DATA SYSTEMS DYNAMIC SIMULATOR

Christopher Rouff
Melana Clark
Bill Davenport
NASA Goddard Space Flight Center
Code 522.1
Greenbelt, MD 20771

Philip Message
Stanford Telecom
7501 Forbes Boulevard
Seabrook, MD 20706

ABSTRACT

The Data System Dynamic Simulator (DSDS) is a discrete event simulation tool. It was developed for NASA for the specific purpose of evaluating candidate architectures for data systems of the Space Station era. DSDS provides three methods for meeting this requirement. First, the user has access to a library of standard pre-programmed elements. These elements represent tailorable components of NASA data systems and can be connected in any logical manner. Secondly, DSDS supports the development of additional elements. This allows the more sophisticated DSDS user the option of extending the standard element set. Thirdly, DSDS supports the use of data streams simulation. Data streams is the name given to a technique that ignores packet boundaries, but is sensitive to rate changes. Because rate changes are rare compared to packet arrivals in a typical NASA data system, data stream simulations require a fraction of the CPU run time. Additionally, the data stream technique is considerably more accurate than another commonly-used optimization technique.

INTRODUCTION

Development of the Data Systems Dynamic Simulator (DSDS) started in the late 1970's at Marshall Space Flight Center. Under contract to NASA, the General Electric Company was tasked to build a discrete event simulation tool especially suited for modeling NASA end-to-end data systems of the Space Shuttle and Space Station eras. Since then, DSDS has been in continual use. In 1985 the management and control of DSDS was transferred to Goddard Space Flight Center.

In 1986, Stanford Telecom was tasked by the Customer Data Operations Service (CDOS) project at GSFC to develop an end-to-end model of CDOS. Although CDOS, now called EDOS, is a ground-based system, it was necessary to accurately model the sequence of data arriving at EDOS (from space). The objective of the EDOS model was to study the traffic and mission profile for selected twenty four hour periods. As in any modeling task, the proper tool selection is an essential step. Therefore, EDOS personnel searched for a suitable tool to meet the EDOS model requirements.

Because of EDOS' high data rates, complexity and excessive simulated time requirement, the search proved to be difficult. None of the tools or languages surveyed supported a suitable optimization technique. It was decided that the data stream modeling optimization technique offered the best potential solution. While the data stream theory was well known, the method had never been implemented on a scale suitable for the EDOS model. A survey of existing tools was done, and DSDS was selected for the following reasons: it was NASA-owned and maintained, it was extensible and it supported orbital modeling.

Since 1986, data streams have been used to model other NASA end-to-end data systems, including Space Station Freedom (SSF) and the Earth Observing System (EOS).

DSDS OVERVIEW

DSDS contains predefined, configurable elements that can be used to represent components of a data system. Examples of elements that are available for simulating systems are CPU's, data generators (scientific instruments or experiments), orbit calculators and schedulers. A model is constructed by linking elements together to represent a network over which simulated data will flow. Sizes of queues and time to send data between elements can be simulated. From the results of the simulation the sizes of buffers, the speed of processors and the speeds of data links that will be needed for the system to have a particular performance can be determined. Though DSDS itself does not process cost information, once the types of components are determined, the modeler can then determine the price of the proposed system.

The model assumes that data is sent between elements in a packet. The packets can be simulated in one of two modes: packet by packet or as a data stream. Using packet simulation, packets are sent through the system one at a time. In the data streams model the system is modeled by using the rate at which data is being sent through the system. Both simulation techniques are discussed further below.

As shown in Figure 1, there are four main parts to DSDS: SETUP, SIMULATE, GRAPHICS and USRLOAD. SETUP is where the user defines the elements to be used and how they are to be interconnected. Parameters for each element allow the elements' behavior to be customized for different applications. SIMULATE is where a model of a system is simulated. SIMULATE displays graphics on the progress of the simulation, displays other statistics, has debugging capabilities to trace through a simulation, and produces reports that provide statistics on queue lengths and time for data to flow through the system. GRAPHICS prints plots and reports generated by SIMULATE. USRLOAD lets users extend DSDS by allowing them to write their own elements.

