General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)
SYSTEM ANALYSIS
FOR THE
HUNTSVILLE OPERATIONAL SUPPORT CENTER
DISTRIBUTED COMPUTER SYSTEM

(NASA-CR-170840) SYSTEM ANALYSIS FOR THE
HUNTSVILLE OPERATIONAL SUPPORT CENTER
DISTRIBUTED COMPUTER SYSTEM Annual Report,
May 1982 - Jun. 1983 (Mississippi State
Univ., Mississippi State.) 106 p

ANNUAL REPORT
MSU-EIRS-EE-83-6
May 1982 - June 1983

Submitted by:
F. M. Ingels, Principal Investigator

Mississippi State University
Electrical Engineering Department
Mississippi State, MS 39762
(601) 325-3912

Submitted to:
NASA/MSFC, Alabama
Technical Monitor: Frank Emmens, EB 32.
(205) 453-4629

NAS8-34906
July, 1983
SYSTEM ANALYSIS
FOR THE
HUNTSVILLE OPERATIONAL SUPPORT CENTER
DISTRIBUTED COMPUTER SYSTEM

ANNUAL REPORT
MSSU-EIRS-EE-83-6
May 1982 - June 1983

Submitted by:
F. M. Ingels, Principal Investigator

Mississippi State University
Electrical Engineering Department
Mississippi State, MS 39762
(601) 325-3912

Submitted to:
NASA/MSFC, Alabama
Technical Monitor: Frank Emmens, EB 32
(205) 453-4629

NAS8-34906

July, 1983
The Huntsville Operations Support Center (HOSC) is a distributed computer system used to provide real-time data acquisition, analysis, and display during NASA space missions and to perform simulation and study activities during non-mission times. The primary purpose of this research is to provide a HOSC system simulation model that may be used to investigate the effects of various HOSC system configurations. Such a model would be valuable in planning the future growth of HOSC and in ascertaining the effects of data rate variations, update table broadcasting and smart display terminal data requirements on the HOSC HYPER channel network system.

A simulation model was developed and programmed in three languages: BASIC, PASCAL, and SLAM. Two of the programs are included in this report, the BASIC and the PASCAL language programs. SLAM is not supported by NASA/MSFC facilities and hence was not included. The statistical comparison of simulations of the same HOSC system configurations are in good agreement and are in agreement with the operational statistics of HOSC that were obtained.

Three variations of the most recent HOSC configuration have been run and some conclusions drawn as to the system performance.
under these variations. Section 3.4 discusses these results and conclusions.
## LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VOSC Data Transfers</td>
</tr>
<tr>
<td>2</td>
<td>Summary of DEC VAX 11/780 Data I/O Characteristics</td>
</tr>
<tr>
<td>3</td>
<td>Summary of Perkin-Elmer 3244 I/O Characteristics</td>
</tr>
<tr>
<td>4</td>
<td>Summary of Perkin Elmer 8/32 I/O Characteristics</td>
</tr>
<tr>
<td>5</td>
<td>A400 Adapter Function Descriptions</td>
</tr>
<tr>
<td>6</td>
<td>Comparisons of BASIC, PASCAL and SLAM Programs</td>
</tr>
<tr>
<td>7</td>
<td>Comparison of Four Simulation Runs</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Original ROSC HYPER channel interconnection</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Proposed HYPER channel interconnection</td>
<td>8</td>
</tr>
<tr>
<td>2a</td>
<td>Proposed HYPER channel Network Configuration Rev. 1</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Block Diagram of VAX 11/780 Computer</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>Basic Bus Configuration of VAX 11/780</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>Block Diagram of PE 3244 Computer</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>NSC HYPER Channel Adapter Block Diagram</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>Host to A400 Adapter Data Exchange</td>
<td>31</td>
</tr>
<tr>
<td>8</td>
<td>Trunk Contention Delay Algorithm</td>
<td>35</td>
</tr>
<tr>
<td>9</td>
<td>ROSC Simulation Configuration</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>System Configuration Parameters for 1333 Data Transfer Simulation</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>System Configuration Parameters for 1333 Data Transfers</td>
<td>41</td>
</tr>
<tr>
<td>12</td>
<td>Typical Printout When Data Transfer Collision Occurs</td>
<td>43</td>
</tr>
<tr>
<td>13</td>
<td>Typical Printout When Data Transfer Collision Occurs (Continued)</td>
<td>44</td>
</tr>
<tr>
<td>14</td>
<td>Typical Printout for Data File Dump and intersource Data Transfers</td>
<td>45</td>
</tr>
<tr>
<td>15</td>
<td>Statistical Summary of 1333 Data Transfers Simulation</td>
<td>46</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 HOSC System Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Scope of Report</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Conclusions</td>
<td>3</td>
</tr>
<tr>
<td>2.0 HOSC SYSTEM DETAILS</td>
<td>10</td>
</tr>
<tr>
<td>2.1 HOSC System Activities</td>
<td>14</td>
</tr>
<tr>
<td>2.1.1 Total POGC Preplanning</td>
<td>10</td>
</tr>
<tr>
<td>2.1.2 ECIO Data Stream</td>
<td>10</td>
</tr>
<tr>
<td>2.1.3 Main Engine Data</td>
<td>13</td>
</tr>
<tr>
<td>2.1.4 OI Data Stream</td>
<td>13</td>
</tr>
<tr>
<td>2.1.5 Engineering Display Charges</td>
<td>14</td>
</tr>
<tr>
<td>2.1.6 Proposed Activities</td>
<td>14</td>
</tr>
<tr>
<td>2.2 HOSC System Components</td>
<td>15</td>
</tr>
<tr>
<td>2.2.1 Computer Resources</td>
<td>23</td>
</tr>
<tr>
<td>2.2.1.1 DEC VAX 11/780</td>
<td>15</td>
</tr>
<tr>
<td>2.2.1.2 Perkin Elmer 3244</td>
<td>23</td>
</tr>
<tr>
<td>2.2.1.3 Perkin Elmer 8/32c</td>
<td>25</td>
</tr>
<tr>
<td>2.2.2 NSC HYPER Channel</td>
<td>25</td>
</tr>
<tr>
<td>2.2.3 Other HOSC Components</td>
<td>36</td>
</tr>
<tr>
<td>3.0 HOSC ANALYSIS RESULTS</td>
<td>37</td>
</tr>
<tr>
<td>3.1 Typical Basic Algorithm Information</td>
<td>38</td>
</tr>
<tr>
<td>Printouts</td>
<td></td>
</tr>
<tr>
<td>3.2 Results of A Simulation Run</td>
<td>42</td>
</tr>
<tr>
<td>3.3 Comparison of BASIC, PASCAL and SLAM</td>
<td>47</td>
</tr>
<tr>
<td>Programs</td>
<td></td>
</tr>
<tr>
<td>3.4 Conclusions</td>
<td>49</td>
</tr>
<tr>
<td>4.0 LITERATURE SURVEY</td>
<td>54</td>
</tr>
<tr>
<td>4.1 References</td>
<td>54</td>
</tr>
<tr>
<td>4.2 Bibliography</td>
<td>55</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1 Publications</td>
<td>55</td>
</tr>
<tr>
<td>4.2.2 Reference Manuals (by Manufacturers)</td>
<td>56</td>
</tr>
<tr>
<td>APPENDIX I BASIC SIMULATION ALGORITHM PROGRAM</td>
<td>57</td>
</tr>
<tr>
<td>APPENDIX II PASCAL SIMULATION ALGORITHM PROGRAM</td>
<td>73</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

1.1 HOSC System Overview

Marshall Space Flight Center (MSFC), Huntsville, Alabama, has implemented the Huntsville Operations Support Center (HOSC) to provide real time data acquisition, analysis, and display during NASA space missions. The HOSC is a distributed computer system composed of a network of large minicomputers and various peripheral equipment. Primarily designed to provide support for the Space Shuttle, Space Telescope, and Space Laboratory missions, the HOSC has the inherent flexibility to be expanded to meet the needs of future missions as well as providing MSFC with a large computer resource that can be used to support several non-mission activities.

The HOSC facility has been structured to include five large minicomputers and various peripheral equipment. The current network computers are each semi-dedicated to specific mission tasks (e.g. Space Shuttle Main Engine Data Analysis) and include three Perkin Elmer 3244 computers, a Perkin Elmer 8/32c computer, two DEC VAX 11/780 computers and a DEC 11/24 computer. An important role of the Perkin Elmer computers is acting as real time data receivers for mission data arriving via satellite and direct ground links from the Kennedy Space Center Firing Room at Cape Canaveral FL. These computers also act as a gateway to the network for the data which is needed by other mission activities supported by the other computers and peripherals. Peripheral equipment in the system includes two twelve channel Genisco Digital Television (D/TV's), strip recorders, and numerous unintelligent data terminals.
Foreseeable future expansion will include at least five more mini-computers, many more D/TV displays (possibly to 50), several more strip recorders, and intelligent data terminals.

The HOSC currently provides support for MSFC non-mission activities such as the Total POCC Preplanning Activity with future expansion providing data management resources for other non-mission activities. These activities might include the DEC IGDS (interactive graphics) and XEROX SIGMA (text processing) operations. All of these activities would be permitted use of the network resources through the Network Systems Corporation HYPER channel broadband local area network.

1.2 Scope of Report

In order to achieve the flexibility and efficiency needed by the HOSC, an analysis of the present system has been performed. This analysis coupled with projected system growth will insure that the HOSC remains a viable computing resource for MSFC. This report contains a summary of the baseline data gathered to begin the analysis of the HOSC computer network, Section 2.0, results of the analysis, Section 3.0, and a literature/bibliography Section 4.0. The report describes in detail some of the network components and also makes first iteration recommendations concerning network operations. This document should not be considered an end item since work still remains to be done in completely characterizing all the subtleties of the HOSC system.
1.3 Conclusion

From the work done thus far in the program, several conclusions and recommendations can be made.

A. Proposed IGDS/SIGMA Interface With HOSC

Network Systems Corporation does not currently produce hardware for the HYPER channel to XEROX processor interface. Consequently, a great amount of effort would be required to interface the SIGMA system directly with the HOSC HYPER channel. A possible solution might be to interface the XEROX SIGMA to the network through a HYPER channel supported processor such as another DEC VAX. Feasibility of the VAX/XEROX interface has not been explored and may also present problems. A definite possibility to solve this problem is to develop a suitable software/hardware approach.

The DEC IGDS system interfaces with the HYPER channel and will present no obvious problems since the PDP-11 processor interface adapters are currently marketed by Network Systems Corporation.

B. CSO/HOSC Link Via HOSC HYPER Channel Adapter 4 For OF Data Exchange

The current plan is to interface CSO with HOSC using a separate trunk of adapter 4. By connecting the two installations with a separate trunk, CSO will be disallowed easy and immediate access to the HOSC resources on the HYPER channel. Because of the HYPER channel adapter design, direct trunk to trunk exchange of data is not possible. For trunk to trunk transfers, data from the initiating trunk must be channeled
through a processor on the common adapter and retransmitted by
the processor over the other trunk. It however, it is de-
sireable to prevent CSO from easy access to the total HOSC
resources, then the use of separate trunk is a good approach.

C. Summary of Analysis Activity

Progress on the analysis of HOSC has so far been steady
but somewhat slow due to the difficulty in obtaining some needed
baseline data. Below is a summary of the documentation ac-
cumulated to conduct the analysis effort.

• Perkin Elmer Corporation
  3240 User's Manual 29-685
  3240 Memory System Manual 29-688
  8/32 User's Manual 29-394
  8/32 Memory System Manual 29-428
  (These manuals must contain the actual HOSC Computer
   internal DMA to I/O setup.)

• Network Systems Corporation
  PI40 Peripheral Interface Manual
  (Perkin Elmer I/O Bus Interface)
  PI10 PI Manual or PI11 PI Manual
  (Dependent on configuration of PDP-11 IGDS
   system: PI10 for DR11-B general purpose
eedle memory access or PI11 for DR70
   MASSBUS interface.)

• Marshall Space Flight Center
  Completed system computer data rate flows.
D. **Effects of Data File Dumps**

It is desired to make large data file transfers on a periodic basis to refresh the data display terminals data base. This type of activity can create a log/jam effect on the most active data sources if the number of data bytes to be transferred are large enough to create waiting times. The basic relationship involves a tradeoff between the amount of storage of data by the data sources, their rate of data accumulation and the time required to transfer the data files.

This problem is discussed in Section 3.3 and 3.4 in detail.
2.0 HOSC SYSTEM DETAILS

The primary purpose of the Huntsville Operation Support Center is providing MSFC engineers with a near real-time summary of vital information describing the operational status of certain components of the Space Shuttle during pre-launch and launch activities. This information allows MSFC engineers and contractor personnel to act in a support capacity to mission personnel at Kennedy Space Center (KSC), Cape Canaveral, Florida, and also Johnson Space Center (JSC), Houston, TX. MSFC support is provided by teams responsible for the Space Shuttle Main Engine (SSME), External Tanks (ET), Solid Rocket Boosters (SRB), Main Propulsion System (MPS) and the Range Safety System (RSS). Additional mission support is provided for various mission activities and programs that are the responsibility of MSFC personnel.

During powered flight, the HOSC will receive only data which is in the LPS (Launch Processing System) at KSC. The Shuttle support team will be in the HOSC during this phase of the mission and will be the point of contact with the JSC Mission Evaluation Room (MER) for problem discussion and resolution as required and will be on call during orbital operations. The Space Lab and experiment support team will be located in the HOSC during orbital operations when applicable.

Following completion of the active Shuttle vehicle support activities, data is recalled as required for more detailed analysis, and initial preparation is made to provide support to postflight evaluation.

The HOSC is located in the west end of A-wing, building 4663 on Martin, Road, MSFC. Figures 1 and 2 show the functional components of the HOSC system and gives each component a referencing number that will be used in describing the system activities.
Figure 1. Original HOSC HYPER Channel Network Configuration.
Figure 2. Proposed HYPER Channel Network Configuration
Figure 2a. Proposed HYPER Channel Network Configuration Rev 1.
(MIPS Primary VAX 4 connected to Adapter 5.)
2.1 HOSC System Activities

In addition to the mission activities, the HOSC also provides support to several non-mission activities at MSFC. Details of all the HOSC activities are described below and summarized in Table 1.

2.1.1 Total POCC Preplanning

The POCC activity is an ongoing simulation activity for which the HOSC lends computer resources. This activity is in no way keyed to the real time mission activities and must be viewed as a continuous daily activity.

The POCC activity's impact on the HYPER channel network is basically that of continuous data transfers between the MIPS Primary Computer (VAX4, A400 Adapter 2) and the MIPS Backup Computer (VAX1, A400 Adapter 5). During each 24 hour period 150,000 512-byte blocks of data are transferred. Six times each day, an 8344 byte block is transferred (50,000 bytes cumulative). The remaining 100,000 512-byte blocks are transmitted randomly, but on an evenly distributed basis, throughout the day.

2.1.2 ECIO Data Stream

The POCC activity generates a continual 51.2 kilobit/second data stream known as the Experimental Computer Input/Output (ECIO) data stream. This data stream is ongoing and concurrent with the POCC activity. Data is transferred from MIPS Backup (VAX1, A400 Adapter 5) to the Spacelab 8/32 (PE 8/32c, A400 Adapter 1).
TABLE 1. HOSC DATA TRANSFERS

I. ROUTINE DAILY ACTIVITIES (Launch Independent)

A. **Total POCC Preplanning Activity**
   
   Resources involved: MIPS Primary (VAX4, Adapter 2)  
   MIPS Backup (VAX1, Adapter 5)  
   
   Quantity of data: 150,000 512-byte blocks daily

B. **ECIO Data Stream** (Generated by POCC)
   
   Resources involved: MIPS Backup (VAX1, Adapter 5)  
   Spacelab 8/32 (PE 8/32c, Adapter 1)  
   
   Quantity of data: 51.2 K bits/second concurrent with POCC.

C. **IGDS/SIGMA Activity** (Proposed)
   
   Resources Involved: DEC IGDS and XEROX SIGMA and communication with other resources as needed.
   
