IST's DIS Pdu Flooder: Final Report

1-1-1996

Scott H. Smith
Jada D. Kearns

Find similar works at: http://stars.library.ucf.edu/istlibrary

University of Central Florida Libraries http://library.ucf.edu

Recommended Citation


This Research Report is brought to you for free and open access by the Digital Collections at STARS. It has been accepted for inclusion in Institute for Simulation and Training by an authorised administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
IST’s DIS PDU Flooder
Final Report
IST's DIS PDU Flooder

Final Report

IST-TR-96-05
February 13, 1996

Prepared For:
STRICOM
N61339-94-C-0024
CDRL A00P

Prepared By:
Scott H. Smith

Reviewed By:
Jada D. Kearns
# Table of Contents

1.0 Purpose .............................. 3

2.0 Background .............................. 3

3.0 Design Approach .............................. 3
   3.1 Background .............................. 3
   3.2 Requirements .............................. 4
   3.3 Design Strategy .............................. 4

4.0 Selected Approach .............................. 5
   4.1 Hardware .............................. 5
   4.2 Software .............................. 6
   4.3 Implementation .............................. 7

5.0 Capabilities .............................. 8

6.0 Performance metrics .............................. 9

7.0 Operation .............................. 9
   7.1 Installation guide .............................. 9
   7.2 Operation guide .............................. 10
   7.3 Examples .............................. 13
      7.3.1 Example #1- One-Shot Ramp Profile .............................. 13
      7.3.2 Example #2- Square-Wave Profile .............................. 16
   7.4 Programmer’s Notes .............................. 20
      7.4.1 DIS Version Changes .............................. 20
      7.4.2 Logger Format Changes .............................. 20

8.0 Conclusions and Comments .............................. 20

9.0 References .............................. 20
1.0 Purpose

This report is a deliverable item (CDRL A00P) required in completion of subtask 3.2.5.1, “DATA GENERATION TOOLS”, on STRICOM contract N61339-94-C-0024 entitled, “TRIDIS: A Testbed for Research in Distributed Interactive Simulation.”

2.0 Background

The “FLOODER” is IST’s name for its Distributed Interactive Simulation (DIS) application which is able to place large numbers of syntactically and semantically correct DIS PDUs on a network to simulate the traffic that would be generated by multiple DIS applications. Other terms have been used for similar tools for this sort of “synthetic packet generation.” Sometimes they are called “traffic generators” or “packet blasters.”

The conditions created by a flooder may be used in tests which investigate the ability of other DIS applications to function under varying conditions. The kinds of DIS PDUs produced by IST’s FLOODER, the kinds of entities that may be represented, the kinds of behaviors reflected in its output traffic, the numbers of entities being simulated, and other parameters such as the source IP addresses are all variable. Some of these parameters are also dynamically variable.

3.0 Design Approach

This section discusses various approaches to synthetic traffic generation and describes the requirements which IST generated for the tool.

3.1 Background

Of the many different hardware/software tools that have been created to facilitate DIS development and testing, “traffic generators” or the so-called “packet blasters” have probably received the least attention. Testing of DIS applications is a complex and multivariate process which has been hampered by the lack of a comprehensive and specific standard for specifying the requirements for compliance, interoperability, and compliance and metrics to describe “capability”.

One significant measure of the “capability” of a DIS application is its ability to continue to operate properly while processing incoming network traffic. Systems which can generate the behavior of one or more simulation entities are usually limited in their ability to do so by the requirement that they “track” and interact with entities generated by other applications. Verifying their ability to continue to function in the presence of greater or lesser inputs requires repeatable, measurable and verifiable sources of externally generated traffic. To date, these sources have been derived in three ways:

- Recorded network traffic is reapplied to a test network via a “playback” mechanism
- A Computer Generated Force (CGF) application is used to generate traffic
- A specialized application (sometimes an extremely simplified CGF) is used to dynamically generate large amounts of (relatively) “unsophisticated” traffic
These methods all have their advantages and disadvantages. Playing back a previously recorded exercise is easy. Numerous data loggers exist with the capability to replay the files they generate. Some allow filtering by different criteria, either when recording or during playback. The recorded packets, however, are not easily modified. Some systems (such as IST’s) retain the lower level protocol information but do not allow it to be changed. IP addresses remain the same; entity IDs cannot be modified. Some systems do not retain the lower level information and must recreate it during playback. This may cause all entities to appear to have been generated by a single application.