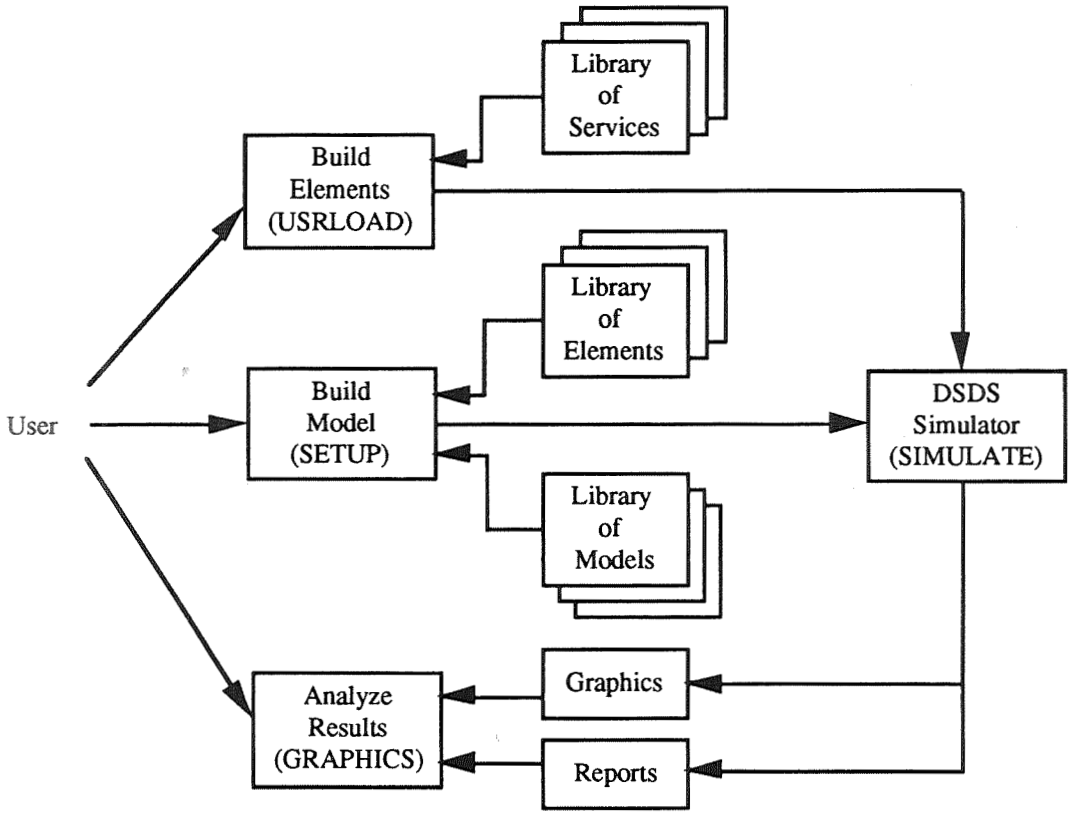

Figure 1: Components and flow of data in DSDS.

DSDS is a block-oriented simulation tool. Standard DSDS consists of a library of about fifty pre-programmed blocks (i.e., elements). The use of standard DSDS blocks require no programming. Each

DSDS block has a unique number identifier plus a name. Block names identify a type of operation such as CPU, DUPLIC, AND, OR, BRANCH, etc. Alias names are optional.

DSDS users are prompted to choose default values or enter appropriate parameters such as packet size, priority, data type, destination block, processing rate, etc. DSDS elements are connected by lines or links that are specified by the model builder. During model execution, information is passed from block to block via these links.

In DSDS, the movement of information is called an event. A DSDS event is represented by a twenty five word FORTRAN array. Array words contain specific attributes about the event (e.g., name, priority, destination, number of bits, data type). A FORTRAN subroutine is associated with each block. The arrival of an event invokes the subroutine code which in turn performs the simulated operation.

Each event arrival changes the state of the system being modeled. System states are recorded in a time-ordered history, called a timeline file, that is created and maintained by DSDS as the simulation proceeds. Data from a timeline can be converted interactively or off line into customized graphs and/or reports. Events occur in simulated time. The next event may happen in a micro-second or a day. Therefore there is no relationship between wall clock time and simulated time. However, the number of events processed has a direct relationship to CPU run time. This issue will be discussed later.

The user of standard DSDS is constrained by the functionality of a finite set of blocks (or elements). Standard DSDS constraints are indicative of similar tools with pre-programmed elements. The constraints are twofold. The first is the nuisance factor. While it may be possible to get the job done with standard elements, the model configuration is not economical. In these cases, simplification could be accomplished with a customized element. In the second, more serious case, functionality does not exist. Then it becomes essential that a new element be constructed.

USRLOAD supports the development of user-defined elements. The modeler is provided with a template and prompted to describe characteristics, such as name, number of user parameters, number of queues, etc. A documented set of DSDS utility functions is available to the user. The use of these utilities guarantees that the user-defined element will function consistently with the elements in the standard set. However, the user is required to write a FORTRAN subroutine with the desired functionality.

