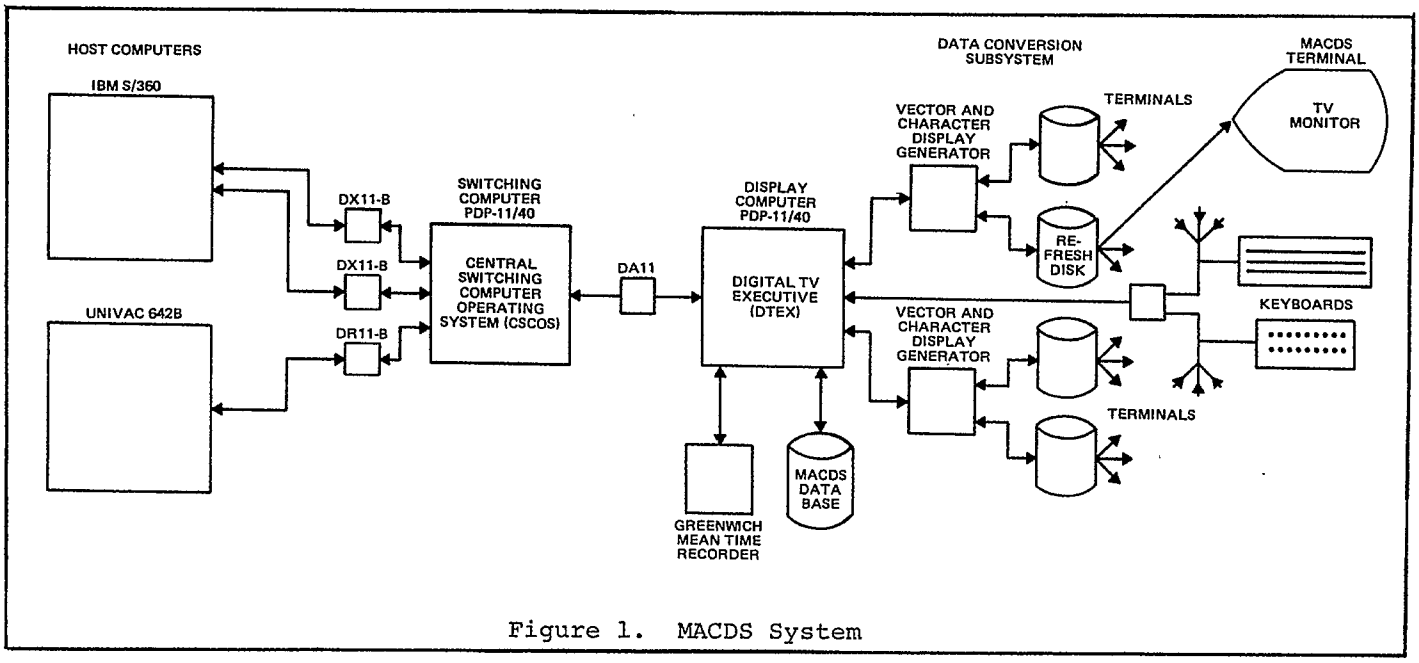

Figure 1. MACDS System

In the first type of processing, typed character input (with subsequent echoing to a display screen) was experimentally controlled by using the keyboard repeat key which generated a steady stream of a known number of characters per second (approximately ten) from each keyboard in the experiment. A primary reason for looking at the character echoing subsystem was the 1/30th second disc rotation time required to transfer each character data block to the display screen. An entire disk revolution is required for the transfer of the data bits for each character because of the existing unbuffered software for keyboard character echoes sent via the vector and character generators. Data for display of an entire screen, if buffered, would also require only 1/30th second (one disc rotation). In addition to single character echoing, this subsystem processed a pulse of 46 data blocks (one for each terminal) every 60 seconds to display the Greenwich Mean Time (GMT). Again, the inclusion or exclusion of the GMT data blocks was easily controlled experimentally.

In the second type of data block processing, scenarios simulating the sending of display data from a host computer were exercised. Variations were made in data rates and in the number of display screens to which the above display data blocks were sent. Model results were again compared to controlled MACDS results. In the third type of processing, system response to display data sent to the RK11 disk data base was simulated via the model and compared to prototypal system response in counterpart experiments. As a final calibration test, all three of the above data types were combined simultaneously at nominal data rates and comparisons made between corresponding simulated and prototypal system performance.

To facilitate control of the prototypal experiments, a special IBM 360 host computer program was written to generate display and add-to-data-base blocks at known rates. Character echo control was achieved by locking out the MACDS keyboards until command activation, and by subsequent use of the repeat keys. The display computer software kept track of processing times to 1/10000th second accuracy, and special subroutines were written for the display computer to tabulate statistics on the system performance.