   Quantity of data: TBD

II. LAUNCH DAY ACTIVITIES:

A. **Routine Daily Activities** (See Above)

B. **Main Engine Data**
   
   Resources Involved: STS Primary (PE 3244, Adapter 1)  
   MIPS Backup (VAX1, Adapter 5)  
   
   Quantity of Data: 50 K bit/second stream (T-8 hours to T+12 minutes)

C. **OI Data Stream**
   
   Resources Involved: FEB SSME (PE 3244, Adapter 4)  
   CSO Computers (Adapter 4)  
   STS Primary (PE 3244, Adapter 1)  
   MIPS Backup (VAX1, Adapter 5)  
   
   Quantity of Data: 128 K bit/second (T-9 sec to T+12 minutes)  
   into FEP and then to CSO. 40% will also be transferred to STS and MIPS.
TABLE 1. HOSC DATA TRANSFERS (Continued)

D. Engineering Display Changes

Resources Involved: STS Primary (PE 3244, Adapter 1)
STS Backup (PE 8/32 Adapter 1)
Spacelab 8/32 (PE 8/32, Adapter 1)

Quantity of Data: Insignificant
2.1.3 Main Engine Data

Space Shuttle Main Engine data is collected and disseminated at the HOSC during a launch day activity only. Data is funneled through the HYPER channel network to MIPS Backup (VAX1, A400 Adapter 5) via STS Primary (PE 3244, A400 Adapter 1). STS Primary accepts a continual 50 kilobit per second data stream directly from the KSC firing room from 9 seconds before launch to 12 minutes after launch (MECO). Approximately 24 percent of this 50 Kb/s stream (12 Kb/s) is transferred over the network to MIPS Backup.

2.1.4 O1 Data Stream

The O1 data stream is a 128 kilobit per second data stream arriving at FEP SSME (PE 3244, A400 Adapter 4) on launch day only (t-9 seconds to T+12 minutes). This data will have a much greater future impact on the network than it does currently. The SSME computer acts as a front end processor for accepting this data stream from Goddard Space Flight Center and then writes the received data directly onto a magnetic tape for later transport to CS0. Later in the program this data will be shipped in its entirety over a separate HYPER channel trunk attached to A400 Adapter 4 to CS0. Additionally, about 40 percent of the data stream will be shipped over the HOSC network to supply and supplant the data currently being transferred by the Main Engine Data Activity.

2.1.5 Engineering Display Changes

This activity adds almost insignificantly to the total HYPER channel trunk traffic. The activity involves a transfer from STS Primary to STS Backup and the Spacelab 8/32 (PE3244 to two PE8/32's,
A400 Adapter, 1 only) of the name of each engineering console display format that is changed during the pre-launch and launch activities (T-9 seconds to T+12 minutes). This activity will be ignored in the HOSC system analysis due to its negligible contribution to total HYPER channel system traffic.

2.1.6 Proposed Activities

The most immediate proposed expansion of the HOSC network would allow two other non-mission activities access to the resources of the HOSC network. This activity would specify an additional A400 Adapter to allow resource sharing with the XEROX SIGMA system and the DEC PDP-11 IDGS system. Direct interface with the A400 is available for the PDP-11 but not for the XEROX system. A possible solution to allow the XEROX system access to the network through the A400 might be to use a compatible computer such as a DEC VAX 11/780 as a front end processor for the XEROX system. This activity is incompletely specified and will not affect the immediate analysis of the HOSC system.

2.2 HOSC System Components

The heart of the HOSC system is the Network Systems Corporation HYPER channel. The HYPER channel is a high speed digital communications facility that is used for interconnection of computer resources in a computing installation. The following sections describe the computer resources of the HOSC and how they are interconnected using the HYPER channel.
2.2.1 Computer Resources

2.2.1.1 DEC VAX 11/780

The HOSC makes use of Digital Equipment Corporation's VAX 11/780 computer as computational devices. The system currently includes two VAX computers (VAX1 and VAX4) designated as MIPS Primary and MIPS Backup. Future expansion will add two other VAX computers (VAX2 and VAX3) designated as Software Development and Space Telescope.

VAX computers support a 32-bit work architecture that is designed to aid in system throughput. Data transfers are accomplished via a 32-bit high speed data structure. This structure ties together the central processor, main memory, the UNIBUS subsystem, the MASS BUS subsystem and the DR780 high speed direct memory access subsystem. The 32-bit word architecture of the VAX establishes a virtual address space of 4.3 billion bytes of user addressable memory. A conceptual diagram of the VAX 11/780 bus structure is shown in Figures 3 and 4.

The Synchronous Backplane Interface (SBI) is the data path that links the central processor, the memory subsystem and the hardware adapters provided for the UNIBUS and MASSBUS. When interfaced to the SBI, the memory subsystem, the central processor, and the I/O controllers are known as NEXUSs.

All NEXUSs receive every SBI transfer. Logic in each NEXUS determines whether the NEXUS is the designated receiver for this transfer. Data transfers can occur from

- CPU to memory subsystem
- I/O controller to memory subsystem
- CPU to I/O controllers.
Figure 3. Block Diagram of VAX 11/780 Computer
Figure 4. Basic Bus Configuration of VAX 11/780
The maximum, aggregate data transfer rate on the SBI is 13.3 megabytes per second which can be derived from the following information.

- 200 Nanoseconds/cycle = 5 million cycles/second
- Each cycle can carry an address (memory request) or for byte of data
- Thus, one cycle is used to request eight bytes of data (to be read or written), and two cycles are used to carry data (at four bytes/cycle).
- Five million cycles/second * 4 bytes/cycle = 20 million bytes/second
- 20 * 2/3 (1 of every 3 cycles is an address) = 13.3 million byte/second.

The memory controller is the NEXUS used to interface the memory subsystem to the SBI. A system may have more than one memory controller as in the case of a two controller interleaved memory configuration.

The UNIBUS (UBUS) is a high speed asynchronous data system that allows communication between peripheral hardware and the VAX 11/780. The VAX 11/780 is capable of supporting 4 UBUS subsystems; one is standard with three more optional. The UBUS is connected to the SBI through a UBUS adapter (UBA) which performs priority arbitration among the devices on the UBUS. The primary functions of the UBA are to provide:
(1) Access to UBUS address space from the SBI
(2) Mapping of UBUS address to SBI addresses for UBUS
direct memory access (DMA) transfers to system memory.
(3) Data transfer paths for UNIBUS device access to random
SBI memory addresses and high speed transfer for devices
that transfer to consecutive increasing address.
(4) UNIBUS interrupt fielding
(5) UNIBUS priority arbitration.

All of these services are completely transparent to UBUS users.

The address mapping function is necessary because the UBUS has
only 18 data lines thus providing an apparent memory addressing
capability of $2^{18}$ or 200 kilobytes. The UBA, however, provides the
capability of mapping the UNIBUS addresses into SBI addresses so
that the full memory of the system can be accessed. (Full system
memory is 16 array boards of 256 kilobytes each for a total of 4
megabytes.)

The UBA accepts either of two forms of input from the UBUS:
hardware generated interrupt or direct memory access transfer. Each
device connected to the UBUS uses one of five priority levels for
requesting bus service. The Non-Processor Request (NPR) is used when
the device requests a direct access transfer to memory or some other
device and does not require processor intervention. A Bus Request
(BR) is used when the device wishes to interrupt the BPU for service.
Such service might be a CPU directed data transfer or the informing
of some error condition that exists at the peripheral. The NPR has
the highest priority with four levels of BR following (BR7- BR4).
Since there are only five priority levels and more than one device may be connected to a specific request level, if more than one device makes the same request, the device that is electrically closest to the UBS receives higher priority.

The Non Processor Request for direct memory access is a very important feature of the UBUS subsystem. These DMA transfers can be divided into two groups: random access of noncontiguous addresses and sequential access of sequentially increasing address. For random access, each UBUS transfer is made through the Direct Data Path (DDP, one per UNIBUS) and is mapped into an SBI transfer. This procedure allows only one word of data to be transferred during a single SBI cycle. For devices capable of requesting sequential access services, use is made of Buffered Data Path (BDP). Each UNIBUS provides 15 such BDPs. The BDP stores the data so that four UBUS transfers are performed for each SBI transfer.

The DDP must be used by devices not transferring to consecutive increasing addresses or by devices that mix read and write functions. The maximum throughput via the DDP is about 425 kilo words per second for write operations and 316 kilo words per second for each read operation. These rates will decrease as other SBI activity increases.4

Maximum published throughput via the BDP is about 695 kilo words per second for both read and write operations but actual expected throughput rates are only 1.5 mega bits per second. This rate will also decrease as other SBI activity increases.1,5 BDP transfers are
restricted to block transfers where a block is defined as equal to or greater than one byte. All transfers within the block must be to consecutive and increasing addresses and all transfers must be of the same function type (Read or Write).

The MASSBUS subsystem and the DR780 high performance 32-bit parallel interface will not be described in this report since an understanding of their functional characteristics is not needed to determine their relative impacts on the HYPER channel network. The influence of both may be felt indirectly, however, since activity on the MASSBUS or DR780 will translate to SBI activity which will affect DDP and BDP transfer rates as described previously.4

Likewise, the VAX CPU will not be described in detail but several comments may be made about the CPU's effects on throughput. The CPU represents the most intensive traffic load on the memory subsystem and hence on the SBI. Obviously if the processor is engaged in computing, it will request data much more often than it will write data. Fortunately the large memory cache (8 kilo bytes) available to the CPU reduces the SBI traffic load considerably.

In terms of the SBI traffic, impact on the processor's speed, published figures4 indicate that in a system with two memory controllers, the processor will be slowed about four percent per averaged megabyte per second of I/O traffic. The impact of a single memory controller is to slow the processor by a factor varying from two to four. Table 2 summarizes the DEC VAX 11/780 I/O characteristics.
### Table 2: Summary of DEC VAX 11/780 Data I/O Characteristics

**Processor**

| Memory | 32 bit words |

**Main Memory**

<table>
<thead>
<tr>
<th>Virtual Address Space</th>
<th>4.3 billion bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle Time</td>
<td>800 nanoseconds per 64-bit read</td>
</tr>
<tr>
<td></td>
<td>1400 nanoseconds per 64-bit write</td>
</tr>
</tbody>
</table>

**I/O Unibus Adapter**

<table>
<thead>
<tr>
<th>Maximum Unibus I/O Rate</th>
<th>1.5 Mb/sec through buffered data paths.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffered Data Path</td>
<td>15 total, 8 byte buffer in each</td>
</tr>
<tr>
<td></td>
<td>695 K words/second for read operations</td>
</tr>
<tr>
<td></td>
<td>695 K words/second for write operations</td>
</tr>
<tr>
<td></td>
<td>Used for fast DMA transfers</td>
</tr>
<tr>
<td>*Direct Data Path</td>
<td>425 K words/second for write operation</td>
</tr>
<tr>
<td></td>
<td>116 K words/second for read operation</td>
</tr>
<tr>
<td></td>
<td>Used for transfers to non-consecutive memory locations.</td>
</tr>
</tbody>
</table>

All data rates subject to degradation as traffic on SBI increases. (SBI allows communication interfaces between CPU, Memory, UNIBUS and MASSBUS.)

**Maximum aggregate throughputs on UNIBUS is only 1.5 Megabytes/second.**
2.2.1.2 Perkin Elmer 3244

The Perkin Elmer (PE) 3240 series computer is a high throughput machine with a 32 bit architecture. The HOSC currently uses two PE 3244 machines with primary responsibilities as front end processors (FEP) receiving real time data streams from the KSC firing room.

A block diagram of the 3240 model computer is shown in Figure 5. Detailed information regarding the 3244 has not been obtained but a brief description of the 3244 architectures follows.

The 3244 memory subsystem is organized into banks each capable of handling 4 megabytes of addressable memory. Total system memory ranges from 256 kilobytes in one bank to a full system complement of four 4 megabyte banks for a maximum of 16 megabytes of addressable memory. All memory is connected to a common memory bus which consists of two undirectional, asynchronous, 32 bit busses. One bus is dedicated to memory write functions and the other is dedicated to memory read functions.

Input/Output is accomplished by up to five external communication busses: one multiplexer bus for medium speed devices and up to four high speed Direct Memory Access (DMA) busses. Each DMA bus supports eight high speed bidirectional ports. Each DMA port is controlled by a selector channel that controls and terminates transfers through the CPU. This selector channel is controlled through the multiplexer bus. Once the channel is activated, the processor is released and is free to continue processing. Published I/O transfer rates for the PE 3244 DMA bus indicate that transfer
Figure 5. Block Diagram of PE 3244 Computer
rates of up to 10 megabytes per second burst mode are possible for each DMA bus.\textsuperscript{6} Table 3 summarizes the PE/3244 I/O characteristics.

2.2.1.3 Perkin Elmer 8/32c

Detailed information about the 8/32 computer has not been obtained, but conceptually, the 8/32 is a machine similar in architecture to the 3244. A significant difference is that the 8/32 is capable of supporting only one DMA bus. This DMA operates in a burst mode capable of transferring 6 megabytes per second. The 8/32 will allow configuration with a buffered selector channel that accomplish the 6 MB/s rate by transferring the data in 14 half-word blocks.\textsuperscript{7} Table 4 summarizes the estimated PE 8/32c I/O characteristics.

2.2.2 NSC HYPER channel

The Network System Corporation HYPER channel (HC) is a broadband local area communication network supporting data transmissions between network users at a rate of 50 megabyte per second. The HYPER channel network (HCN) serves to interface and interconnect various sizes of mainframe computers of differing manufacturers (e.g., UNIVAC, DEC, CRAY, PERKIN ELMER) with other peripherals such as data entry terminal card readers, printers, mass storage devices and other networks. Communication is provided over a passive 75 ohm coaxial cable called a trunk.
<table>
<thead>
<tr>
<th>TABLE 3. SUMMARY OF PERKIN ELMER 3244 I/O CHARACTERISTICS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PROCESSOR</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>MEMORY</strong></td>
</tr>
<tr>
<td>Virtual Address Space: 4 Megabytes</td>
</tr>
<tr>
<td>Basic Memory Access Time: 500 nanoseconds</td>
</tr>
<tr>
<td>DMA BUS DATA TRANSFER RATE: 10 Megabytes/second-burst</td>
</tr>
<tr>
<td>mode Maximum of 4 DMA busses can be supported.</td>
</tr>
</tbody>
</table>
TABLE 4. SUMMARY OF PERKIN ELMER 8/32 I/O CHARACTERISTICS

<table>
<thead>
<tr>
<th>PROCESSOR</th>
<th>32 bit/word</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA BUS DATA TRANSFER RATE:</td>
<td>6 Megabytes/second by transferring data in 14 half-word blocks. Only one DMA can be supported.</td>
</tr>
</tbody>
</table>
Host computers gain access to the network trunk through hardware interfaces called processor adapters; unintelligent peripherals through device adapters. Network to network connection are accomplished with a link adapter which supports not only communication with standard transmission lines but also with microwave frequency RF links. Each network adapter may be connected to as many as four separate trunks and provides the service of trunk selection, trunk access, establishment of adapter to adapter virtual circuits and also provides user-to-adapter protocols. Network adapters contend for trunk control using a Carrier Sense Multiple Access scheme with prioritized staggered delays.\textsuperscript{9}

The heart of the network is the A400 Adapter. The A400 is a microcomputer controlled interface device that allows up to 4 minicomputers of the same or mixed manufacturer types to transmit and receive data over the HYPER channel network. (All four trunk port may be connected to four channels of the same minicomputer.) The A400 provides a buffered interface between the trunk and the adapter. Some of this buffer is used to provide parallel to serial data stream conversion for host to trunk transmissions and serial to parallel conversion for trunk to host transmissions.

Each A400 adapter is composed of

- a 16 bit microprocessor with 4906 words of read only memory.
- a storage section consisting of
  - 1024 8-bit bytes of control memory with odd parity
  - 4096 8-bit bytes of control buffer with odd parity
16 working registers
16 trunk registers
256 extension register
   one trunk interface.

The adapter can be expanded to contain
   4 trunk interfaces
   8192 8-bit bytes of buffer memory
   1024 8 bit bytes of code conversion memory.

Additionally, the adapter has a peripheral device interface that provides a standard interface between the internal busses of the minicomputer and the A400. The peripheral interface adapter is separated from, but connected with ribbon cables to, the nucleus adapter which provides the hardware resources such as the microprocessor and memory register. \(^8,10\) (See Figure 6)

To perform an operation on the network, the minicomputer loads the necessary parameters into the internal registers on the interface and requests the adapter to perform the indicated functions. Whenever an adapter is not performing a function, it scans all attached ports for a request to perform a function. When a function request is detected, the adapter suspends scanning and initiates the execution of the function. The flow diagram of Figure 7 illustrates the handshaking between the A400 and host processor when data transfers are initiated. Notice that the host processor initiates all actions of the adapter. (A compilation of functions that can be accomplished by the A400 is illustrated in Table 5.)\(^8\)
Host places 16 bits of data on DATA OUT bus and issues a GO pulse. Host also disables the READY line which stays inactive until function has been executed.

A function from host initiates every adapter operation.

---

Host responds by putting data on DATA OUT bus and sending END of cycle.

Accept data on bus and transfer into registers or buffers as required.