The integration of user-defined elements into the standard set is decided by the DSDS manager. Examples of user elements which have been ported to the standard set include two TP-4 protocol elements. These elements were developed by a DSDS user (LinCom Corporation) for Johnson Space Flight Center (JSC). The user extension feature gives rise to the term "Dynamic" in the name Data Systems Dynamic Simulator.

OPTIMIZATION TECHNIQUES

Large complex data systems create an extremely large volume of data. A limiting factor, when modeling these data systems, is the amount of simulation (CPU) run time required to complete the simulation. The CPU run time increases when the number of events in the simulation increases. When generated each packet creates an event. As the packets flow through the model (being queued, processed, sorted, etc.) more events are created, thus increasing the CPU run time of the simulation. For large complex data systems with extremely large volumes of data the number of events generated can become overwhelming. Since the purpose of doing a simulation is to try different configurations, determine potential bottlenecks, and test out the effects of different components with different costs and performance, it is important to be able to run a simulation many times in a timely fashion. This allows different versions of the model to be compared to determine which components would work best given system constraints, such as price and performance.

To bring simulation run times down to a reasonable value, modelers often use optimization techniques. The next two sections describe two different optimization techniques and their use with DSDS.

Artificially-Inflated Packets

When modeling a NASA system, the data that passes through it is divided into packets. Packets (or messages) may be characterized by size. In standard DSDS, packet size is a parameter that is supplied by

the user. The size of these data packets can vary, but the typical NASA packet size is in the range of 10,000 bits. Very large numbers of packets are associated with the simulation of current and emerging NASA data systems. Every packet transfer results in a state change within the model. Each state change requires CPU run time. For instance, to simulate a 24 hour "true" EDOS model it would take more than a day of CPU time on a relatively fast computer (VAX 8600).

As complexity, data rates and need for longer model run times increases, so does the need for an optimization technique. Artificially raising the packet size is one way of optimizing the simulation. By increasing the packet size, say from 10,000 to 100,000 bits, CPU run time is reduced by a factor of 10 due to the decrease in events flowing through the system. Unfortunately, increasing the size of packets results in errors. The magnitude of the error can be predicted by comparing the results to a "truth" model. A truth model is one constructed with the "real" packet size. It is compared to a test model which is constructed with the elevated packet size. However, it is impractical to develop a truth model for some large systems, such as EDOS. The model, when fully configured, would consist of more than 100 payloads (experiments). Before reaching users, payload data must pass through 10 or more processing points. Thus it is not easy to determine the effects on the fidelity of the modeling results when the packets are inflated artificially.

Data Streams

The data streams technique is another way of optimizing the modeling process. As stated above, the objective is to reduce the number of events in a given simulation run. The data streams method models rate changes rather than individual packets. Consider the example of an experiment which transmits a 10,000 bit packet at the constant rate of 100 packets per second. Consider also that the experiment transmits 10 minutes each orbit. In a "true" model this translates into 600,000 packets (events) per 10 minute duty cycle. A data stream represents this duty cycle as two events; namely one start and one end. (The data stream would be characterized as a 10 minute stream with a transmission rate of 600,000 bits per second.)

The key to understanding the data stream methodology is that it takes advantage of the linear flow of data between state changes. The speed at which the simulation runs for the data streams method is not dependent on the volume of packets, but instead is dependent on the number of times the data rate changes. The data streams method requires less computation and thus reduces the time to simulate the passing of data through the system. Data streams take advantage of the fact that data systems behave linearly between state changes. Therefore, data streams can model the effects of the changes in the data rates of a system rather than modeling each individual packet. This optimizing method will also reduces the CPU run time of a simulation due to the decrease in events.

One difference between packet and data stream modeling is the way a processor's bandwidth is allocated. A packet, no matter its size, occupies the entire bandwidth of its processor for some finite period of time. Data streams, on the other hand, occupy only that portion of the bandwidth which is equal to or less than its transmission rate. Furthermore, data streams share the bandwidth proportionally on a first in, first out basis with other competing streams.

Comparison of Data Streams to Packet Modeling

Data stream simulation is tantamount to modeling with infinitesimally small (approximately 1 bit) packets. In terms of magnitude, 1 bit is closer to 10,000 than 20,000 bits and more. Based upon this empirical point alone, an assumption could be made that errors induced by data stream optimization would be less.

An experiment was performed to assess the error produced by artificially increasing the packet size of a packet model of a system and the error introduced by data streams modeling. Models using these optimization techniques were compared to a "Truth Model" - a model which is run using the actual packet size. Figure 2 shows the model that was used for the experiment. Figure 3 shows the run times of the Truth Model where the packet sizes were 15 Kilobits (reflecting the actual implementation), a model where the packet size was artificially expanded to 1 Megabit packets, and the data streams methodology. As expected, the Truth Model ran the longest, 3,279 CPU seconds. The expanded packet model ran for 47 seconds and the data streams model for 62 seconds.