Two measures of performance were compared. The percentage of CPU idle time during the course of each experiment was obtained. The wall-clock time required to totally process each display and add-to-data-base block was measured. The times and their squares were summed to obtain the mean data block response times and associated standard deviations. Peripheral device performance is implicitly imbedded in these statistics.

EXPERIMENT VERSUS MPSS RESULTS

The following ten distinct experiments were performed:

1. 20 character echoes per second to one VCG, and a GMT update every 60 seconds to 23 terminals on each of the two VCGs.

2. 30 character echoes per second to one VCG with no GMT updates.

3. 20 character echos per second to one VCG with no GMT updates.

4. Five host-send-to-terminal data blocks per second to one VCG.

5. One forced message data block every four seconds for display to twenty terminals, all connected to one VCG.

6. Two add-to-data base updates per second.

7. Four add-to-data-base updates per second.

8. Five host-send-to-terminal data blocks per second plus one forced message per four seconds to twenty terminals (both to one VCG), plus two add-to-data-base updates per second, plus a GMT update every sixty seconds to both VCGs.

9. The same as 8 above, plus seven character echoes per second to one VCG.

10. Same as 9 above, except four add-to-data-base updates per second instead of two per second.

The experiments were executed pairwise on the MACDS system and via MPSS. The experimental results are summarized in Tables 2 through 5. In all of the experiments, the percentage of time the CPU was idle was compared. These results are summarized in Figure 2. The worst difference occurred in experiment eight in which the model showed 7% more idle time than the display computer (86.51% vs. 79.57%). In experiments one through three the difference did not exceed 0.5%. For these three experiments the correlation coefficient was .9995.

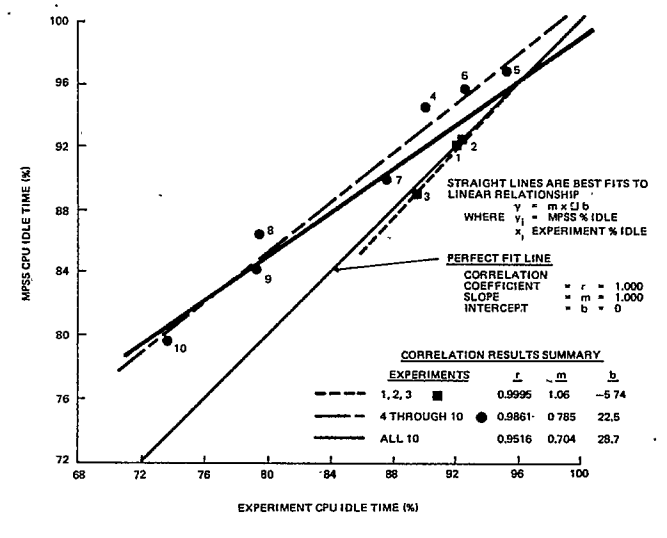

Figure 2. % CPU Idle - MPSS and Experiment Comparisons

In experiments four through ten, the average difference in CPU idle time percentage

Table 2. Percentage of Time CPU Was Idle

EXPERIMENT #	MACDS	MPSS
1	92.34	92.21
2	92.47	92.24
3	89.64	89.29
4	90.13	94.61
5	95.06	96.94
6	92.72	95.06
7	87.68	89.98
8	79.57	86.51
9	79.21	84.11
10	73.74	79.91

CORRELATION COEFFICIENT = 0.9516

Table 3. Average Host-Send-to-Terminal Data Block Response Time (msec)

EXPERIMENT #	MACDS	MPSS
5	80.1	87.3
8	543.7	477.8
9	655.5	860.3
10	638.2	875.7

CORRELATION COEFFICIENT = 0.9439

Table 4. Average Add-to-Data Base Data Block Response Time (msec)

EXPERIMENT #	MACDS	MPSS
6	228.7	250.4
7	260.5	269.0
8	241.6	256.0
9	243.6	255.8
10	267.2	271.9

CORRELATION COEFFICIENT = 0.9886

Table 5. Average Forced Message Data Block Response Time (msec)

EXPERIMENT #	MACDS	MPSS
5	646.3	1070
9	722.8	1282
10	720.6	1317

CORRELATION COEFFICIENT = 0.9878

was 4%. The correlation coefficient in this case was .9861. The overall value of the correlation coefficient for all ten experiments was .9516. This lower value occurs because the display computer showed consistently higher idle time in the first three experiments and consistently lower idle time in experiments four through ten.