---

Figure 7. Host to A400 Adapter Data Exchange
<table>
<thead>
<tr>
<th>CODE</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>Transmit message</td>
</tr>
<tr>
<td>08</td>
<td>Transmit data</td>
</tr>
<tr>
<td>0C</td>
<td>Transmit last data</td>
</tr>
<tr>
<td>10</td>
<td>Transmit local message</td>
</tr>
<tr>
<td>24</td>
<td>Input message</td>
</tr>
<tr>
<td>28</td>
<td>Input data</td>
</tr>
<tr>
<td>40</td>
<td>Status</td>
</tr>
<tr>
<td>50</td>
<td>Dump extension register</td>
</tr>
<tr>
<td>60</td>
<td>Mark down port 0</td>
</tr>
<tr>
<td>64</td>
<td>Mark down port 1</td>
</tr>
<tr>
<td>68</td>
<td>Mark down port 2</td>
</tr>
<tr>
<td>6C</td>
<td>Mark down port 3</td>
</tr>
<tr>
<td>70</td>
<td>Mark down port 0 and re-route messages</td>
</tr>
<tr>
<td>74</td>
<td>Mark down port 1 and re-route messages</td>
</tr>
<tr>
<td>78</td>
<td>Mark down port 2 and re-route messages</td>
</tr>
<tr>
<td>7C</td>
<td>Mark down port 3 and re-route messages</td>
</tr>
<tr>
<td>A0</td>
<td>Read statistics</td>
</tr>
<tr>
<td>A4</td>
<td>Read and clear statistics</td>
</tr>
<tr>
<td>C0</td>
<td>Set test</td>
</tr>
<tr>
<td>C4</td>
<td>Set address and length</td>
</tr>
<tr>
<td>C8</td>
<td>Write buffer</td>
</tr>
<tr>
<td>CC</td>
<td>Read buffer</td>
</tr>
<tr>
<td>E0</td>
<td>Clear adapter</td>
</tr>
<tr>
<td>E4</td>
<td>End operation</td>
</tr>
<tr>
<td>E6</td>
<td>Clear and wait for message</td>
</tr>
<tr>
<td>E8</td>
<td>Wait for message</td>
</tr>
</tbody>
</table>

TABLE 5. A400 ADAPTER FUNCTION DESCRIPTION
Data can be transferred from host to adapter in two different modes: direct memory access (DMA) and register mode. In the DMA mode, the adapter uses an alternating buffer scheme. The adapter accepts data from the device into buffer memory. When the buffer is half full, trunk transmission of that amount of data is initiated as, the other half of the buffer is being filled. This filling and sending is continued until all data has been transferred. All DMA transfers are through the extension registers and are initiated by the adapter microprocessor and controlled by the adapter hardware.

In the register mode, data movement is also between the device interface and the nucleus adapter but the DMA controls are not used. These data transfers are also through the extension register but initiated and controlled by the microprocessor.

Data transfers from adapter to adapter are accomplished by the trunk interface. The trunk interface consists of a passive coaxial cable that transmits data serially between two adapters. Each trunk can have up to 64 drops depending on the length of the trunk cable and its transmission qualities.

Transmissions on a trunk are initiated and monitored by the trunk driver which is a microcode program stored in the adapter PROMs. The extension register and the trunk registers support the PROM trunk driver. When an adapter is ready to transmit, it must first contend for use of the trunk. The method for contention is called contention allocation. It is so called because the trunk is allocated to an adapter based on the adapter's need to transmit.
The contention process can be summarized as follows. The adapter first just attempts to transmit on the trunk. If the trunk is busy, the transmitter is disabled. When the trunk becomes free, a fixed delay is initiated by the adapter. This prevents the adapter from transmitting until the receiving party of the most recent transmission has had time to receive a response frame. Upon expiration of the fixed delay, another delay called the priority delay is initiated. This delay is different for each adapter and provides a unique time slot for each adapter on the trunk. Another delay, called end delay, is provided following the fixed delay. This delay is provided to insure that all adapters with higher priority have first access to the trunk. Obviously, with this trunk allocation scheme, higher priority adapters can dominate the trunk. To prevent this, each adapter has a flip flop in it that is known as the wait flip flop. This flip flop is set when the adapter transmits and is cleared when an end delay is signalled. This flip flop is intended to provide a more equitable contention environment. Although all adapters are equipped with wait flip flops, they may be disabled to provide assured trunk access.\textsuperscript{9,10}

Figure 8 shows the flow of the wait algorithm.

Upon gaining access to the trunk, either a function message or data can be transmitted in trunk frames. When a frame is transmitted all adapters receive the frame. The adapter compares the received adapter access code which is part of the frame header with its own code which can be set by thumbwheel switches in the adapter. If and only if the codes match can the communication be accepted. (A zero in the receiving adapter code represents a "don't care" condition and
Original page is of poor quality

Data to transmit on trunk?

Attempt to transmit

Transmit

Trunk busy?

Transmit line disabled?

Initiate fixed delay

Fixed delay expired?

Initiate priority delay?

Priority delay expired?

Wait flip-flop enabled?

Initiate end delay

End delay expired?

Fixed delay prevents adapter from transmitting until the receiving party of ongoing transmission has sent a trunk response frame.

Priority delay provides unique time slot for each adapter on trunk.

Wait fused to keep higher priority adapters from dominating the trunk may be disabled or enabled.

All other adapters have free access to trunk during this

Figure 8. Trunk Contention Delay Algorithm
the receiving adapter will accept any character in that code
position.)

The receiving adapter responds to the receipt of a trunk
transmission with a trunk response frame. This notifies the sending
adapter of the status of the received message. Every transmission
frame requires the receipt of a response frame or the sending adapter
will time out and retry the transmission. This process will be
repeated 256 times. If unsuccessful at transmitting the message,
the adapter will terminate the operation and record some status bits
for the host in the adapter extension registers. 10

2.2.3 Other HOSC Components

In addition to the computer resources are several other devices
in the HOSC. Currently these other devices act as peripherals to the
processors on the HYPER channel network and consequently do not directly
affect traffic on the network. Indirectly, they represent overhead
processor activity and thus slow traffic throughput on the processor
I/O busses. These devices will include a Gandalf solid state switching
matrix that acts to interface the MIPS consoles through VAX4. Also
included in the peripherals are various strip recorders and three
twelve-channel Genesco digital televisions that interface the engineering
consoles through STS Primary (PE 3244), STS Backup (PE 8/32c) and
Spacelab 8/32. No further descriptions of these devices are currently
available.
3.0 HOSC ANALYSIS RESULTS

The HOSC system analysis initiated with a study of the system componente, the computers, the HYPER channel network, the data flow activity of each device and the input-output characteristics of each device. The system operation is statistical in nature and, although a mathematical analysis is possible, it is not feasible to make such an analysis with much fidelity. Rather a simulation model that emulates the HOSC system with good fidelity can be used to achieve information concerning average bus traffic, average waiting time, collision frequency and maximum waiting times. Furthermore, these parameters can be investigated as a function of HOSC system configuration, input-output variations, and data file dump requirements.

The development of a simulation model with good fidelity has been accomplished. The HOSC system has been modeled with three different program simulations and these three algorithms have been compared against each other. The purpose in using three algorithms was to insure validity of the simulation results, a necessity due to the lack of sufficient system statistics to validate a single simulation algorithm. The three algorithms are similar, but have been programmed in BASIC, PASCAL and SLAM.

BASIC is an engineering oriented language not at all unlike FORTRAN. This simulation program is the main program. The program is listed in Appendix I.

Although many simulation runs were made with simple system configurations that allowed the simulation algorithm to be verified, there is no need to present those in this report. The monthly reports
document these earlier runs and the development of the algorithm. Rather it suffices to illustrate the simulation of the HOSC system as it is projected in configuration in Summer 1983.

3.1 Typical Basic Algorithm Information Printouts

Figure 9 depicts the HOSC simulation configuration which is documented in this report. This configuration is perhaps more complex than the actual system configuration for the present, but it is the type of configuration that is desired in the near future. Not all devices are transmitters of data in this system configuration. The A400 labeled port 4 only receives data transmitted on the HYPER channel bus. Other devices receive outside data and transmit and received data over the HYPER channel bus. This system configuration was devised at a meeting between this investigator and NASA/MSFC HOSC personnel on March 9, 1983, and is typical of the configurations to be utilized for HOSC applications in the near future.

In order to determine the number of simulation runs necessary to produce representative statistics and to let the system algorithm achieve steady state, as would occur in the actual system, several runs with the same system configuration parameters but with varying numbers of data transfers were made and the statistics compared.

Figures 10 and 11 illustrate the program printout that depicts the system configuration of figure 9. The # of bytes accumulated refers to the number of bytes which a particular device will accumulate refers to the number of bytes which a particular device will accumulate from a source before it transmits that data to the appropriate destination. As may be noted in Figure 10, there is a
Figure 9. HOSC Simulation Configuration
<table>
<thead>
<tr>
<th>PORT, DEVICE, SOURCE</th>
<th>AVERAGE ARRIVAL RATE</th>
<th># OF BYTES ACCUMULATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>15000</td>
<td>2048</td>
</tr>
<tr>
<td>2 1 1</td>
<td>4000</td>
<td>4000</td>
</tr>
<tr>
<td>2 2 1</td>
<td>6400</td>
<td>6400</td>
</tr>
<tr>
<td>3 1 1</td>
<td>1000</td>
<td>512</td>
</tr>
<tr>
<td>3 2 1</td>
<td>1000</td>
<td>512</td>
</tr>
<tr>
<td>3 3 1</td>
<td>625</td>
<td>625</td>
</tr>
<tr>
<td>4 1 1</td>
<td>0</td>
<td>2000</td>
</tr>
<tr>
<td>4 2 1</td>
<td>0</td>
<td>2000</td>
</tr>
<tr>
<td>4 3 1</td>
<td>0</td>
<td>2000</td>
</tr>
</tbody>
</table>

(204 DUMP DATA FILE DUMP 64000 12)

Figure 10. System Configuration Parameters for 1333 Data Transfer Simulation.
WAIT TIME OF PORT 1 BEFORE TRANSMISSION REQUEST (IN EVENT OF A COLLISION) .000001
WAIT TIME OF PORT 2 BEFORE TRANSMISSION REQUEST (IN EVENT OF A COLLISION) .000002
WAIT TIME OF PORT 3 BEFORE TRANSMISSION REQUEST (IN EVENT OF A COLLISION) .000003
WAIT TIME OF PORT 4 BEFORE TRANSMISSION REQUEST (IN EVENT OF A COLLISION) .000004

CHANNEL SETUP AND RELEASE TIME = .000025

CHANNEL DATA TRANSFER RATE = 6250000

<table>
<thead>
<tr>
<th>PORT.DEVICE</th>
<th>TRANSFER RATE</th>
<th>LOAD TIMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1</td>
<td>500000</td>
<td>.000002</td>
</tr>
<tr>
<td>1 2</td>
<td>500000</td>
<td>.000002</td>
</tr>
<tr>
<td>2 2</td>
<td>500000</td>
<td>.000002</td>
</tr>
<tr>
<td>3 1</td>
<td>500000</td>
<td>.000002</td>
</tr>
<tr>
<td>3 2</td>
<td>500000</td>
<td>.000002</td>
</tr>
<tr>
<td>3 3</td>
<td>3300000</td>
<td>3.0303030303E-7</td>
</tr>
<tr>
<td>4 1</td>
<td>3300000</td>
<td>3.0303030303E-7</td>
</tr>
<tr>
<td>4 2</td>
<td>1200000</td>
<td>8.3333333333E-7</td>
</tr>
<tr>
<td>4 3</td>
<td>1200000</td>
<td>8.3333333333E-7</td>
</tr>
</tbody>
</table>

COMBINATIONS (IJK - LMN) PROBABILITY TIME TO TRANSMIT Q(IJK) BYTES (IJK - LMN)
111311        .7     .004441
111321        .3     .004441
211431        1      .008345
221421        1      .013145
311321        .98    .001369
311221        .01    .001369
311211        .01    .001369
321311        .98    .001369
321221        .01    .001369
321211        .01    .001369
331421        1      6.6583333333E-4

Figure 11. System Configuration Parameters for 1333 Data Transfers.
data file dump of 64,000 bytes every 12 seconds as would be typical of a data file refresh operation.

The channel setup and release time is a parameter used to allow the HYPER channel to establish a transmission link and then to release the link after data transfer is complete.

As data is transferred across the system, each port vies for the bus in a contention scheme described in Section 2. Occasionally the ports will collide trying to transmit simultaneously. In Figures 12 and 13, a printout record of the results of a collision is illustrated. These printout records allow the operator to ensure the collision algorithm is working properly. As may be noted, 331 and 111 incurred a collision and 111 retransmitted first, since its assigned waiting time is less than 331.

Every time a data transfer occurs between two devices on the same port, the HYPER channel bus is not utilized and thus it is free for other transmissions except to the port involved in an inter-port data transfer. Figure 14 illustrates a printout record of an inter-device data transfer. Figure 14 also illustrates a record of a data file dump.

The statistical printout of a simulation run is illustrated in Figure 15. This run had 1333 data transfers and over three million bytes transferred.

3.2 Results of A Simulation Run

Inspection of Figure 15 will illustrate the information garnered by a simulation run. The items of interest are the bus busy time, collision frequency of each source, the average waiting time, the longest waiting time and the relative activity of each source.
A collision occurred between 331 and 111

ITERATION NUMBER 336

PROBABILITY 0.3
RANDOM NUMBER .364597057222
PROBABILITY 1

ESTIMATION 321

<table>
<thead>
<tr>
<th>ORT</th>
<th>DEVICE</th>
<th>SOURCE</th>
<th>TOTAL TIME GONE BY SINCE LAST TRANSMISSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.3571916677</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1.358634999984</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1.6684604999377</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1.201452499979</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1.301452499979</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>6</td>
<td>1.0000001</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7</td>
<td>9999999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>8</td>
<td>9999999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>9</td>
<td>9999999</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORT</th>
<th>DEVICE</th>
<th>SOURCE</th>
<th>REMAINING TIME TO TRANSMIT (NORMALIZED FASHION)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.000614166622</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.001969506918</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0.331539506021</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>0.20547506021</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>0.210547506021</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>6</td>
<td>9999999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7</td>
<td>9999999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>8</td>
<td>9999999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>9</td>
<td>9999999</td>
</tr>
</tbody>
</table>

COLLISION NUMBER 3
DATA TRANSFER .004443

Figure 12. Typical Printout When Data Transfer Collision Occurs.
### Accumulated Waiting Times Since the Last Transmission

<table>
<thead>
<tr>
<th>ORT</th>
<th>DEVICE</th>
<th>SOURCE</th>
<th>Accumulated Waiting Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.502502499984</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.672901499977</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.305893499979</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.305893499979</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1.004442</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>999999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>999999</td>
</tr>
</tbody>
</table>

**DATA TRANSFER COMPLETED: TOTAL TIME GONE BY 28.340377331**

### Remaining Time to Transmit (Normalized Fashion)

<table>
<thead>
<tr>
<th>ORT</th>
<th>DEVICE</th>
<th>SOURCE</th>
<th>Remaining Time to Transmit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.136533333333</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.197497506616</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.327098500023</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.206106500021</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.206106500021</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>-0.004442</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>999999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>999999</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>999999</td>
</tr>
</tbody>
</table>

**EXT TRANSMISSION REQUEST BY 331**

**TOTAL TIME THAT WILL BE GONE BY WHEN 331 STARTS TO TRANSMIT 28.340377331**

**ITERATION NUMBER 387**

**RISKABILITY 1**

**RANDOM NUMBER .608213.51646**

**ESTIMATION 421**

Figure 13. Typical Printout When Data Transfer Collision Occurs (Continued).
TRANSMISSION ON SAME A400, SOURCE/DEST = 32131
NO COLLISION OCCURED.