Use of a CGF system to generate traffic is likely to result in reasonable and correct behavior and may provide the flexibility to create a wide range of entity types and behaviors. This capability, however, comes at the expense of significant computational burden and such a system may be limited to small numbers of entities, perhaps in the range of 10 to 60.

Specialized packet blasters which are actually oversimplified CGF systems may achieve higher output rates at the expense of behavioral correctness. They may create more ground entities, for example, but not be able to spend the computational effort necessary to place them correctly on the terrain surface.

3.2 Requirements

IST compiled the following list of requirements based on experience in developing DIS applications (crewed simulators and CGF), test procedures, and other test tools during the period between 1989 and 1995.

To be useful, a floodér should be able to:

- Generate syntactically and semantically correct DIS traffic with the option to introduce “flawed traffic” at will. This requirement includes the ability to generate the protocol layers below the application layer as well.
- Generate traffic at rates up to and above those which applications are likely to tolerate.
- Generate traffic at variable rates (both statically and dynamically variable.) The data rates for the FLOODER need to vary from a high percentage to a low percentage of the available Ethernet bandwidth. High rates are useful for stressing simulators and networks to the limit of their performance. Although low rates are easy to generate using the other methods listed in section 3.1, they are useful features to include in a traffic generator because they provide a repeatable source of test data
- Generate traffic of variable character, i.e. different mixes of entity types, velocities, maneuvering patterns, etc. in different environments (ground, air, surface, etc.)
- Generate traffic which is valid at any location on the simulated “Earth”.

3.3 Design Strategy

IST considered ways in which some of the three earlier mentioned approaches might be modified to achieve the required goals.
A fast Computer Generated Forces (CGF) simulator with a set of script files and some automated controls could provide some flooding capabilities. Although it could be simplified to ignore network input, it would, unfortunately, be limited by the computational burden of its (typically) “higher fidelity.”

A second option that was considered was a “playback” program which plays logged files back faster than real-time. This approach, although simple, is not useful as it would generate “unrealistic” behavior, i.e. velocities, locations, timestamps, and transmission times would not correlate.

A third approach that was considered was using a “PDU Editor” to merge logged DIS files to generate sufficient traffic to load systems being tested. This would require recording a number of exercises and interleaving the records in one master file which would then be played back in “real” or “correct” time. Neither IST’s PDU Editor (described in TRIDIS monthly reports) nor any other tools known to IST include the necessary sort of file merge capability.

IST decided to use a combination of a logged file editor and a specialized high capacity playback mechanism to build a program that creates traffic patterns that can be spatially and temporally altered to test a wide range of conditions. This approach offers considerable flexibility. To a large extent, IST’s FLOODER meets the stated objectives.

4.0 Selected Approach

IST attempted to select a combination of hardware, software, and implementation techniques which would accommodate all of the requirements listed above.

4.1 Hardware

Hardware is required which can place data on an interface at rates approaching Ethernet’s theoretical limit of 10 MB/s (million bits per second). IST chose to use a computer that consists of a VME chassis with two MVME197 processor boards installed. These boards each contain a Motorola 88110 Reduced Instruction Set Computer (RISC) Central Processing Unit (CPU) with 64MB of main memory which may be shared by any other CPU in the chassis. One processor is called the “host” board and the other is the “target” board.

The operator interface is through an X-Windows terminal. The terminal connects to the system through the LAN port on the host board. The FLOODER’s DIS output is through the LAN port on the target board. The MVME197 processor board uses the Intel 82596 Ethernet Controller chip to provide the Ethernet interface. This Ethernet interface is capable of transferring data at the full 10Mb/s rate.

4.2 Software

The host board uses a V/88 release R40V4.2 UNIX operating system from Motorola. The target board uses the Motorola R4 VMExec 3.0 operating system. The applications are written in ANSI C.

The PDU FLOODER software on the target board communicates directly with the Intel 82596
Ethernet Controller chip, bypassing the standard VMEexec driver. This approach allows the FLOODER to generate network traffic at a sustained maximum rate of approximately 5000 DIS Entity State PDUs per second.