Model and MACDS data block response times were compared for the host-send-to-terminal, forced message, and add-to-data-base blocks (this statistic was not available from the display computer for character echoes). Figures 3 and 4 summarize the results, including correlation coefficients. The add-to-data-base updates showed the least discrepancy; the model showed an average 5% longer response time. High positive correlation was indicated by a correlation coefficent value of .9886. The host-send-to-terminal response times had pairwise varying and relatively wide (70% - 80% of the mean value) standard deviations. In all cases, however, the model and MACDS mean response times were within each other's standard deviation. In three cases the model generated a longer response time than MACDS, and in one case a shorter time. The minimum discrepancy was 10% of the MACDS mean value and the maximum difference was 37%. High correlation was nevertheless affirmed with a correlation coefficient value of .9539. The model has been determined to be incapable of simulating forced message processing in the identical manner as MACDS because of differing queuing logic for I/O. Total response time to get the message to 20 terminals is about the same but average response time per terminal is consistently higher by 65% to 80% in the model. The relative trend is, accordingly, consistent with a correlation coefficient of .9878.

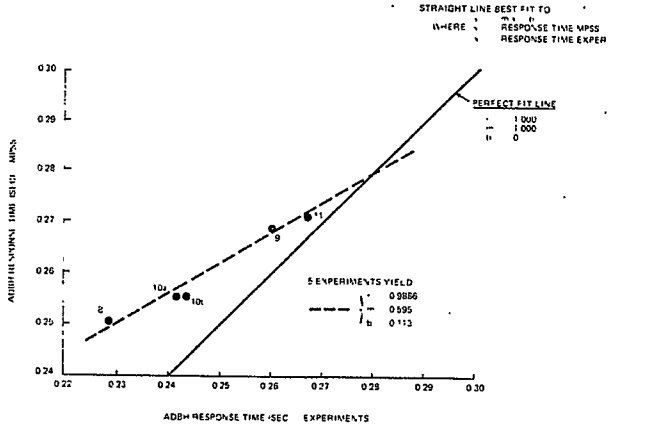

Figure 3. Add-to-Data-Base (ADBH) Response Times - MPSS vs. Experiments

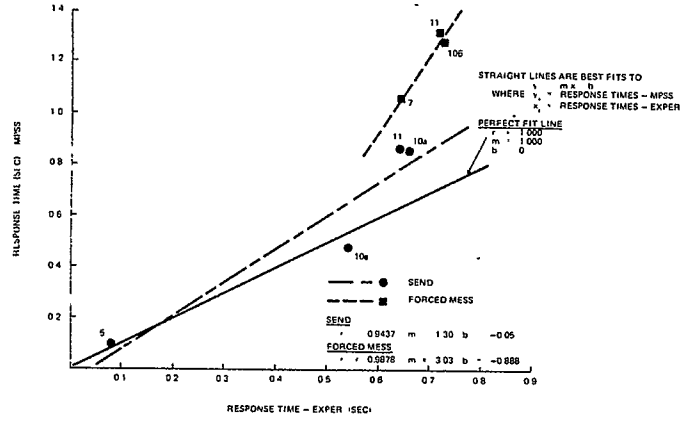

Figure 4. Send and Forced Message Response Time - MPSS vs. Experiments

INTERPRETATION OF RESULTS

On a relative basis, MPSS is shown to simulate the display computer ensemble of MACDS in a consistent manner. On an absolute basis, the model shows good fidelity in simulating CPU utilization and in generating average data block response times except in the case of forced messages. The accuracy of host-send-to-terminal message response times is uncertain because of the wide standard deviations generated by both the model and MACDS. In addition the model resources required vary linearly with data load whereas the MACDS system exhibits the following non-linear dependencies on data load and backlog:

1. As data load increases the computer time required for buffer allocation increases significantly due to the time required to search through memory for adequate buffer space. Likewise, queue processing times increase with increasing queue lengths.

2. As backlogs build up in a particular data type (e.g., character echoes) data is merged within existing buffers and both the number of buffer allocations required and the overhead for process initiation and termination decreases significantly. This decreases the computer time required to process each data block.

   The first non-linear effect is most significant in the mixed-data scenarios, whereas the latter effect predominates in high data rates of a given type, e.g., 30 character echoes per second. The model, in contrast, requires identical system resources for each data block, independent of data load and backlogging.

CONCLUSION

The model was developed primarily as a tool to evaluate the performance of existing computer systems under loading or hardware variations and to give projected performance measures of proposed alternative systems. It is capable of simulating a real-time multi-CPU interactive system having main or extended memories, multiple peripheral accesses per data block, a data block load sharing option among the CPUs, and interruptible or non-interruptible software module processing modes. The positive correlation between simulated MPSS results and MACDS measurements yields a high level of confidence in the fidelity of the model.