SOURCE 1, 1 DUMPED TO DESTINATION 4, 1 64000 BYTES OF DATA
TOTAL TIME GONE BY (BEFORE THE DUMP) 96.4481409909
TOTAL TIME GONE BY (AFTER THE DUMP) 96.5764859909

TRANSMISSION ON SAME A400 SOURCE/DEST = 311321
NO COLLISION OCCURED
TRANSMISSION ON SAME A400 SOURCE/DEST = 32131
NO COLLISION OCCURED

Figure 14. Typical Printout for Data File Dump and Inter Source Data Transfers.
### Source to Destination

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th># of Times Transmitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>111321</td>
<td>111321</td>
<td>48</td>
</tr>
<tr>
<td>111321</td>
<td>311431</td>
<td>194</td>
</tr>
<tr>
<td>311431</td>
<td>311431</td>
<td>95</td>
</tr>
<tr>
<td>311431</td>
<td>311421</td>
<td>34</td>
</tr>
<tr>
<td>311421</td>
<td>311431</td>
<td>181</td>
</tr>
<tr>
<td>311421</td>
<td>311421</td>
<td>3</td>
</tr>
<tr>
<td>311421</td>
<td>311321</td>
<td>6</td>
</tr>
<tr>
<td>311321</td>
<td>321311</td>
<td>185</td>
</tr>
<tr>
<td>321311</td>
<td>321311</td>
<td>1</td>
</tr>
<tr>
<td>321311</td>
<td>321311</td>
<td>1</td>
</tr>
<tr>
<td>321311</td>
<td>321311</td>
<td>1</td>
</tr>
<tr>
<td>321311</td>
<td>331421</td>
<td>95</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>1333</td>
</tr>
</tbody>
</table>

### Longest Waiting Time

<table>
<thead>
<tr>
<th>Source</th>
<th>Port</th>
<th>Device</th>
<th>Source</th>
<th>Average Waiting Time</th>
<th>Longest Waiting Time</th>
<th>Collisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-0.128345</td>
<td>-0.128345</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-0.128345</td>
<td>-0.045081</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-0.128345</td>
<td>-0.17091357036E-3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-0.129714</td>
<td>-0.10163916665</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-0.10163916665</td>
<td>-0.03009355127</td>
<td>1</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Original Page 15 Quality of Poor Quality**

‘Original Page 15 Quality of Poor Quality’

### Percent Busy Time

Percent Bus Busy Time = 6.34%

### Data Transfer Simulation

#### Number of Data Transfer Collisions

Number of Data Transfer Collisions = 6 Collisions

#### Number of Data File Dumps

Number of Data File Dumps (11 to 41) = 8 (64,000 Bytes Each)

#### Bytes Transferred by Data File Dump

Bytes Transferred by Data File Dump = 512,000 Bytes

#### Bytes Transferred Source to Destination

Bytes Transferred Source to Destination = 2,616,399 Bytes

#### Total Bytes Transferred

Total Bytes Transferred = 3,128,399 Bytes

---

Figure 15. Statistical Summary of 1333 Data Transfer Simulation.
For the 1333 data transfer simulation with data file dump of 64,000 bytes every 12 seconds the parameters of interest are depicted in the figure. Everything points to a satisfactory operation at this point with one notable exception, the longest waiting time. This Figure is -128,345 milliseconds (the negative sign indicates waiting time) for P.D.S. 111 which has an average time between transmission requests of 136.533 milliseconds (# of bytes accumulated divided by the average arrival rate of the bytes. The data is drawn from Figure 10.)

This waiting time amounts to 94% of the average time between transmission requests for P.D.S. 111 and serves a warning that P.D.S. 111 is on the verge of being overloaded. This may be alleviated by several means—changing the number of bytes to be accumulated before making a transmission request or by changing the data file dump time interval. In Section 3.4, this is discussed more fully.

3.3 Comparison of BASIC, PASCAL and SLAM Programs

Appendix II contains a listing of the PASCAL program and summary sheet for the HOSC simulation. The SLAM program listing is not included since NASA/MSFC does not carry SLAM software support. The three algorithms are compared in a broad sense in Table 6. The three algorithms were run for comparison purposes using 500 data transfers as the benchmark. The total bus time, bus utilization percentage, number of bytes transferred all compare very favorably. The number of collisions incurred vary due to differences in the collision algorithms used in the programs which were programmed by three different programmers as a check on the algorithms. The
### Table 6
COMPARISONS OF BASIC PASCAL AND SLAM PROGRAMS
NO DATA FILE DUMPS SIMULATED

<table>
<thead>
<tr>
<th></th>
<th>BASIC</th>
<th>PASCAL</th>
<th>SLAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Transmissions</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Elapsed Time on Bus</td>
<td>36.08</td>
<td>36.44</td>
<td>35.24</td>
</tr>
<tr>
<td>(Seconds)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collisions</td>
<td>8</td>
<td>29</td>
<td>19</td>
</tr>
<tr>
<td>Waits</td>
<td>N.T.</td>
<td>83</td>
<td>36</td>
</tr>
<tr>
<td>Bus Utilization</td>
<td>5.82%</td>
<td>4.69%</td>
<td>6.02%</td>
</tr>
<tr>
<td>Bytes Transferred</td>
<td>979,795</td>
<td>987,360</td>
<td>985,939</td>
</tr>
</tbody>
</table>

N.T. = Not Tabulated
The algorithms programmed all have the following features:

1. Allows up to 9 A400 adaptors
2. Allows up to 9 computer devices per A400 adaptor
3. Allows up to 9 data sources per device
4. Allows each device to transfer large block of data on a periodic basis such as for CRT data base refresh
5. Allows assignment of individual waiting times to be assigned to each A400 in event of a collision
6. Allows the shortest assigned waiting time A400 to retry a transmission in the event of a collision
7. If a data transfer occurs between two devices on the same A400 (Inter A400 data transfer) it allows this to occur without tying up the bus
8. Allows for individual source data arrival rates
9. Allows for individual source data buffer sizes (relates to time between transmission requests)
10. Allows for individual device to A400 I/O data rates.

The results indicate no major discrepancies lie in the HOSC system model used for the algorithm development. The BASIC program has been emphasized since it is more transportable than SLAM or PASCAL. However, for a next generation simulation model, PASCAL will be constructed in a user friendly format since it has some features which make it suitable for this type of simulation.

3.4 Conclusions

Results of a fair run simulation using the configuration of Figure 9 has been tabulated in Table 7. The purpose of this comparison was to determine:
<table>
<thead>
<tr>
<th>P.D.S.</th>
<th>TTMN(HS)</th>
<th>RUN A</th>
<th>RUN B</th>
<th>RUN C</th>
<th>RUN D</th>
<th>RUN E</th>
<th>AVERAGE WAITING TIME (%)</th>
<th>BUS BURST TIME (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>136</td>
<td>128 (94%)</td>
<td>128 (94%)</td>
<td>256 (188%)</td>
<td>128 (94%)</td>
<td>8 (6%)</td>
<td>29</td>
<td>27</td>
</tr>
<tr>
<td>2 1 1</td>
<td>1000</td>
<td>128 (132%)</td>
<td>128 (132%)</td>
<td>118 (11.8%)</td>
<td>19 (2%)</td>
<td>4 (.4%)</td>
<td>2.8</td>
<td>4.1</td>
</tr>
<tr>
<td>2 2 1</td>
<td>1000</td>
<td>45 (5%)</td>
<td>45 (5%)</td>
<td>8.4 (.8%)</td>
<td>8 (.8%)</td>
<td>9 (1%)</td>
<td>1.5</td>
<td>1.7</td>
</tr>
<tr>
<td>3 1 1</td>
<td>512</td>
<td>129 (25%)</td>
<td>128 (25%)</td>
<td>256 (50%)</td>
<td>116 (23%)</td>
<td>5 (1%)</td>
<td>1.9</td>
<td>10.9</td>
</tr>
<tr>
<td>3 2 1</td>
<td>512</td>
<td>128 (25%)</td>
<td>130 (26%)</td>
<td>258 (50%)</td>
<td>116 (23%)</td>
<td>4 (1%)</td>
<td>3.2</td>
<td>1.2</td>
</tr>
<tr>
<td>3 3 1</td>
<td>1000</td>
<td>101 (10%)</td>
<td>101 (10%)</td>
<td>202 (20%)</td>
<td>101 (10%)</td>
<td>13 (1.3%)</td>
<td>3.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

P.D.S = PORT, DEVICE, SOURCE

Average Waiting Time is expressed as percent of TTMN

TTMN = Time to Next Transmission Request = Time Between Data Transfer Requests

Number of Collisions: RUN A = 10; RUN B = 6; RUN C = 14; RUN D = 3; RUN E = 3

RUN A 1500 data transfers, Data file dump once every 12 seconds, 64,000 Bytes each dump
RUN B 1333 data transfers, Data file dump once every 12 seconds, 64,000 Bytes each dump
RUN C 1000 data transfers, Data file dump once every 12 seconds, 128,000 Bytes each dump
RUN D 500 data transfers, Data file dump once every 12 seconds, 64,000 Bytes each dump
RUN E 500 data transfers, Data file dump once every 120 seconds, 64,000 Bytes each dump
(For a 500 iteration run time of 36 milliseconds no data file dump occurs.)
a) The number of iterations necessary to produce consistent results.

b) Determine the effects of varying the data file dump interval.

Since each simulation run uses random number generators as part of the program, it should not be surprising to see small differences in the output summary statistics. Indeed two runs of the same configuration with the same number of iterations will produce slightly different results. This is completely expected and does not reduce the value of the results at all.

The results tabulated in Table 7 indicate that the system performs as desired with one exception. The data file dump. Every 24 seconds, Run C, creates a waiting time of 256 milliseconds for P.D.S. 1.1, which is 188% of the time between transmission requests for source P.D.S. 111. The impact of these results from a realization that 100% waiting time for a source would mean that source has been waiting for an opportunity to transmit for such a long time that it now has three transmission requests rather than one. It is obvious from the data in Table 7 that large data file dumps will tend to create a log jam for devices with active external data sources. These external data sources may not be extinguishable; hence, the need to provide sufficient data storage in the device to hold incoming data during a large data file dump is paramount.

An analytical feeling for this problem is easily derived. Any source with a data file dump will experience an external source transmission request whenever the data file dump time exceeds the total time between transmission requests for that source. In the
configuration of Figure 9, we have a data file dump from P.D. 11 to P.D. 41. The I/O rate for P.D. 11 is 500 K bytes per second and the I/O rate for P.D. 41 is 3.30 M bytes per second. The channel transfer rate is 0.28 M bytes per second so that data file dump will take a total time equal to the channel setup release time plus the time to dump X bytes at the slowest I/O rate or for our configurations the data file dump time (DFDT) is

\[ \text{DFDT} = 25 \text{ msec} + X(2 \text{ msec/Byte}) \]

For 64,000 bytes DFDT = 128 m sec and
for 128,000 bytes DFDT = 256 m sec.

In fact whenever the DFDT exceeds the fastest source average time to transmission request time a problem will definitely arise. For the example where the number of bytes in a data file dump exceeds

\[
\frac{\text{Number of Bytes}}{\text{In Data File Dump}} = 126,531 - 0.25 = 68254 \text{ Bytes,}
\]

the fastest source will possibly incur two transmission requests.

There are several ways to correct this situation:

a) Increase the buffer size of the source so it can store more than two full sets of data between transmission opportunities. This is a viable option since source P.D.S. 11 is the most active with highest outside data arrival rate and only 2048 bytes to be accumulated between transmission requests.
b) Perform data file dumps more often but transfer proportionally less bytes per dump thereby reducing any sources waiting time. This may not be a viable option due to the lack of data file data being ready in a somewhat steady occurrence rate. Also this would require a data file storage medium at the destinations; however, this is usually the case.

c) Break up the data file dump by transmitting it in smaller packets. That is rather than 64,000 bytes in a steady stream for a short time once every 12 seconds, transmit 8,000 bytes, break and release channel to allow another user but request transmission rights immediately and repeat for 8 times. This would transfer the data in almost the same time as sending all 64,000 bytes while allowing the active devices a chance to clear their stored data.

All the above options could be accomplished through software programming of the source device P.D. 11; thus, it is a HOSC system operators choice of which method to utilize.
4.0 LITERATURE SURVEY

4.1 References

   (205) 882-2366.


   p. 1.6.


   (504) 888-7230.


7. Staggs, Clint. Telephone Conversation, Sept. 17, 1982. Perkin Elmer Corporation, Huntsville, AL,
   (205) 533-6123.


   82-3, WR Franta and John Harth. Univ. of Minn, 1982.

4.2 Bibliography

4.2.1 Publications


4.2.2 Reference Manuals (by Manufacturer)


APPENDIX I

BASIC SIMULATION ALGORITHM PROGRAM

A listing of the BASIC simulation algorithm program is presented in this appendix. The program contains plentiful comments and is user oriented, with prompts and options displayed on the interactive screen. The BASIC language is common to many machines; however, the input output commands are usually particular to a single machine, in this case the HP-87 system.
PROGRAM: HUSC
AUTHOR: DR. FRANK INGELS
PROGRAMMER: TERESA BENNET
DATE: MAY 9, 1988

1. I -- # OF PORTS
2. IJ -- # OF DEVICES FOR EACH PORT
3. IJK -- # OF SOURCES FOR EACH DEVICE
4. LAI(J,K) -- AVERAGE ARRIVAL RATE OF DATA TO SOURCE
5. QUI(J,K) -- # OF BYTES ACCUMULATED BY SOURCE K, DEVICE J, PORT I
6. TSR -- CHANNEL SETUP AND RELEASE TIME
7. CR -- CHANNEL DATA TRANSFER RATE
8. PR(IJK,LMN) -- PROBABILITY THAT SOURCE IJK WILL TRANSMIT DATA TO LMN
9. BBT = BUSY BUSY TIME
10. BEGIN PROGRAM EXECUTION
11. INTEGER I,J,K,L,M,N,O
12. BBT=0 : CH1=999999 : CH2=999999 : CLNS="N" : CLNS="N" : EGS="N" : AFS="F"
13. DIM PDS(200),PRO(200),T(200),TIMES(200),DT(200),BYTES(20),TIM(20),BIFL(20)
14. CLEAR : DISP TAB(20):"MENU" : DISP : DISP "1 -- RECALL AN EXISTING DATA FILE"
15. DISP "2 -- EXECUTE PROGRAM WITHOUT THE FILE OPTION"
16. DISP "3 -- CREATE A NEW DATA FILE AND EXECUTE PROGRAM"
17. DISP : DISP "ENTER YOUR SELECTION " : INPUT AN
18. IF AN<1 OR AN>3 THEN 210
19. PRINT "PRINT IS 701"
20. DISP " " : DISP "INDICATE THE NUMBER OF PROGRAM ITERATIONS " : INPUT NUMBER
21. ON AN GOTO 250,310,280
22. DIS "ENTER THE DATA FILE NAME " : INPUT DFNAMES
23. IF AN=1 THEN GETDATA=999 : GOTO 1550 ELSE GETDATA=888
24. DATA ENTRY ROUTINE (IF AN=2 OR AN=3)
25. LIN 310-1330
26. CLEAR : DISP "PLEASE ENTER THE FOLLOWING DATA ITEMS:" : Disp
27. DIS "# OF HYPERCHANNEL PORTS " : INPUT HY
28. FOR I=1 TO HY : Disp "# OF DEVICES FOR PORT " : I: " : DIS " INPUT DVI(I) " : NEXT I 
29. CLEAR
30. FOR K=1 TO DV(I)
32. NEXT K = I+1 : IF I< HY THEN 340 ELSE I=0
370 I=I+1 CLEAR FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) DO CLEAR
380 DISP "FOR " .I:" DEVICE ":J:" SOURCE ":"-" ;G DISP $ DISPL
390 DISP "AVERAGE ARRIVAL RATE ":I ;G INPUT LA(I)
400 DISP "+ OF BYTES ACCUMULATED BEFORE A TRANSMISSION REQUEST IS TO BE MADE BY
410 DECI ":I ";G DISPI "VALUE ASSIGNED 500 BYTES UNLESS"
420 QS=1G QS(J,J)=500 & DISP " OTHERWISE SPECIFIED" :G INPUT QS
430 IF QS <> " " THEN QS(J,J)=VAL (QS)
440 NEXT J & NEXT I & ON I GOSUB 450,460,530,570,610,650,690,730,770
450 IF I=HY THEN 810 ELSE GOTO 370
460 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & Lai(J,I)=LA(J,K) & LA(J,J)=0 & QA(J,
470 =QQ(J,J) & QQ(J,J)=0
480 IF LA1(J,1)=0 THEN ATR1(J,1)=99999 ELSE ATR1(J,1)=Q1(J,1)/LA1(J,1)
490 NEXT I & NEXT J & RETURN
500 IF LA1(J,1) & QQ(J,1)=QQ(J,1)
510 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA2(J,1)=LAI(J,1) & LA(J,J)=0 & QA(J,
520 =QQ(J,J) & QQ(J,J)=0
530 IF LA2(J,1)=0 THEN ATR2(J,1)=99999 ELSE ATR2(J,1)=Q2(J,1)/LA2(J,1)
540 NEXT J & NEXT J & RETURN
550 IF LA2(J,1) & QQ(J,1)=QQ(J,1)
560 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA3(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
570 =QQ(J,J) & QQ(J,J)=0
580 IF LA3(J,1)=0 THEN ATR3(J,1)=99999 ELSE ATR3(J,1)=Q3(J,1)/LA3(J,1)
590 NEXT J & NEXT J & RETURN
600 IF LA3(J,1) & QQ(J,1)=QQ(J,1)
610 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA4(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
620 =QQ(J,J) & QQ(J,J)=0
630 IF LA4(J,1)=0 THEN ATR4(J,1)=99999 ELSE ATR4(J,1)=Q4(J,1)/LA4(J,1)
640 NEXT J & NEXT J & RETURN
650 IF LA4(J,1) & QQ(J,1)=QQ(J,1)
660 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA5(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
670 =QQ(J,J) & QQ(J,J)=0
680 IF LA5(J,1)=0 THEN ATR5(J,1)=99999 ELSE ATR5(J,1)=Q5(J,1)/LA5(J,1)
690 NEXT J & NEXT J & RETURN
700 IF LA5(J,1) & QQ(J,1)=QQ(J,1)
710 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA6(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
720 =QQ(J,J) & QQ(J,J)=0
730 IF LA6(J,1)=0 THEN ATR6(J,1)=99999 ELSE ATR6(J,1)=Q6(J,1)/LA6(J,1)
740 NEXT J & NEXT J & RETURN
750 IF LA6(J,1) & QQ(J,1)=QQ(J,1)
760 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA7(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
770 =QQ(J,J) & QQ(J,J)=0
780 IF LA7(J,1)=0 THEN ATR7(J,1)=99999 ELSE ATR7(J,1)=Q7(J,1)/LA7(J,1)
790 NEXT J & NEXT J & RETURN
800 IF LA7(J,1) & QQ(J,1)=QQ(J,1)
810 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA8(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
820 =QQ(J,J) & QQ(J,J)=0
830 IF LA8(J,1)=0 THEN ATR8(J,1)=99999 ELSE ATR8(J,1)=Q8(J,1)/LA8(J,1)
840 NEXT J & NEXT J & RETURN
850 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA9(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
860 =QQ(J,J) & QQ(J,J)=0
870 IF LA9(J,1)=0 THEN ATR9(J,1)=99999 ELSE ATR9(J,1)=Q9(J,1)/LA9(J,1)
880 NEXT J & NEXT J & RETURN
890 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA10(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
900 =QQ(J,J) & QQ(J,J)=0
910 IF LA10(J,1)=0 THEN ATR10(J,1)=99999 ELSE ATR10(J,1)=Q10(J,1)/LA10(J,1)
920 NEXT J & NEXT J & RETURN
930 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA11(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,
940 =QQ(J,J) & QQ(J,J)=0
950 IF LA11(J,1)=0 THEN ATR11(J,1)=99999 ELSE ATR11(J,1)=Q11(J,1)/LA11(J,1)
960 NEXT J & NEXT J & RETURN
970 FOR J=1 TO Dv(I) & FOR K=1 TO Sr(I,J) & LA12(J,1)=LA(J,1) & LA(J,J)=0 & QA(J,