The CPU interfaces with the 82596 through the CPU's memory. The CPU writes Command Blocks (CBs) into memory and then the Command Unit (CU) of the 82596 uses Direct Memory Access (DMA) to fetch and execute the commands. The CU can execute a list of commands without CPU intervention. To facilitate this type of operation a CB has a link field which points to the next CB and a flag to mark the end of a list. There are also flags in a CB that tell the CU to go into a suspended state or interrupt the CPU after executing a command. The Transmit commands have a link to a data buffer area. Figure 1.1 shows this relationship.

![Figure 1.1 Command Buffer Linkage for Command Unit Execution](image)

During normal operation the FLOODER uses one of the timer channels on Peripheral Channel Controller (PPC) Chip #2 on the MVME197 processor board to provide a variable delay between the transmission of PDUs. The software starts the timer after the 82596 interrupts to signal completion of the previous SEND command. Using the different delay times between each PDU, the FLOODER can vary the network utilization between 0.0% and to 71.75% or 3,869 PDUs per second.

As the interpacket delay decreases, the proportion of time that is spent in handling interrupt overhead increases. The limit of increasing network utilization (71.75%) occurs with approximately a 10 microsecond interpacket delay. To achieve higher rates of output the FLOODER chains multiple transmit commands together, pausing between groups of PDUs. This decreases the number of interrupts, resulting in a smaller average inter-PDU delay. Figure 1.2 shows the timeline for a transmission cycle using the PCC timer and "send complete" interrupts.
The chains of transmit commands effectively lower the interrupt overhead because more packets are sent per interrupt cycle. Using chains of four or more packets, utilization up to 92.73% can be achieved.

![Figure 1.2 Transmission Cycle Timing Diagram](image)

### 4.3 Implementation

The hardware and software arrangement described above allows playback of recorded PDUs which are placed in special data structures in the target board's 64 MB of main memory. A process was defined which would allow construction and loading of these data structures containing syntactically and semantically correct DIS PDUs which would represent different configurations of entities, behaviors, locations, and dynamic changes in these configurations.

The process includes the following steps:

- Record DIS network traffic involving the types of entities and behaviors (maneuvering, shooting, etc.) to be present in the output data. This implementation uses IST's MVME197-based logger and the logging format which places an entire Ethernet frame in a data structure which also includes the frame receipt time and frame length in bytes. This logger and its data structure are described in [95-02].
- Specify the physical location in the world for possible relocation of the exercise. The output may be assigned to a geographic locality different from the that in which it was originally recorded.
- Select entities from the logged file to use as examples
- Select the beginning and ending numbers of entities (allows dynamics change in traffic rates and types)
- Specify the period over which the entities will be introduced into the output
- State the time at which output is to begin
• Command the flooder to load the data into the target board’s memory in patterns which reflect the above specified arguments
• Wait until the scheduled output begins

5.0 Capabilities

The FLOODER allows considerable flexibility in specifying output traffic patterns. By specifying the UTM location of the data as logged and the UTM location of a second location on the Earth’s surface, activity can be translated to a new piece of terrain. This is recommended only for air and surface entities as it does not provide proper placement of ground vehicles on the terrain skin.

Translation of locations is accomplished as follows:

• The origin of the database which was used during the logging process is specified in one configuration file.
• During profile generation the DIS Geocentric coordinates are translated into UTM.
• Then the UTM coordinates are translated back into Geocentric, using the UTM origin specific in a second configuration file. This is the origin of the database in which the profile output is to be generated.

A large number of entities can be created by “time delayed replication.” An entity selected from the logged data can be replicated, using most of the data in its entity state PDUs but with new IDs and modified timestamps. The appearance of such a group usually resembles that of a convoy of vehicles. Because each follows in the track of the model entity they will all conform to the terrain surface as well as the model does. This may be used in all domains but is particularly useful in the ground domain.

The traffic rate may be varied dynamically, resulting in what might be described as a pulse train or sawtooth pattern. In the former, output for the desired number of entities is “switched” on and off simultaneously. In the latter, a time period is specified during which the replicated entities are created one-by-one. In both cases these patterns may be produced as a single-shot or repeated as many times as desired.