740 IF LA8(J,K)=0 THEN ATR8(J,K)=99999 ELSE ATR8(J,K)=Q8(J,K)/LA8(J,K)
750 NEXT I & NEXT J & RETURN;
760 LA=LA8(J,K) @ QQ=Q8(J,K) @ ATR=ATR8(J,K) @ RETURN
70 FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ LA9(J,K)=LA8(J,K) @ LA(J,K)=0 @ QQ(J,K)=0 @ NEXT K @ NEXT J @ RETURN
80 IF LA9(J,K)=0 THEN ATR9(J,K)=99999 ELSE ATR9(J,K)=Q9(J,K)/LA9(J,K)
90 NEXT K & NEXT J & RETURN
00 LA=LA9(J,K) @ QQ=Q9(J,K) @ ATR=ATR9(J,K) @ RETURN
10 CLEAR @ FOR I=1 TO HY @ FOR J=1 TO DV(I)
20 DISP "PROPAGATION DISTANCE BETWEEN PORT ":I:" AND DEVICE ":J:" IN MICROSECONDS;
30 INPUT D(I,J)
40 NEXT J & NEXT I
50 CLEAR & DISP "CHANNEL SETUP AND RELEASE TIME ASSIGNED THE VALUE 25 MICROSECONDS"
60 QQ=.000025 @ DISP "" @ DISP "PRESS <END LINE> OR ENTER THE PROPER VALUE 
70 INPUT QQ @ IF QQ <> "" THEN TSR=VAL(QQ) ELSE TSR=.000025
80 DISP "" @ DISP "CHANNEL DATA TRANSFER RATE IS 6.25MBS" @ DISP ""
90 QQ=6250000 @ DISP "PRESS <END LINE> OR ENTER THE PROPER VALUE 
100 INPUT QQ @ IF QQ <> "" THEN CR=VAL(QQ) ELSE CR=6250000
00 CLEAR @ FOR I=1 TO HY @ FOR J=1 TO DV(I)
10 DISP "RATE AT WHICH DATA IS TRANSFERRED FROM DEVICE ":J:" TO PORT ":I:
20 INPUT R(I,J)
30 L(I,J)=1/R(I,J)
40 NEXT J & NEXT I
50 GOSUB 960 @ GOTO 1080
50 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
70 ON I GOSUB 990,1000,1010,1026,1030,1034,1046,1050,1056,1070
80 NEXT K & NEXT J & NEXT I @ RETURN
90 MNEG1(J,K)=0 @ TNG1(J,K)=0 @ TN1(J,K)=0 @ COL1(J,K)=0 @ RETURN;
100 MNEG2(J,K)=0 @ TNG2(J,K)=0 @ TN2(J,K)=0 @ COL2(J,K)=0 @ RETURN
110 MNEG3(J,K)=0 @ TNG3(J,K)=0 @ TN3(J,K)=0 @ COL3(J,K)=0 @ RETURN
120 MNEG4(J,K)=0 @ TNG4(J,K)=0 @ TN4(J,K)=0 @ COL4(J,K)=0 @ RETURN
130 MNEG5(J,K)=0 @ TNG5(J,K)=0 @ TN5(J,K)=0 @ COL5(J,K)=0 @ RETURN
140 MNEG6(J,K)=0 @ TNG6(J,K)=0 @ TN6(J,K)=0 @ COL6(J,K)=0 @ RETURN
150 MNEG7(J,K)=0 @ TNG7(J,K)=0 @ TN7(J,K)=0 @ COL7(J,K)=0 @ RETURN
160 MNEG8(J,K)=0 @ TNG8(J,K)=0 @ TN8(J,K)=0 @ COL8(J,K)=0 @ RETURN
170 MNEG9(J,K)=0 @ TNG9(J,K)=0 @ TN9(J,K)=0 @ COL9(J,K)=0 @ RETURN
180 NUMPROB=0
190 PDS(0)=0 @ PRO(0)=0 @ FOR I=1 TO 200 @ CLEAR @ DISP "PREVIOUS COMBINATION 
200 PDS(I-1)
" @ PROBABILITY ":PRO(I-1) @ DISP "" 
210 DIS"SELECT (PORT,DEVICE,SOURCE) WHICH TRANSMITS TO (PORT,DEVICE,SOURCE)"
1110 DISP "FORMAT <PDS,PDS>, THESE COMBINATIONS MUST BE ENTERED IN ORDER."
1120 DISP "ENTER <0> IF FINISHED ENTERING PROBABILITIES";: INPUT PDS(I),TIMEST
1130 I=0
1140 IF PDS(I)=0 THEN 1260 ELSE NUMPROB=NUMPROB+1
1150 DISP "" ; DISP "PROBABILITY OF OCCURRENCE"; : INPUT PRO(I)
1160 GOSUB 1160 & GOTO 1250
1170 J=INT (PDS(I)/100000) & K=INT (PDS(I)/10000)-J=10
1180 L=-(J=100)-K=10+INT (PDS(I)/100) & M=-(J=1000)-K=100-L=10+INT (PDS(I)/100)
1190 N=-(J=10000)-K=1000-L=100-M=10+INT (PDS(I)/10)
1200 O=-(J=10000)-K=1000-L=1000-M=10+PDS(I)
1210 A=J & J=K & K=L & ON A GOSUB 480,520,560,600,640,680,720,760,800
1220 CP1=QQ=LT(A,J)
1230 CP2=QQ=LT(M,N)
1240 IF CP1>CP2 THEN TL=CP1 ELSE TL=CP2
1250 TL(I)=TSR+2000/CR+TL & RETURN
1260 NEXT I & PREJ=99 & PREK=95
1270 "NUM=0 & PREJ=99 & PREK=99 & CLEAR & FOR I=1 TO 200 & CLEAR & DISP "ENTER I
1280 "IJ-LM COMBINATIONS WHICH HAVE SOURCE DATA FILE DUMPS."
1290 "DISP "ENTER A <0> WHEN COMPLETED." & DISP "" & DISP "SOURCE TO DESTINATION
1300 ""; : INPUT DITFL(I) & IF DITFL(I)=0 THEN 1330
1310 WT(I)=0 & SNUM=SNUM+1 & J=INT (DITFL(I)/1000) & K=DITFL(I)-J=1000 & K=INT (K/10)
1320 IF J=PREJ AND K=PREK THEN BYTES(I)=BYTES(I-1) & TIM(I)=TIM(I-1) & GOTO 1320
1330 PREJ=J & PREK=K & DISP "# OF BYTES TO BE DUMPED" ; DISP "INPUT BYTES(I)
1340 DISP "TIME INTERVAL BETWEEN DUMPS (IN SECONDS)"; : INPUT TIM(I)
1350 NEXT I
1360 IF AN=2 THEN 1720
1370 !******************************************************************************
1380 ! READ OR WRITE DATA TO THE DATA FILE (GETDATA - 999 READ OR 888 WRITE)
1390 ! LINES 1340-1710
1400 !******************************************************************************
1410 CLEAR & CREATE DFNAMES,2123.8
1420 DFNMS=DFNAMES$;:D700" & ASSIGN# 1 TO DFNMS
1430 IF GETDATA=999 THEN READ# 1,1 ; HY ELSE PRINT# 1,1 ; HY
1440 II=1 & FOR J=1 TO HY & II=II+1
1450 IF GETDATA=999 THEN READ# 1,II ; DV(J) ELSE PRINT# 1,II ; DV(J)
1460 NEXT J & FOR I=1 TO HY & FOR J=1 TO DV(I)
1470 IF GETDATA=999 THEN READ# 1,II ; SR(I,J) ELSE PRINT# 1,II ; SR(I,J)
1480 NEXT J & NEXT I
1490 FOR I=1 TO HY & FOR J=1 TO DV(I) & FOR K=1 TO SR(I,J)
1500 IF GETDATA=888 THEN ON I GOSUB 480,520,560,600,640,680,720,760,800..
1440 II=II+1
1450 IF GETDATA=999 THEN READ# 1.II : QQ(I,K) ELSE PRINT# 1.II : F0
1460 II=II+1
1470 IF GETDATA=999 THEN READ# 1.II : LA(I,K) ELSE PRINT# 1.II : LA
1480 NEXT K & NEXT J & IF GETDATA=999 THEN ON 1 GOSUB 456.496.530.570.610.650.690 730.770
1490 NEXT I
1500 FOR I=1 TO HY & FOR J=1 TO DV(I) & II=II+1
1510 IF GETDATA=999 THEN READ# 1.II : D(I,J) ELSE PRINT# 1.II : D(I,J)
1520 NEXT J & NEXT I
1530 FOR I=1 TO HY & FOR J=II+1 & IF GETDATA=999 THEN READ# 1.II : W(I) ELSE PRINT# 1.II : W(I)
1540 NEXT I
1550 II=II+1 & IF GETDATA=999 THEN READ# 1.II : TSR ELSE PRINT# 1.II : TSR
1560 II=II+1 & IF GETDATA=999 THEN READ# 1.II : CR ELSE PRINT# 1.II : CR
1570 FOR I=1 TO HY & FOR J=1 TO DV(I) & II=II+1
1580 IF GETDATA=999 THEN READ# 1.II : R(I,J) & LT(I,J)=1/R(I,J) ELSE PRINT# 1.II : R(I,J)
1590 NEXT J & NEXT I
1600 II=II+1 & IF GETDATA=999 THEN READ# 1.II : NUMPROB ELSE PRINT# 1.II : NUMPROB
1610 FOR I=1 TO NUMPROB & II=II+1 & IF GETDATA=999 THEN READ# 1.II : PDS(I) ELSE PRINT# 1.II : PDS(I)
1620 II=II+1 & IF GETDATA=999 THEN READ# 1.II : PRO(I) ELSE PRINT# 1.II : PRO(I)
1630 IF GETDATA=999 THEN TIMES=0 & GOSUE 1160
1640 NEXT I
1650 II=II+1 & IF GETDATA=999 THEN READ# 1.II : SNUM ELSE PRINT# 1.II : SNUM
1660 FOR I=1 TO SNUM & W(T(I))=0 & II=II+1 & IF GETDATA=999 THEN READ# 1.II : DFL(I) ELSE PRINT# 1.II : DFL(I)
1670 II=II+1 & IF GETDATA=999 THEN READ# 1.II : BYTES(I) ELSE PRINT# 1.II : BYTE(I)
1680 II=II+1 & IF GETDATA=999 THEN READ# 1.II : TIM(I) ELSE PRINT# 1.II : TIM(I)
1690 NEXT I
1700 ASSIGN# 1 TO *
1710 GOSUB 960
1720 CLEAR & DISP TAB (20):"MENU" & DISP "1" & DISP "2" -- PRINT OUT CURRENT DATA & DISP "2" -- CORRECT DATA ERRORS & DISP "3" -- RUN PROGRAM" & DISP "4" & DISP "ENTER YOUR SELECTION":& INPUT, SELECT
1740 IF SELECT<1 OR SELECT>3 THEN 1720
1750 IF SELECT=3 THEN 1760 ELSE ON SELECT GOTO 3840.4210
1760 SAVEAT=99999 & SAVEA=0
1770 FOR I=1 TO HY & FOR J=1 TO DV(I) & FOR K=1 TO SR(I,J)
1780 ON I GOSUB 480.520.560.600.640.680.720.760.800
1790 IF LA>SAVELA THEN SAVELA=LA & SAVEIJK=I*100+J=10+K.
1800 IF ATR=0 THEN 1820
1810 IF ATR<SAVEATR THEN SAVEATR=ATR & SAVEATRIJK=I*100+J=10+K
1820 IF ATR=99999 THEN TLST=99999 ELSE TLST=0
1830 ON I GOSUB 1850,1860,1870,1880,1890,1900,1910,1920,1930
840 NEXT K & NEXT J & NEXT I & GOTO 1940
850 TLST1(J,K)=TLST & RETURN
860 TLST2(J,K)=TLST & RETURN
870 TLST3(J,K)=TLST & RETURN
880 TLST4(J,K)=TLST & RETURN
350 TLST5(J,K)=TLST & RETURN
300 TLST6(J,K)=TLST & RETURN
310 TLST7(J,K)=TLST & RETURN
320 TLST8(J,K)=TLST & RETURN
330 TLST9(J,K)=TLST & RETURN
340 PRINT " " & PRINT "FIRST USER IS ";SAVEIJK & ITERNUM=0 & NUMDEST=0
350 PRINT " " & PRINT "TOTAL TIME GONE BY ";SAVEATR & CN=0 & TTGB=SAVEATR
360 CLEAR & DISP "PRINTER MENU" & DISP " " & DISP "1 -- NO PRINUT GIVEN ON EACH CYCLE"
370 DISP "2 -- PRINTOUT GIVEN ON EACH CYCLE" & DISP "3 -- SPECIFY THE LAST CYCLES TO BE PRINTED" & DISP " "
380 INPUT PR & IF PR<1 OR PR>3 THEN CLEAR & GOTO 1960
390 IF PR=2 THEN PRINTER IS 701 ELSE PRINTER IS 1
00 IF PR=3 THEN DISP "ENTER THE LAST X CYCLES WHICH ARE TO BE PRINTED ";& INPUT NUMCYCLES
05 !*****************************************************************************
06 ! GENERATE A RANDOM NUMBER, THEN USE IT TO SELECT THE DESTINATION DEVICE
07 ! LINES 2010-2140
08 !*****************************************************************************
10 XXX=RND
120 IF PR=3 THEN 2030 ELSE 2040
30 IF ITERNUM>= NUMITER-2060 THEN PRINTER IS 701
40 PRINT " " & PRINT " " & PRINT "ITERATION NUMBER ";ITERNUM+I & PRINT " "
50 FOR I=1 TO NUMPROB
60 IJK=INT (PDS(I)/1000)
70 IF IJK=SAVEIJK THEN NUMDEST=NUMDEST+I & SAVEEND=I ELSE 2090
30 IF NUMDEST=1 THEN START-I & PROB=PRO(I) & PRINT "PROBABILITY ";PROB
90 NEXT I & PRINT "RANDOM NUMBER ";XXX
100 IF XXX<PROB THEN SOURCE=START & GOTO 2140
110 FOR I=START TO SAVEEND
120 IF XXX< PROB THEN SOURCE=I & GOTO 2140
230 PROB=PROB+PRO(I+1) & PRINT "PROBABILITY ";PROB & NEXT I
240 PRINT " " & PRINT "DESTINATION ";PDS(SOURCE)-SAVEIJK*1000
250 PRINT " " & PRINT "PORT,DEVICE,SOURCE TOTAL TIME GONE BY SINCE LAST TRANSMISSION" & TTGBSLT-SAVEATR
2160 TIMES{SOURCE}=TIMES{SOURCE} +
2170 FOR I=1 TO HY  @ FOR J=1 TO DV(I)  @ FOR K=1 TO SR(I,J)
2180 TLST=999999  @ ON I GOSUB 2210,2220,2230,2240,2250,2260,2270,2280,2290
2190 PRINT I;'";J;'";K;TAB (30);TLST
2200 NEXT K  @ NEXT J  @ NEXT I  @ GOTO 2300
2210 IF TLST1(J,K)=999999 THEN RETURN ELSE TLST1(J,K)=TLST1(J,K)+TTGS|TLST
2220 IF TLST2(J,K)=999999 THEN RETURN ELSE TLST2(J,K)=TLST2(J,K)+TTGS|TLST
2230 IF TLST3(J,K)=999999 THEN RETURN ELSE TLST3(J,K)=TLST3(J,K)+TTGS|TLST
2240 IF TLST4(J,K)=999999 THEN RETURN ELSE TLST4(J,K)=TLST4(J,K)+TTGS|TLST
2250 IF TLST5(J,K)=999999 THEN RETURN ELSE TLST5(J,K)=TLST5(J,K)+TTGS|TLST
2260 IF TLST6(J,K)=999999 THEN RETURN ELSE TLST6(J,K)=TLST6(J,K)+TTGS|TLST
2270 IF TLST7(J,K)=999999 THEN RETURN ELSE TLST7(J,K)=TLST7(J,K)+TTGS|TLST
2280 IF TLST8(J,K)=999999 THEN RETURN ELSE TLST8(J,K)=TLST8(J,K)+TTGS|TLST
2290 IF TLST9(J,K)=999999 THEN RETURN ELSE TLST9(J,K)=TLST9(J,K)+TTGS|TLST
2300 GOSUB 2310  @ GOTO 3020
2305 !*****************************************************************************
2306 ! FIND SMALL & SMALL2 TO BE USED TO CHECK FOR COLLISIONS AND NRTT
2307 ! LINES 2310-2540
2308 !*****************************************************************************
2310 PRINT " "  @ PRINT "PORT,DEVICE,SOURCE           REMAINING TIME TO TRANSMIT (NO
2311 ALIZED FASHION)"
2320 SMALL=999988  @ SAVESM=999988  @ SMALL2=99988  @ SAVESM2=99988
2330 FOR I=1 TO HY  @ FOR J=1 TO DV(I)  @ FOR K=1 TO SR(I,J)
2340 ON I GOSUB 480,520,560,600,640,680,720,760,800
2350 ON I GOSUB 2370,2380,2390,2400,2410,2420,2430,2440,2450
2360 GOTO 2460
2370 TLST=TLST1(J,K)  @ RETURN
2380 TLST=TLST2(J,K)  @ RETURN
2390 TLST=TLST3(J,K)  @ RETURN
2400 TLST=TLST4(J,K)  @ RETURN
2410 TLST=TLST5(J,K)  @ RETURN
2420 TLST=TLST6(J,K)  @ RETURN
2430 TLST=TLST7(J,K)  @ RETURN
2440 TLST=TLST8(J,K)  @ RETURN
2450 TLST=TLST9(J,K)  @ RETURN
2460 IF ATR=999999 THEN NRTT=999999 ELSE NRTT=ATR-TLST
2470 IF CLSN$="Y" THEN SUB=-1 ELSE SUB=1
2480 ON I GOSUB 2550,2600,2650,2700,2750,2800,2850,2900,2950
2490 IF CLSN$="Y" THEN 2510
2500 PRINT I:" ";J;" ";I;TAB(30);NRTT
510 IF NRTT<SMALL THEN CH1=SMALL @ SMALL.+NRTT @ CH2=SAMESM @ SAVESM=I*100+J*10+
6 @ EQS="Y"
520 IF CH1<SMALL2 THEN SMALL2=CH1 @ SAVESM2=CH2
530 IF NRTT=SMALL AND NRTT<SMALL2 AND EQS="N" THEN SMALL2=NRTT @ SAVESM2=I*100+
9*10+K
540 EQS="N" @ NEXT K @ NEXT J @ NEXT I @ EQS="N" @ RETURN
550 NRTT1(J,K)=NRTT @ IF NRTT<0 THEN TNG1(J,K)=TNG1(J,K)+NRTT @ TN1(J,K)=TN1(J,
)+SUB
560 IF CLSN$ <> "Y" THEN 2580
570 IF NRTT<0 THEN TNG1(J,K)=TNG1(J,K)-2*NRTT
580 IF NRTT<MNEG1(J,K) THEN MNEG1(J,K)=NRTT
590 RETURN
560 NRTT2(J,K)=NRTT @ IF NRTT<0 THEN TNG2(J,K)=TNG2(J,K)+NRTT @ TN2(J,K)=TN2(J,
)+SUB
510 IF NRTT<MNEG2(J,K) THEN MNEG2(J,K)=NRTT
520 IF CLSN$ <> "Y" THEN 2640
530 IF NRTT<0 THEN TNG2(J,K)=TNG2(J,K)-2*NRTT
540 RETURN
550 NRTT3(J,K)=NRTT @ IF NRTT<0 THEN TNG3(J,K)=TNG3(J,K)+NRTT @ TN3(J,K)=TN3(J,
)+SUB
560 IF CLSN$ <> "Y" THEN 2680
570 IF NRTT<0 THEN TNG3(J,K)=TNG3(J,K)-2*NRTT
580 IF NRTT<MNEG3(J,K) THEN MNEG3(J,K)=NRTT
590 RETURN
700 NRTT4(J,K)=NRTT @ IF NRTT<0 THEN TNG4(J,K)=TNG4(J,K)+NRTT @ TN4(J,K)=TN4(J,
)+SUB
710 IF CLSN$ <> "Y" THEN 2730
720 IF NRTT<0 THEN TNG4(J,K)=TNG4(J,K)-2*NRTT
730 IF NRTT<MNEG4(J,K) THEN MNEG4(J,K)=NRTT
740 RETURN
750 NRTT5(J,K)=NRTT @ IF NRTT<0 THEN TNG5(J,K)=TNG5(J,K)+NRTT @ TN5(J,K)=TN5(J,
)+SUB
760 IF CLSN$ <> "Y" THEN 2780
770 IF NRTT<0 THEN TNG5(J,K)=TNG5(J,K)-2*NRTT
2780 IF NRTT=MNEG5(J,K) THEN MNEG5(J,K)=NRTT
2790 RETURN
2800 NRTT6(J,K)=NRTT @ IF NRTT<0 THEN TNG6(J,K)=TNG6(J,K)+NRTT @ TN6(J,K)=TN6(J,
)+SUB
2810 IF CLSN$ <> "Y" THEN 2830
2820 IF NRTT<0 THEN TNG6(J,K)=TNG6(J,K)-2*NRTT
2830 IF NRTT<MNEG6(J,K) THEN MNEG6(J,K)=NRTT
2840 RETURN
2850 NRTT7(J,K)=NRTT  IF NRTT<0 THEN TNG7(J,K)=TNG7(J,K)+NRTT  IF N7(J,K)=N7(J, )
2860 IF NRTT<MNEG7(J,K) THEN MNEG7(J,K)=NRTT
2870 IF CLSN$ <> "Y" THEN 2890
2880 IF NRTT<0 THEN TNG7(J,K)=TNG7(J,K)-2*NRTT
2890 RETURN
2900 NRTT8(J,K)=NRTT  IF NRTT<0 THEN TNG8(J,K)=TNG8(J,K)+NRTT  IF T8(J,K)=T8(J, )
2910 IF NRTT<MNEG8(J,K) THEN MNEG8(J,K)=NRTT
2920 IF CLSN$ <> "Y" THEN 2940
2930 IF NRTT<0 THEN TNG8(J,K)=TNG8(J,K)-2*NRTT
2940 RETURN
2950 NRTT9(J,K)=NRTT  IF NRTT<0 THEN TNG9(J,K)=TNG9(J,K)+NRTT  IF T9(J,K)=T9(J, )
2960 IF NRTT<MNEG9(J,K) THEN MNEG9(J,K)=NRTT
2970 IF CLSN$ <> "Y" THEN 2990
2980 IF NRTT<0 THEN TNG9(J,K)=TNG9(J,K)-2*NRTT
2990 RETURN
3000  *********************************************************************
3010 ! THE COLLISION ALGORITHM.
3020 ! LINES 3020 - 3300
3030 !  *********************************************************************
3040 AF$="F"  IF INT ((PDS(SOURCE)-SAVESM*1000)/100)=INT (SAVESM/100) THEN AF$=
3050 ! "I"  IF PRINTER IS 701
3060 ! IF AF$="I" THEN PRINT "TRANSMISSION ON SAME A400 SOURCE/DEST = ":PDS(SOURCE
3070 ! & CLSN$="N" & GOTO 3260
3080 ! IF CLSN$="Y" THEN 3090
3090 ! IF ABS (SMALL-SMALL2)<= .001 THEN PRINTER IS 701 ELSE GOTO 3260
3100 PRINT "A COLLISION OCCURRED BETWEEN ";SAVESM;" AND ";SAVESM2;
3110 I=INT (SAVESM/100)  & L=INT (SAVESM2/100)
3120 IF W(L)< W(L) THEN 3090
3130 SAVEIJK=SAVESM2 & NUMDEST=0 & SAVETR=W(L) & TTBG=TTBG+W(L) & TIMESL=SOURC
3140 =TIMESTT(SOURCE)-1 & CLSN$="Y"
3150 PROB=0 & CLSN$="Y" & GOSUB 2320 & CLSN$="N" & GOTO 3260
3160 CN=CN+1 & PRINT "COLLISION NUMBER ";CN
3170 I=INT (SAVESM/100)  & J=INT (SAVESM/10)-I*10  & K=SAVESM-I*100-J*10
3180 ON I GOSUB 3150,3160,3170,3180,3190,3200,3210,3220,3230
3190 ON J GOSUB 3150,3160,3170,3180,3190,3200,3210,3220,3230
3200 GOTO 3240
3150  COL(J,K)=COL(J,K)+1 @ RETURN
3160  COL2(J,K)=COL2(J,K)+1 @ RETURN
3170  COL3(J,K)=COL3(J,K)+1 @ RETURN
3180  COL4(J,K)=COL4(J,K)+1 @ RETURN
3190  COL5(J,K)=COL5(J,K)+1 @ RETURN
3200  COL6(J,K)=COL6(J,K)+1 @ RETURN
3210  COL7(J,K)=COL7(J,K)+1 @ RETURN
3220  COL8(J,K)=COL8(J,K)+1 @ RETURN
3230  COL9(J,K)=COL9(J,K)+1 @ RETURN
240  IF CLNS="Y" THEN DT(SOURCE)=T(SOURCE) @ CLNS="N" ELSE DT(SOURCE)=T(SOURCE)+
250  GOTO 3300
260  DT(SOURCE)=T(SOURCE)
270  PRINT "NO COLLISION OCCURED" @ IF PR=1 THEN PRINTER IS 1
280  IF PR=3 THEN 3290 ELSE 3300
290  IF ITERNUM=NUMITER-NUMCYCLES THEN 3300 ELSE PRINTER IS 1
300  PRINT "DATA TRANSFER" ; DT(SOURCE) @ IF AF$="T" THEN AF$="F" @ GOTO 3320
310  BBB=BBT+DT(SOURCE)
320  PRINT "" @ PRINT "PORT,DEVICE,SOURCE" ACCUMULATED WAITING TIMES SINCE
330  FOR I=1 TO HI @ FOR J=1 TO DV(I) @ FOR K=1 TO 3R(I,J)
340  IF I*100+J*10+K=SAVEIJK THEN TLST=0 ELSE GOTO 3370
350  ON I GOSUB 1850,1860,1870,1880,1890,1900,1910,1920,1930
360  GOTO 3390
370  TTBGS=DT(SOURCE) @ TLST=999999
380  ON I GOSUB 2210,2220,2230,2240,2250,2260,2270,2280,2290
390  PRINT I; ";J; ";K;TAB(30);TLST
400  NEXT K @ NEXT J @ NEXT I
410  TTBG=TTBG+DT(SOURCE)
420  PRINT "DATA TRANSFER COMPLETED: TOTAL TIME GONE BY ";TTBG
430  GOSUB 2310
440  PRINT "NEXT TRANSMISSION REQUEST BY ";SAVESM @ SAVEIJK=SAVESM
450  TTNR=SMALL
460  IF TTNR<0 THEN 3480
470  TTBG=TTBG+TTNR
480  TTBGS=TTNR @ SAVEM2=0 @ SMALL=0 @ SMALL2=0
481  ! PRINT OUT CURRENT ENTRY DATA - EITHER FROM FILE OR USER ENTERED
490  PRINT "TOTAL TIME THAT WILL BE GONE BY WHEN ";SAVESM;" STARTS TO TRANSMIT "
500  TTBG
505  NUMDEST=0 @ PROB=0 @ SAVEATR=TTNR @ SAVEM=0
506  ! *******************************************
507  ! SOURCE DUMP ALGORITHM LINES 3510 - 3590
508  ! *******************************************
3510 FOR H=1 TO SNUM 9 IF TIM(H)<= TTGB-WT(H) THEN 3520 ELSE 3590
3520 WT(H)=TTGB & J=INT (DTFL(H)/1000) & K=INT ((DTFL(H)-J*1000)/100) & L=INT ((
TFL(H)-J*1000-K*100)/10) & M=DTFL(H)-J*1000-K*100-L*10
3530 CP1=BYTES(H)=LT(J,K) & CP2=BYTES(H)=LT(L,M)
3540 Printer IS 701 & PRINT "" & PRINT "SOURCE ";&J,"",;K:" DUMPED TO DESTIN
3550 Printer IS 701 & PRINT '"" TOTAL TIME GONE BY (BEFORE THE DUMP) "";TTGB
3560 IF CP1>CP2 THEN TL=CP1 ELSE TL=CP2
3570 TRANS=TSR+2000/CR+TL & TTGB=TTGB+TRANS & BBT=BBT+TRANS & TGBSLT=TRANS
3580 PRINT "TOTAL TIME GONE BY (AFTER THE DUMP) "";TTGB & PRINT "" & IF PR=1 THE
PRINTER IS 1
3590 IF PR=3 THEN 3557 ELSE 3560
3557 IF ITERNUM>= NUMITER-NUMCYCLES THEN 3560 ELSE PRINTER IS 1
3560 FOR I=1 TO HY & FOR J=1 TO DV(I) & FOR K=1 TO SR(I,J)
3570 ON I GOSUB 2210,2220,2230,2240,2250,2260,2270,2280,2290
3580 NEXT K & NEXT J & NEXT I
3590 NEXT H & TGBSLT=TNR
3600 IF ITERNUM=NUMITER-1 THEN 3610 ELSE ITERNUM=ITERNUM+1 & GOTO 2010
3610 Printer IS 701
3615 !******************************************************************************************
3616 ! THE FINAL PRINTOUT GIVING A SUMMARY OF THE RUN
3617 ! LINES 3620 - 3830
3618 !******************************************************************************************
3620 TOT=0 & PRINT "" & PRINT "SOURCE TO DESTINATION" # OF TIMES TRANSMIT
3630 FOR I=1 TO NUMPROB & PRINT PDS(I);TAB (30);TIMESTT(I); TOT=TOT+TIMESTT(I)
3640 NEXT I
3650 PRINT "TOTAL";TAB (30);TOT
3660 PRINT "" & PRINT "PORT,DEVICE,SOURCE LONGEST WAITING TIME AVG WAITIN
3670 TIME COLLISIONS" & PRINT "" & PRINT "" & PRINT "" & PRINT "" & PRINT "" & PRINT ""
3680 FOR I=1 TO HY & FOR J=1 TO DV(I) & FOR K=1 TO SR(I,J)
3690 ON I GOSUB 3720,3730,3740,3750,3760,3770,3780,3790,3790,3806
3690 IF TN=0 THEN AVG=0 ELSE AVG=TNEG/TN
3700 PRINT I;"" ;J;";K;TAB (25);MNEG;TAB (46);AVG;TAB (68);COLL
3710 NEXT K & NEXT J & GOTO 3810
3810 PRINT "TOTAL TIME GONE BY = ";TTGB
320 PRINT "BUSY TIME = ";BBT
330 END
340 ! ******************************************************
345 ! PRINT OUT CURRENT DATA - EITHER FROM FILE OR ENTERED BY USER
350 ! ******************************************************
350 PRINT "# OF HYPERCHANNEL PORTS = ";HY @ PRINT " ";
370 FOR I=1 TO HY @ PRINT "# OF DEVICES FOR PORT ";I;" = ";DV(I) @ NEXT I @ I=1
380 PRINT " ";
380 FOR K=1 TO DV(I)
390 PRINT "# OF SOURCES FOR PORT ";I;" DEVICE ";K;" = ";SR(I,K)
390 NEXT K @ NEXT I @ IF I=HY THEN 3890 ELSE I=0 @ PRINT " ";
410 PRINT "PORT,DEVICE,SOURCE AVERAGE ARRIVAL RATE # OF BYTES ACCU-
420 LATED"
430 I=I+1 @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
430 ON J GOSUB 480,520,560,600,640,680,720,760,800
430 PRINT I;" ";J;" ";K:TAB (35);LA:TAB (60);QQ
450 NEXT K @ NEXT J
460 IF I=HY THEN 3970 ELSE GOTO 3920
470 PRINT " ";
480 PRINT "PROPORTION DISTANCE BETWEEN PORT ";I;" AND DEVICE ";J;" = ";D(I,J)
490 NEXT J @ NEXT I @ PRINT " ";
500 FOR I=1 TO HY
510 PRINT "WAIT TIME OF PORT ";I;" BEFORE TRANSMISSION REQUEST (IN EVENT OF A C-
520 LISION) ";W(I)
530 NEXT I
540 PRINT " ";
550 PRINT "CHANNEL SETUP AND RELEASE TIME = ";TSR @ PRI-
560 T " ";
570 PRINT "CHANNEL DATA TRANSFER RATE = ";CR
580 PRINT " ";
590 PRINT "PORT,DEVICE TRANSFER RATE LOAD T-
600 MES"
610 FOR I=1 TO HY @ FOR J=1 TO DV(I)
620 PRINT I;" ";J:TAB (30):R(I,J);TAB (54);LT(I,J)
630 NEXT J @ NEXT I
640 PRINT " ";
650 PRINT "COMBINATIONS (IJK - LMN) PROBABILITY TIME TO TRANSMIT-
660 (IJK) BYTES (IJK - LMN)"
670 FOR I=1 TO NUMPROB
690 NEXT I
700 PRINT " ";
710 PRINT "PORT,DEVICE.SOURCE AVERAGE TIME TO NEXT TRANSMISSION-
720 REQUEST"
730 FOR I=1 TO HY @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J)
740 ON J GOSUB 480,520,560,600,640,680,720,760,800
4160 PRINT I:”; “;J:”;K:0 TAB (30); ATR
4170 NEXT K @ NEXT J @ NEXT I
180 PRINT ”; PRINT “I-J-LM DATA FILE DUMP; BYTES TO BE DUMPED; TIME BETWEEN
DUMPS <IN SECONDS>”
190 FOR I=1 TO SNUM @ PRINT DTFL(I);TAB(30);BYTES(I);TAB(50);TIM(I)
200 NEXT I @ GOTO 1720
210 !******************************************************************************
215 ! THE EDIT ROUTINE - SAVES CORRECTED DATA TO FILE
216 ! LINES 4220 - 4950
217 !******************************************************************************
220 CLEAR & DISP ”EDIT MENU” & DISP ” ”
230 DISP ”1 -- AVERAGE ARRIVAL RATES”
240 DISP ”2 -- NUMBER OF BYTES ACCUMULATED”
250 DISP ”3 -- PROPAGATION DISTANCES”
260 DISP ”4 -- WAIT TIMES”
270 DISP ”5 -- CHANNEL SETUP AND RELEASE TIME”
280 DISP ”6 -- CHANNEL DATA TRANSFER RATE”
290 DISP ”7 -- TRANSFER RATES”
300 DISP ”8 -- PROBABILITY DATA”
310 DISP ”9 -- SOURCE DATA DUMPS”
320 DISP ”10 -- TERMINATE EDIT MODE”
330 DISP ” ” & DISP ”ENTER YOUR SELECTION ”; @ INPUT EDT
340 IF EDT<1 OR EDT>10 THEN 4210
350 ON EDT GOTO 4360,4450,4540,4590,4630,4660,4690,4760,4850,4950
360 FOR I=1 TO HY @ CLEAR @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ CLEAR
370 ON I GOSUB 480.520,560,600,640,680,720,760,800
380 DISP ”PORT ”;I:”;DEVICE ”;J:”;SOURCE ”;K @ DISP & DISP
390 LA(J,K)=LA @ QQ(J,K)=QQ
400 DISP ”AVERAGE ARRIVAL RATE =”;LA
410 Q$=” ” & DISP ” ” & DISP ” ” & DISP ”PRESS ENTER TO RETAIN CURRENT DATA OR
420 ITER THE CORRECT VALUE ”; @ INPUT Q$
430 IF Q$ <> ”” THEN LA(J,K)=VAL (Q$)
440 NEXT K @ NEXT J @ ON I GOSUB 450,490,530,570,610,650,690,730,770
450 NEXT I @ GOTO 4210
460 FOR I=1 TO HY @ CLEAR @ FOR J=1 TO DV(I) @ FOR K=1 TO SR(I,J) @ CLEAR
470 ON I GOSUB 480,520,560,600,640,680,720,760,800
480 DISP ”PORT ”;I:”;DEVICE ”;J:”;SOURCE ”;K @ DISP & DISP
490 QQ(J,K)=QQ @ LA(J,K)=LA
500 DISP ”NUMBER OF BYTES ACCUMULATED =”;QQ
510 Q$=” ” & DISP ” ” & DISP ” ” & DISP ”PRESS ENTER TO RETAIN CURRENT DATA OR
520 ITER THE CORRECT VALUE ”; @ INPUT Q$
530 IF Q$ <-> ”” THEN QQ(J,K)=VAL (Q$)
4520 NEXT K @ NEXT J @ ON I GOSUB 450,490,530,570,610,650,690,730,770
4530 NEXT I @ GOTO 4840
540 CLEAR @ FOR I=1 TO HY @ FOR J=1 TO DV(I) @ CLEAR
550 DISP "PROPAGATION DISTANCE BETWEEN PORT ",:I:" AND DEVICE ",:J:" IN MICROSEC
DS = ",:D(I,J) @ QS=" 
560 DISP "" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VA
UE ",: INPUT QS
570 IF QS <> "" THEN D(I,J)=VAL (QS)
580 NEXT J @ NEXT I @ GOTO 4210
590 CLEAR @ FOR I=1 TO HY @ CLEAR @ DISP "W AIT TIME OF PORT ",:I:" BEFORE TRANSM
SSION REQUEST = ",:W(I)
600 DISP "" @ QS="" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE C
RECT VALUE ",: INPUT QS
610 IF QS <> "" THEN W(I)=VAL (QS)
620 NEXT I @ GOTO 4210
630 CLEAR @ DISP "CHANNEL SETUP AND RELEASE TIME = ",:TSR
640 QS="" @ DISP "" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE P
PER VALUE ",: INPUT QS@ IF QS <> "" THEN TSR=VAL (QS)
650 GOTO 4840
660 CLEAR @ DISP "CHANNEL DATA TRANSFER RATE = ",:CR @ DISP ""
670 QS="" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE PROPER VA
LUE ",: INPUT QS@ IF QS <> "" THEN CR=VAL (QS)
680 GOTO 4840
690 CLEAR @ FOR I=1 TO HY @ FOR J=1 TO DV(I) @ CLEAR
700 DISP "RATE AT WHICH DATA IS TRANSFERED FROM DEVICE ",:J:" TO PORT ",:I:" = ",:I,J)
710 QS="" @ DISP "" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE C
RECT VALUE ",: INPUT QS
720 IF QS <> "" THEN R(I,J)=VAL (QS)
730 LT(I,J)=1/R(I,J)
740 NEXT J @ NEXT I
750 GOTO 4840
760 CLEAR @ FOR I=1 TO NUMPROB
770 CLEAR @ DISP "(PORT,DEVICE,SOURCE) TRANSMITS TO (PORT,DEVICE,SOURCE). CODE ",:PDS(I) @ DISP ""
780 QS="" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
E ",: INPUT QS
790 IF QS <> "" THEN PDS(I)=VAL (QS)
800 DISP "" @ DISP "PROBABILITY = ",:PRJ(I) @ DISP ""
810 QS="" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
E ",: INPUT QS
820 IF QS <> "" THEN PRO(I)=VAL (QS)
1830 NEXT I
1840 FOR I=1 TO NUMPROB $ GO To SUB 1160 $ NEXT I $ GOTO 4210
1850 CLEAR @ FOR I=1 TO SNUM @ CLEAR @ DISP "(I) - LINE) SOURCE DATA FILE DUMPS TO
DESTINATION LM. CODE = "@ DTFL(I) @ DISP " "
1860 Q$="" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
": @ INPUT Q$
1870 IF Q$ <> "" THEN DTFL(I)=VAL (Q$)
1880 DISP "" @ DISP "# OF BYTEST TO BE DUMPED ";BYTES(I) @ DISP " "
1890 Q$="" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
": @ INPUT Q$
1900 IF Q$ <> "" THEN BYTES(I)=VAL (Q$)
1910 DISP "" @ DISP "TIME INTERVAL BETWEEN DUMPS ";TIM(I) @ DISP " "
1920 Q$="" @ DISP "PRESS ENTER TO RETAIN CURRENT DATA OR ENTER THE CORRECT VALU
": @ INPUT Q$
1930 IF Q$ <> "" THEN TIM(I)=VAL (Q$)
1940 NEXT I $ GOTO 4210
1950 GETDATA=888 @ ON AN GOTO 1350.1720.1350
APPENDIX II