Many of the fields in the lower levels of the protocol stack can be modified including the site, host, and exercise in the DIS PDU header, UDP port in the UDP header, and Ethernet address in the frame header. Any or all of these parameters can be modified so that the desired PDU data is sent in an environment that is also flooded with data that should be rejected. This allows testing of the robustness of the protocol stack handling routines.
6.0 Performance metrics
Upon completion of the prototype, some simplified measurements were taken using a Hewlett-Packard LAN Analyzer. On an isolated Ethernet segment, with no other sources of traffic to generate collisions, it was found that it was possible to output PDUs at a maximum rate equivalent to 92.73% of Ethernet bandwidth (i.e. 9.273 MB/s). This was equivalent to 5,808 DIS Entity State PDUs per second. It involved loading the target board memory with 288,199 PDUs, which took 3 minutes. The FLOODER was able to maintain this rate for 52 seconds without repeating.

7.0 Operation
This section describes the installation and operation of the FLOODER. Some examples are provided to show how the tool may be used to generate different patterns of simulated DIS traffic.

7.1 Installation guide
Follow the steps listed below to install the system from the TAR tape submitted with this report.

1. Copy the file flooder.tar to the desired base directory. During the installation process subdirectories will be created under the base directory. The sub-directories and their contents are listed below:
   . /FloodInc - Contains a header file used by both the host and target programs.
   . /mvme197/c2.nm - Symbols from the target kernel program.
   . /mvme197/vmexgen - Sub-directories and files used by vmexgen to generate a kernel for the target board.
   . /mvme197/target - Sub-directories and files which describe the target board configuration. This information is used by vmexgen to generate a kernel for the target board.
   . /mvme197/src/ctask/flooder_target - The source code for the target board program.
   . /mvme197/src/host/cmd/flooder_host - The source code for the host board program.
   The directory structure under the mvme197 directory is used by the Motorola supplied utility mkmk, which creates a Makefile for compiling and linking the programs. Documentation of this directory structure is supplied by Motorola in the VMEexec User's Guide on starting on page 4-1.

2. Extract the FLOODER files with the command tar -xf flooder.tar
3. Change directories to mvme197/src/host/cmd/flooder_host.
4. You may want to edit some of the information in flooder.cfg. This ASCII text file contains the simulator ID information (Site, Host, etc.) that will be used to form the FLOODER output data stream.
5. Modify the script file "flood" to match the selected paths. The FLOODER program may be
started by executing the script file “flood”.

7.2 Operation guide

The process of using the FLOODER involves three steps; defining a profile to transmit during the “flooding” process, transferring the profile to the target board, and starting the transmission. Figure 1.3 shows the procedural flow of this operation.

When a profile is defined the FLOODER builds a description of that profile in the host board. The process of defining a profile starts with the selection of an entity which will be duplicated in the FLOODER output. The first step is to select a logger file which contains PDUs from the desired entities. If the entity is an aircraft then a configuration file must also be used. This allows the FLOODER to translate the entities spatially (please refer back to Section 5.0). This converts a single airplane into a sky full of airplanes all flying the same pattern. Land based entities do not require a configuration file since they are duplicated temporally. This converts a single tank into a line of tanks moving single file. After the logger file has been selected the FLOODER will display a list of the entities in the file and the user may select an entity from this list.

The next step in defining the profile is to replicate the entity. If the entity is an aircraft, the menu option “Enter a linear profile” is used to duplicate it. The FLOODER will prompt the user to enter the starting and ending number of entities, the time to take going from start to end number of entities, the lower-near-left corner and the upper-far-right corner of the bounding cube, the x y z coordinate step sizes, and whether or not the entities should be removed at the end time. This process of selecting entities and defining profiles can be repeated as many times as needed to build up the desired combination of entities.

All coordinates are entered in Universal Transverse Mercator (UTM) coordinates. The entity starting locations are spread over the bounding cube. If the entity is a land based item, the menu option “Enter a time replication profile” is used to duplicate it.

The last part of the process is to send the profile to the target board and give the target board the profile data and the start command. The method used to perform these functions depends on the type of time stamps that the user wants to use for the output data. If relative time stamping is desired, loading the data into the target board and issuing the start command are independent operations. This is not the case with absolute time stamps because the host board must calculate the values for the time stamps before loading the data into the target board. For these time stamps to be accurate the start command must be given at the proper time. The FLOODER facilitates this by providing separate menu options for each choice.
Figure 1.3 FLOODER Operation
7.3 Examples

The following examples show how several different output profiles can be specified. In each, FLOODER output is shown in bold typeface and User input is underlined.

VME Exec’s makefile build utility called “mkmk” requires that source code be located in subdirectory structures with the pathname:

```
    _I_I_/mvm3197/src/host/cmd/.*
```

where *.* includes the source and include files. Mkmk will look at all of the source files in the directory and build a makefile.

The directory which was archived and transferred to compressed TAR tape to accompany this deliverable is:

```
tiger:/home/tridis/slewis/mvme197/src/host/cmd/flooder_host
```

To start the FLOODER, go to this directory and enter:

```
flood
```

This runs a Motorola script file which resets the target board, loads the target board with the executable called flooder_host, and starts that executable.

To open a window to the target board enter

```
cv c21
```

To disconnect from the target board, enter:

```
-
```

Due to timing dependencies between the two boards this may require more than one attempt.

Once the FLOODER starts, its main menu will appear. The next sections describe examples of input sequences required to generate, load, and run different kinds of profiles.

7.3.1 Example #1- One-Shot Ramp Profile.

A “Ramp Profile” increases the output rates over a specified period of time by adding new entities. Each entity executes the same behavior. In this case the entities are aircraft. They are added at grid locations within a 3D array The beginning and ending number of entities and the ramping period are variable as are the location of the grid and its internal spacing.

To generate a ramp profile follow the following procedure.

1. Verify off-line that the terrain database information in “flooder.cfg” matches the desired output terrain database.
2. Open the appropriate windows and start the flooder using the "flood" script.

**FLOOD**

The main menu will appear. As selections are made this menu will be redrawn.

*** Input data selection ***

(0) Select a logger file containing airborne entities
(1) Select an airborne entity
(2) Duplicate airborne entities
(3) Select a logger file containing land based entities
(4) Select a land based entity
(5) Duplicate land based entities
(6) Mark the start of a loop
(7) Mark the end of a loop
(8) Load profile data into the target board
(9) Start transmission
(10) Timed start
(11) Get available memory from target board
(12) Reset the target board
(13) Quit

?  

3. Develop specifications for the pattern to be loaded.

3.1 First, select a logged file data source.

?Q

Enter the name of the logger file

fly1a.bin

Enter the name of the configuration file

fly1a.cfg

Menu re-displays here
3.2 Select an entity whose PDUs we will use.
Selection 1 will cause a list of entity Ids to be printed. Select one of them. In this example one ID is generated because the .bin file contained PDUs from only one entity.

?1
1) Site = 17, Application = 99, Entity = 100
Pick one?1
81 PDUs loaded from the file

Menu re-displays here

3.3 Request a duplication of air entities

?2

Specify the beginning and ending numbers of entities and the time period over which to change the count. The following input will create 10 entities over the course of 15 minutes

Enter the starting number of entities: 1
Enter the ending number of entities: 10
Enter the time of the profile in minutes: 15

Enter coordinates (in UTM x,y,z) of a bounding box in which to replicate entities. The z-coordinate is absolute altitude.

Enter the coordinates of the bounding volume min point: 300.300,500
Enter the coordinates of the bounding volume max point: 600.600.500
Enter the coordinate step sizes: 100,50,0
Deactivate entities at the end of the profile (y,n)? y

Menu re-displays here
4. Start the output.
Selection 8 will load and wait until selection 9 starts output. This will generate relative timestamps. The “Timed Start” option (10, shown in this example) is required for absolute time stamps. When it is used the FLOODER will determine how long it will require to load the output pattern. It will notify the operator of the time required, compute absolute timestamps as it loads, and then start transmitting at the correct time (provided the host’s clock has been properly synchronized.

?10

The current time is: Wed Nov 29 09:24:42 1995
It will take approximately 1 minutes
Start transmission in how many minutes? 2
Transmission will start at: Wed Nov 29 09:27:12 1995
Adjusting time stamps
Loading the profile
10000 PDUs generated
20000 PDUs generated
24669 PDUs were loaded
Waiting to start

Program will pause here until the start time

Transmission started

Menu re-displays here

5. The FLOODER will continue outputting data until 15 minutes have expired. To stop the output early, exit the program as follows:

?13

7.3.2 Example #2- Square-Wave Profile.

A “Square wave Profile” repeatedly generates a period of fairly constant output rate followed by a period of zero output. The constancy of output depends on the logged behavior which is used. To generate a square-wave profile follow the following procedure.
1. Verify off-line that the terrain database information in “floodor.cfg” matches the desired output terrain database.

2. Open the appropriate windows and start the flooder using the “flood” script.

FLOOD
A menu will appear. As selections are made this menu will be redrawn.