PASCAL SIMULATIONS ALGORITHM PROGRAM

A listing of the PASCAL simulation algorithm program is presented in this appendix. The program contains comments, but is not considered to be user oriented at this point. This program is FORTRAN based, but requires software support for PASCAL in the host computer.
**Simulation of a HYPERchannel Network:**
A Model of the Huntsville Operations Support Center

**AUTHOR:**
John D. Pauldin

**Mississippi State University**

**May, 1983**

**ABSTRACT:**

---

```lisp
program MODL(INPUT,OUTPUT,TERMINAL,DESCRIPT FILE,AUX OUT):

const
MAX NUM SOURCES = 4;
MAX NUM ADAPTERS = 4;
NUM OF ADAPTERS = 4;
CPUS = 4;
NUM DEV = 11;
TRUNK OVERHEAD = 0.000025; (* Trunk set-up and release time *)
TRUNK RATE = 5000; (* Trunk to rate in bits/second *)
ONE = 1;
ZERO = 0.001; (* Increment to determine collisions *)

SOURCE DESCRIPTIONS = record
   TX RATE : real; (* Avg number of bytes/sec *)
end-

DEVICE RECORD = record
   OPEN : boolean; (* TRUE if device attached to port *)
   DEV NUM : integer; (* Device number *)
   BUFFER SIZE : integer; (* Number of bits accumulated from adapter before the device requests trunk transmission *)
   NUM OF SOURCES : integer;
   SOURCE : array[1..MAX NUM SOURCES] of SOURCE DESCRIPTIONS;
   TX RATE : real; (* I/O port transfer rate from device to adapter in bytes/sec *)
   LOAD TIME : real; (* Time for device to load adapter buffer *)
   NEXT : real; (* Time until next transmission *)
   LAST TRUNK TX : real; (* Time between tx requests based on aggregate source transmission rates from offset sources *)
   TX INTRVL : real; (* Device transmission time *)
   RX CT : integer; (* Device collision tally *)
   WATT CT : integer; (* Device wait tally *)
   TX CT : integer; (* Device receiving tally *)
   TX TIME : real; (* Time device spent in transmissions *)
```
function TRANSFER_TIME(SENDER, RECEIVER, DEVICE_RECORD): real;
(* Calculates time required to Transmit a Message *)
(* from a sender to a receiver *)
(* Trunk overhead time includes the fixed delay for each *)
(* adapter *)
(* length of time required by a sending *)
(* device to receive a response frame *)
(
var
SLOWER_RATE: real;
PKCT_CT: integer;
)
begin
SLOWER_RATE := SENDER.TRNS_RATE;
if RECEIVER.TRNS_RATE < SLOWER_RATE then
SLOWER_RATE := RECEIVER.TRNS_RATE;
PKCT_CT := round(SENDER.BUFFER_SIZE/2048);
case PKCT_CT of
0: T := TRUNK_OVERHEAD;
1: T := 2048/4096 + TRUNK_OVERHEAD;
otherwise: T := 2048/4096 + TRUNK_RATE + TRUNK_OVERHEAD;
end;
TRANSFER_TIME := T;
end;
(* procedure UPDATE_CLOCKS(COND, CLOCK_COND) *)
(* Updates both the system clocks and device clocks whenever a system action has occurred. *)
var
TA, TD, RA, RD; 1, J; integer;
DELAY_TIME, TIME, TI: real;
begin
A is transmitter adapter no., D is device no. *)
T := (TMTR.DEV.NUM mod 100) div 10;
RA := (RCVR.DEV.NUM mod 100) div 10;
TD := TMTR.DEV.NUM mod 10;
RD := RCVR.DEV.NUM mod 10;
(* Update system clock *)
CASE COND OF
NORMAL: begin
TIME := TRANSFER_TIME(TMTR, RCVR);
BITS_TX := BITS_TX + TMTR.BUFFER_SIZE;
(* Update each device clock *)
end;
for \( i = 1 \) to NUM_OF_ADAPTERS do
  for \( j = 1 \) to \( 4 \) do
    with ADAPTER(\( i \)).DEVICE(\( j \)) do
      if (not OPEN) and (NOT IRA=\( i \)) and (ID=\( j \)) then
        begin
          NEXT TX <= CURRENT TIME then
            begin
              DELAY TIME:= ADAPTER(\( i \)).PRIOR.DELAY + ADAPTER(\( i \)).END_DELAY;
              NEXT TX:= CURRENT TIME + DELAY.TIME;
              WAIT.TALLY:= WAIT.TALLY + 1;
              WAIT.CT:= WAIT.CT + 1;
              WAIT.TIME:= WAIT.TIME + DELAY.TIME + TIME;
            end;
          if PRINT.ENABLED then
            begin
              SRTM := [ADAPTER(\( i \)).DEVICE(\( j \))];
            end;
          (* Reset transmitter clock *)
          begin
            with ADAPTER(\( i \)).DEVICE(\( j \)) do
              begin
                NEXT TX:= CURRENT_TIME + TX_INTRVL;
                LAST_TRUNK IX:= CURRENT_TIME;
                TX.CT:= TX.CT + 1;
                TX.TIME:= TX.TIME + TIME;
              end;
            with ADAPTER(\( i \)).DEVICE(\( j \)) do
              begin
                RX.CT:= RX.CT + 1;
                RX.TIME:= RX.TIME + TIME;
              end;
          (* with ADAPTER(\( i \)).DEVICE(\( j \)) *)
          TIME_ACTIVE:= TIME_ACTIVE + TRANSFER_TIME(TXTR, RXVR);
          end;
        end;
      end;
    end;
  end;
end;

COLLISION := begin
  with ADAPTER(\( i \)).DEVICE(\( j \)) do
    begin
      TX:= COLLECPS + (UNIFORM(ZERO, ONE))/(1 - COLL_EPS);
      NEXT TX:= NEXT TX + TX;
      COLLECT:= COLLECT + 1;
      COLLECT:= COLLECT + 1;
    end;
  (* with ADAPTER(\( i \)).DEVICE(\( j \)) *)
end;

OTHER := begin
  (* other case *)
end;

end; (* case *)

end; (* UPDATE.CLOCKS *)

function TX_INTERVAL(DEVICE : DEVICE_RECORD) : real;
begin
  war := integer;
  AGGREGATE_RATE := real;
  for \( i = 1 \) to DEVICE.NUM_OF_SOURCES do
if not O in then
if DEV_NUM div 100 = 1 then
be:
X = true;
end; (* for each i *)
end; (* with *)
end; (* while L<4 and not FOUND *)
end;
end; (* while K< num adapters and not FOUND *)
RECEIVER := ADAPTER[A].DEVICE[D];
end; (* PICK A RECEIVER *)

begin
A_COLLISION := false;
for j:=1 to NUM_OF_ADAPTERS do
for i:=1 to 4 do
with ADAPTER[i].DEVICE[i] do
if not OPEN then
if INIT.DEV_NUM <> DEV_NUM then
if abs(INIT.NEXT_TA - NEXT_TA) < COLL_EPS then
A_COLLISION := true;
end; (* COLLISION DETECTION *)

procedure ACTIVITY_SUMMARY;

begin
write(AUX_OUT, "DEVICE ACTIVE TIME = ");
wait PC_DEV;
write(AUX_OUT, "TIME IN " :7, "TIME", "TIME" :10);
end;
```plaintext
Listing 1: Code snippet from a software program