*** Input data selection ***
(0) Select a logger file containing airborne entities
(1) Select an airborne entity
(2) Duplicate airborne entities
(3) Select a logger file containing land based entities
(4) Select a land based entity
(5) Duplicate land based entities
(6) Mark the start of a loop
(7) Mark the end of a loop
(8) Load profile data into the target board
(9) Start transmission
(10) Timed start
(11) Get available memory from target board
(12) Reset the target board
(13) Quit
?

3. Develop specifications for the patterns to be loaded.

3.1 First, select a logged file data source to be used to generate traffic during the low phase of the Square-Wave.

?0

Enter the name of the logger file

fly la.bin

Enter the name of the configuration file

fly la.cfg

Menu re-displays here

3.2 Select an entity whose PDUs we will use during the low phase.

Selection 1 will cause a list of entity IDs to be printed. Select one of them. In this example one ID is generated because the .bin file contained PDUs from only one entity.

?1
1) Site = 17, Application = 99, Entity = 100
Pick one?1
81 PDUs loaded from the file

Menu re-displays here

4. Define the activity level for the low segment of the square wave.

Enter the starting number of entities: 1
Enter the ending number of entities: 1
Enter the time of the profile in minutes: 1
Enter the coordinates of the bounding volume min point:
0,0,500
Enter the coordinates of the bounding volume max point:
100,100,500
Enter the coordinate step sizes:
10,10,0
Deactivate entities at the end of the profile (y,n)?
y

Menu re-displays here

Select another logged file data source to be used to generate traffic during the high phase of the Square-Wave.

?Q

Enter the file name

fly1a.bin

Enter the data configuration file name

fly1a.cfg

Menu re-displays here

Select another entity

?1
1) Site = 17, Application = 99, Entity = 100
Pick one?1
81 PDUs loaded from the file

Menu re-displays here
5. Define the activity level for the high segment of the square wave

?2
Enter the starting number of entities: 20
Enter the ending number of entities: 20
Enter the time of the profile in minutes: 5
Enter the coordinates of the bounding volume min point: 100,100,600
Enter the coordinates of the bounding volume max point: 600,600,600
Enter the coordinate step sizes: 50,50,0
Deactivate entities at the end of the profile (y,n)? y

Menu re-displays here

6. Make the square wave repeat by marking the start of a loop

?6

Menu re-displays here

7. Load the Profile data into the target board

?8
3301 PDUs were loaded

Menu re-displays here

8. Indicate when the square wave will end by marking the end of a loop

?7

Menu re-displays here

9. Begin to send the packets. The square wave repeats until the enter key is pressed

?9
Transmitting loop #0
Press enter to continue past the loop
Broken out of loop #0
Done transmitting the profile
10. Exit the FLOODER

7.4 Programmer’s Notes

7.4.1 DIS Version Changes

The LOGGER was developed using DIS Version 2.0.3. Adapting to newer versions will require modification of the include file named loc_dis.h in the logger subdirectory in the flooder_host directory. For alignment purposes the floating point fields are copied to aligned local variables as hex bytes so, as long as the size of these fields doesn’t change, they should still work.

7.4.2 Logger Format Changes

The FLOODER currently interprets the IST LOGGER file format. Changes in the format of the logger files (for example, to use DLIF-95 Interchange format) will require modification of the routine ReadPacketFromFile in the module binary.c. This function reads the PDU data from the logger file and supplies it to the FLOODER in a PACKET_NODE structure.

8.0 Conclusions and Comments

The prototype of the FLOODER demonstrates the flexibility of the chosen approach. In combination with a LOGGER and original source of network traffic, such as a CGF system, a wide variety of traffic types and intensities can be generated.

The prototype FLOODER has only a simple menu driven interface which could benefit from a better design. It would probably be useful to provide the operator with immediate feedback about the rates of output that will be generated before output begins, the number of entities that will be represented at any time during a run, the number of PDUs of each type being generated per unit time, and total transmitted PDUs at the end of a run.

The system was designed to provide maximum usage of standard (10 Mb/s) Ethernet. The DMA controller and Ethernet interface hardware have no problem with this. Current development in DIS, however, such as CCTT, will make use of higher capacity networks such as FDDI or Fast Ethernet (100Mb/s). It is recommended that an attempt be made to scale the FLOODER to stress the limits of some of these other protocols.

9.0 References