```
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0131</td>
<td>J := J + 1;</td>
</tr>
<tr>
<td>0132</td>
<td>end; (* while J &lt;= *)</td>
</tr>
<tr>
<td>0133</td>
<td>if J = 1 then J := J + 1;</td>
</tr>
<tr>
<td>0134</td>
<td>while (J &gt; 1) and (J &lt;= MAX_NUM_ADAPTERS) do</td>
</tr>
<tr>
<td>0135</td>
<td>begin</td>
</tr>
<tr>
<td>0136</td>
<td>write(AUX_OUT, &quot;-:-.Proportion distance to adapter &quot; + J + &quot;, M, sec&quot;);</td>
</tr>
<tr>
<td>0137</td>
<td>LCT := CT + 1;</td>
</tr>
<tr>
<td>0138</td>
<td>J := J + 1;</td>
</tr>
<tr>
<td>0139</td>
<td>end; (* while J &gt; 1 AND J &lt;= MAX_NUM_ADAPTERS *)</td>
</tr>
<tr>
<td>0140</td>
<td>write(AUX_OUT);</td>
</tr>
<tr>
<td>0141</td>
<td>LCT := CT + 1;</td>
</tr>
<tr>
<td>0142</td>
<td>write(AUX_OUT, &quot;-:-.Priority delay: &quot;);</td>
</tr>
<tr>
<td>0143</td>
<td>write(AUX_OUT,PRIORITY_DELAY:9:7, sec&quot;);</td>
</tr>
<tr>
<td>0144</td>
<td>LCT := CT + 1;</td>
</tr>
<tr>
<td>0145</td>
<td>write(AUX_OUT, &quot;:-:-.End delay: &quot;);</td>
</tr>
<tr>
<td>0146</td>
<td>write(AUX_OUT, END_DELAY:9:7, sec&quot;);</td>
</tr>
<tr>
<td>0147</td>
<td>write(AUX_OUT);</td>
</tr>
<tr>
<td>0148</td>
<td>LCT := CT + 2;</td>
</tr>
<tr>
<td>0149</td>
<td>(* Begin individual device descriptions *)</td>
</tr>
<tr>
<td>0150</td>
<td>for J := 1 to 4 do</td>
</tr>
<tr>
<td>0151</td>
<td>begin</td>
</tr>
<tr>
<td>0152</td>
<td>with ADAPTER[1..DEVICE[3]] do</td>
</tr>
<tr>
<td>0153</td>
<td>begin</td>
</tr>
<tr>
<td>0154</td>
<td>write(AUX_OUT);</td>
</tr>
<tr>
<td>0155</td>
<td>LCT := CT + 1;</td>
</tr>
<tr>
<td>0156</td>
<td>write(AUX_OUT, &quot;:-:-.Device number &quot;, DEV_NUM:2);</td>
</tr>
<tr>
<td>0157</td>
<td>write(AUX_OUT, &quot;-:-.Status: &quot;);</td>
</tr>
<tr>
<td>0158</td>
<td>if OPEN then begin write(AUX_OUT, &quot;OPEN&quot;);</td>
</tr>
<tr>
<td>0159</td>
<td>LCT := CT + 1; end</td>
</tr>
<tr>
<td>0160</td>
<td>else write(AUX_OUT, &quot;CLOSED&quot;);</td>
</tr>
<tr>
<td>0161</td>
<td>if not(OPEN) then</td>
</tr>
<tr>
<td>0162</td>
<td>begin</td>
</tr>
<tr>
<td>0163</td>
<td>write(AUX_OUT, &quot;:-:-.2, Device number &quot;, DEV_NUM:2);</td>
</tr>
<tr>
<td>0164</td>
<td>write(AUX_OUT, &quot;-:-.Status: &quot;);</td>
</tr>
<tr>
<td>0165</td>
<td>if OPEN then begin write(AUX_OUT, &quot;OPEN&quot;);</td>
</tr>
<tr>
<td>0166</td>
<td>LCT := CT + 1; end</td>
</tr>
<tr>
<td>0167</td>
<td>else write(AUX_OUT, &quot;CLOSED&quot;);</td>
</tr>
<tr>
<td>0168</td>
<td>if not(OPEN) then</td>
</tr>
<tr>
<td>0169</td>
<td>begin</td>
</tr>
<tr>
<td>0170</td>
<td>write(AUX_OUT, &quot;-:-.32, Load time: &quot;, LOAD_TIME:12:2, sec&quot;);</td>
</tr>
<tr>
<td>0171</td>
<td>write(AUX_OUT, &quot;-:-.Number of offset sources: &quot;);</td>
</tr>
<tr>
<td>0172</td>
<td>write(AUX_OUT, &quot;-:-.Number of offset sources: &quot;);</td>
</tr>
<tr>
<td>0173</td>
<td>LCT := CT + 1;</td>
</tr>
<tr>
<td>0174</td>
<td>for K := 1 to NUM_OF_SOURCES do</td>
</tr>
<tr>
<td>0175</td>
<td>begin</td>
</tr>
<tr>
<td>0176</td>
<td>write(AUX_OUT, &quot;-:-.Source &quot;, K:2, &quot;, Transmission rate: &quot;);</td>
</tr>
<tr>
<td>0177</td>
<td>write(AUX_OUT, SOURCE[K]:9:2, Bps&quot;);</td>
</tr>
<tr>
<td>0178</td>
<td>LCT := CT + 1; end</td>
</tr>
<tr>
<td>0179</td>
<td>end; (* for K := 1 to num of sources *)</td>
</tr>
<tr>
<td>0180</td>
<td>write(AUX_OUT, &quot;-:-.Trunk transmission interval: &quot;, TX_INTERVAL:7:4, sec&quot;);</td>
</tr>
<tr>
<td>0181</td>
<td>LCT := CT + 1;</td>
</tr>
<tr>
<td>0182</td>
<td>end; (* if not(OPEN) *)</td>
</tr>
<tr>
<td>0183</td>
<td>end; (* with ADAPTER[1..DEVICE[3]] *)</td>
</tr>
<tr>
<td>0184</td>
<td>end; (* for J := 1 to 4 *)</td>
</tr>
<tr>
<td>0185</td>
<td>end; (* with ADAPTER[1..DEVICE[3]] *)</td>
</tr>
<tr>
<td>0186</td>
<td>end; (* with ADAPTER[1..DEVICE[3]] *)</td>
</tr>
<tr>
<td>0187</td>
<td>write(AUX_OUT, &quot;:-:-.Cumulative probability of transmission matrix&quot;);</td>
</tr>
<tr>
<td>0188</td>
<td>write(AUX_OUT);</td>
</tr>
<tr>
<td>0189</td>
<td>write(AUX_OUT);</td>
</tr>
<tr>
<td>0190</td>
<td>write(AUX_OUT);</td>
</tr>
<tr>
<td>0191</td>
<td>LCT := CT + 1;</td>
</tr>
<tr>
<td>0192</td>
<td>write(AUX_OUT, &quot;:-:-.Probabilities of transmission matrix&quot;);</td>
</tr>
<tr>
<td>0193</td>
<td>for J := 1 to NUM_OF_ADAPTERS do</td>
</tr>
<tr>
<td>0194</td>
<td>for J := 1 to 4 do</td>
</tr>
<tr>
<td>0195</td>
<td>with ADAPTER[1..DEVICE[3]] do</td>
</tr>
<tr>
<td>0196</td>
<td>if not(OPEN) then</td>
</tr>
<tr>
<td>0197</td>
<td>write(AUX_OUT, 1:6, J:1);</td>
</tr>
</tbody>
</table>
(= write the probability matrix =)

for I := 1 to D_NUM do
begin
  for J := 1 to D_NUM do
    begin
      write(DESCRIPT_FILE, PR[I,J]);
      writeln(DESCRIPT_FILE);
    end;
  A end; (* procedure CREATE =)

begin (* main program *=

(* New run initializations *=
for COND1 := NORMAL to OTHER do
  CONDITION[COND1] := COND1;
PRINT_ALL := true;
U := 31622; (* U is seed for RNG *=
WAIT_TIME := 0;
COLLISION_TIME := 0;
TOTAL_ATTEMPTS := 0;
ACE := 0;
TIME_ACTIVE := 0.0;
BITS_TA := 0;

writeln(TERMINAL, "Maximum run time in secs?");
readl(TERMINAL, MAX_TIME);
write(TERMINAL, "Maximum successful transmissions?");
readl(TERMINAL, MAX_TX);
writeln;
write(TERMINAL, "Seed for random number generator? ");
readl(TERMINAL, U); ID := U;
writeln;
write(TERMINAL, "PRINT_ALL condition on? ");
readl(TERMINAL, RESP);
if (RESP[1] = 'y') or (RESP[1] = 'Y') then
  PRINT_ALL := true
else
  PRINT_ALL := false;
writeln;

(* Network cold start conditions *=
CHARACTERIZE_NETWORK;
page(OUTPUT);
CREATE_DESCRIPTOR_FILE;
PRINT_NET_DESCRIPTION;
FIND_NEXT(TITTRI);
CURRENT_TIME := TRITR.I.NEXT_TIME;

(* Network steady state operation *=
repeat (* until time exceeds max time *)
<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Time</td>
<td>36.6361 secs</td>
</tr>
<tr>
<td>Successful transmissions</td>
<td>500</td>
</tr>
<tr>
<td>Collisions</td>
<td>29</td>
</tr>
<tr>
<td>Wait attempts</td>
<td>529</td>
</tr>
<tr>
<td>Total attempts</td>
<td>529</td>
</tr>
<tr>
<td>Total time active</td>
<td>1.7105 secs</td>
</tr>
<tr>
<td>% Total time active</td>
<td>4.87%</td>
</tr>
<tr>
<td>Total bytes transmitted</td>
<td>987360</td>
</tr>
<tr>
<td>Seed for RNG</td>
<td>312577</td>
</tr>
<tr>
<td>END OF RUN</td>
<td></td>
</tr>
</tbody>
</table>
### END OF RUN: NETWORK STATISTICS ###

Current time: 36,4361 secs
Successful transmissions: 520
Collisions: 20
Total attempts: 520
Total time active: 1,212525 secs
Total Bytes transmitted: 987360
Seed for RNG: 332577

### DEVICE ACTIVITY SUMMARIES (SECONDS) ###

<table>
<thead>
<tr>
<th>ADP</th>
<th>DEV</th>
<th>TIME TRANSMITTING</th>
<th>TIME WAITING</th>
<th>TIME IN COLLISIONS</th>
<th>TIME RECEIVING</th>
<th>TIME ACTIVE</th>
<th>TRANSMISSION COUNT</th>
<th>RECEPTION COUNT</th>
<th>WAIT COUNT</th>
<th>COLLISION COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.9196</td>
<td>0.3897</td>
<td>0.0273</td>
<td>0</td>
<td>1.3557</td>
<td>252</td>
<td>0</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.2711</td>
<td>0.4398</td>
<td>0.0045</td>
<td>0</td>
<td>0.7455</td>
<td>35</td>
<td>0</td>
<td>37</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.4224</td>
<td>0.0037</td>
<td>0.0011</td>
<td>0.0012</td>
<td>0.4233</td>
<td>35</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0.0410</td>
<td>0.0110</td>
<td>0.0043</td>
<td>0.0938</td>
<td>0.7530</td>
<td>71</td>
<td>249</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.0410</td>
<td>0.0073</td>
<td>0</td>
<td>0.3460</td>
<td>0.4352</td>
<td>71</td>
<td>143</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.0035</td>
<td>0.0037</td>
<td>0.0035</td>
<td>0</td>
<td>0.0123</td>
<td>36</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4376</td>
<td>0.4376</td>
<td>0</td>
<td>71</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.2771</td>
<td>0.2771</td>
<td>0</td>
<td>35</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### DEVICE ACTIVITY SUMMARIES (PERCENT) ###

<table>
<thead>
<tr>
<th>ADP</th>
<th>DEV</th>
<th>% TIME ACTIVE</th>
<th>% TIME TRANSMITTING</th>
<th>% TIME RECEIVING</th>
<th>% TIME IN COLLISIONS</th>
<th>% TIME IN COLLISIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>76.15</td>
<td>5.77</td>
<td>22.78</td>
<td>25.71</td>
<td>0.26</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>41.82</td>
<td>1.95</td>
<td>0.77</td>
<td>25.71</td>
<td>0.19</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>25.62</td>
<td>25.26</td>
<td>1.26</td>
<td>0.21</td>
<td>0.37</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>43.83</td>
<td>2.40</td>
<td>0.11</td>
<td>0.64</td>
<td>0.71</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>20.77</td>
<td>2.40</td>
<td>0.11</td>
<td>0.43</td>
<td>0.71</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.72</td>
<td>0.10</td>
<td>0.01</td>
<td>0.21</td>
<td>0.57</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0.70</td>
<td>0.20</td>
<td>0.21</td>
<td>0</td>
<td>0.20</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>25.58</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>15.85</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Network Description
#### Adapter # 2

<table>
<thead>
<tr>
<th>Adapted to</th>
<th>Propagation distance to adapter 1</th>
<th>Propagation distance to adapter 3</th>
<th>Propagation distance to adapter 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 sec</td>
<td>0 sec</td>
<td>0 sec</td>
<td></td>
</tr>
<tr>
<td>Priority delay</td>
<td>0.0000000 sec</td>
<td>End delay: 0.0000000 sec</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device 1</th>
<th>status: CLOSED</th>
<th>Device number 227</th>
<th>Buffer size: 4000 Bytes</th>
<th>1/0 bus transfer rate: 960000.00 Bps</th>
<th>Load time: 0.00000002 sec</th>
<th>Number of offset sources: 1</th>
<th>Source # 1 transmission rate: 600.00 Bps</th>
<th>Trunk transmission interval: 1.0000 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device 2</td>
<td>status: CLOSED</td>
<td>Device number 327</td>
<td>Buffer size: 6400 Bytes</td>
<td>1/0 bus transfer rate: 960000.00 Bps</td>
<td>Load time: 0.00000002 sec</td>
<td>Number of offset sources: 1</td>
<td>Source # 1 transmission rate: 600.00 Bps</td>
<td>Trunk transmission interval: 1.0000 sec</td>
</tr>
<tr>
<td>Device 3</td>
<td>status: OPEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device 4</td>
<td>status: OPEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### NETWORK DESCRIPTION

<table>
<thead>
<tr>
<th>Adapter # 3</th>
<th>Propagation distance to adapter 1: C sec</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Propagation distance to adapter 2: C sec</td>
</tr>
<tr>
<td></td>
<td>Priority delay: 0 sec</td>
</tr>
<tr>
<td></td>
<td>End delay: 0.000020 sec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device 1 status: CLOSED</th>
<th>Device number 431</th>
<th>Buffer size: 512 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O bus transfer rate: 500000.00 Bps</td>
<td></td>
</tr>
<tr>
<td>Load time: 0.0000002 sec</td>
<td>Number of offset sources: 1</td>
<td></td>
</tr>
<tr>
<td>Source # 1 transmission rate: 1000.00 Bps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trunk transmission interval: 0.512 sec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device 2 status: CLOSED</th>
<th>Device number 532</th>
<th>Buffer size: 512 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O bus transfer rate: 500000.00 Bps</td>
<td></td>
</tr>
<tr>
<td>Load time: 0.0000002 sec</td>
<td>Number of offset sources: 1</td>
<td></td>
</tr>
<tr>
<td>Source # 1 transmission rate: 1000.00 Bps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trunk transmission interval: 0.512 sec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device 3 status: CLOSED</th>
<th>Device number 633</th>
<th>Buffer size: 624 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I/O bus transfer rate: 600000.00 Bps</td>
<td></td>
</tr>
<tr>
<td>Load time: 0.0500000 sec</td>
<td>Number of offset sources: 1</td>
<td></td>
</tr>
<tr>
<td>Source # 1 transmission rate: 625.00 Bps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trunk transmission interval: 0.9954 sec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device 4 status: OPEN</th>
</tr>
</thead>
</table>
**NETWORK DESCRIPTION**

**Adapter # 4**

<table>
<thead>
<tr>
<th>Propagation distance to adapter 1:</th>
<th>0 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagation distance to adapter 2:</td>
<td>0 sec</td>
</tr>
<tr>
<td>Priority delay:</td>
<td>0 sec</td>
</tr>
<tr>
<td>End delay:</td>
<td>0.000744 sec</td>
</tr>
</tbody>
</table>

**Device 1** status: CLOSED  
Device number 741  
Buffer size: 2000 Bytes  
I/O bus transfer rate: 1320000.00 Bps  
Load time: 0.0000001 sec  
Trunk transmission interval: 1000000.3499 sec

**Device 2** status: CLOSED  
Device number 842  
Buffer size: 2000 Bytes  
I/O bus transfer rate: 1200000.00 Bps  
Load time: 0.0000001 sec  
Trunk transmission interval: 1000000.3499 sec

**Device 3** status: CLOSED  
Device number 943  
Buffer size: 2000 Bytes  
I/O bus transfer rate: 1200000.00 Bps  
Load time: 0.0000001 sec  
Trunk transmission interval: 1000000.3499 sec

**Device 4** status: OPEN
<table>
<thead>
<tr>
<th>11</th>
<th>21</th>
<th>31</th>
<th>41</th>
<th>51</th>
<th>61</th>
<th>71</th>
<th>81</th>
<th>91</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>21</td>
<td>31</td>
<td>41</td>
<td>51</td>
<td>61</td>
<td>71</td>
<td>81</td>
<td>91</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>