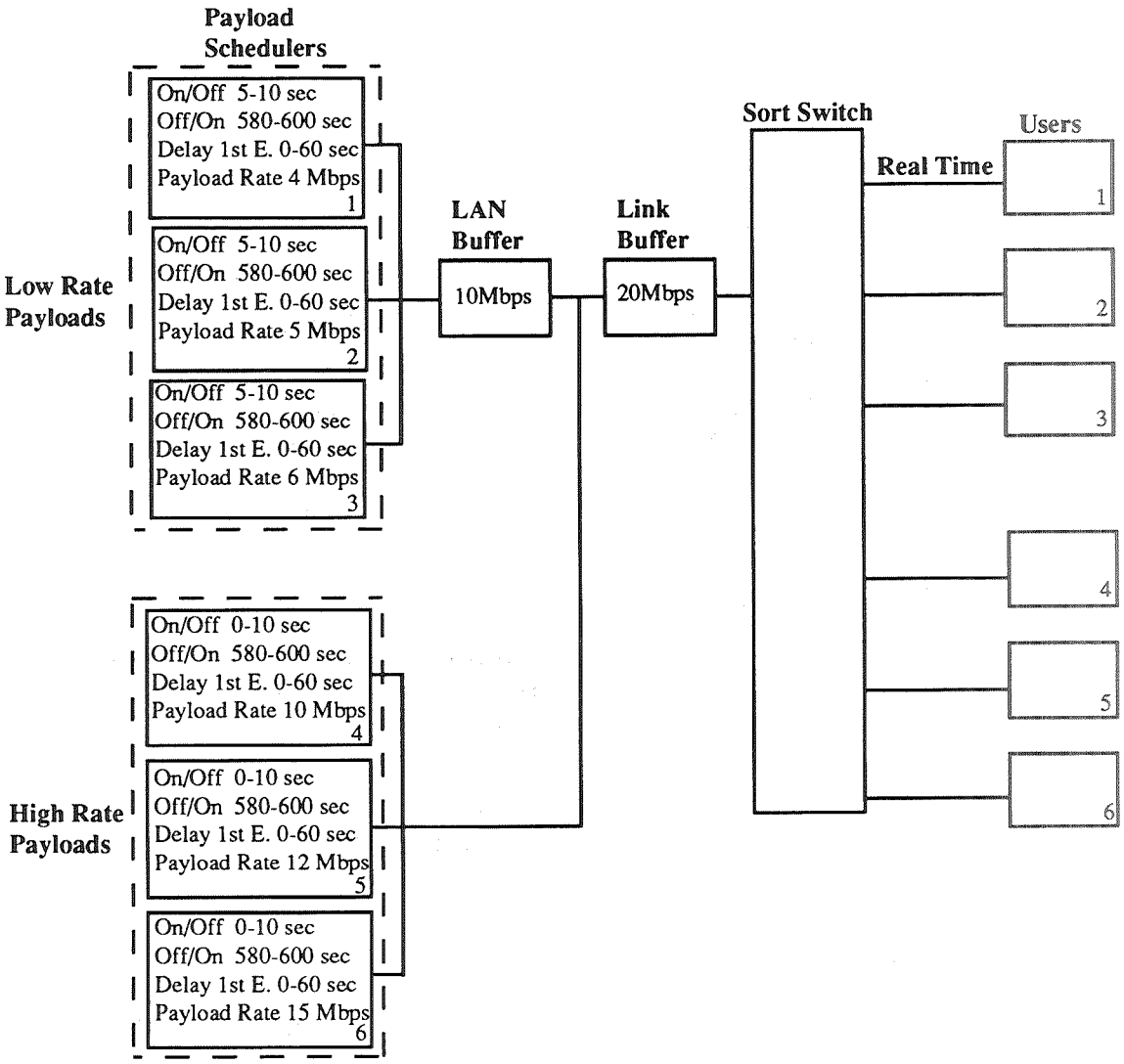

Figure 2: System used to compare data streams, artificially inflated packet and truth models.

Figure 4 illustrates the percentage of difference for the data's mean transit times from source to sink of the expanded packet model and the data stream model when compared to the truth model. As can be seen, there is a significant amount of error introduced in the mean transit times of the data when the packet size is artificially increased. The mean transit time errors in the data stream model were negligible when compared to the expanded packet model for all six payloads.

A second experiment was also designed to determine the reason for the mean transit time errors found when artificially expanding the packet sizes. It was speculated that the error was due largely to the amount of queuing experienced in the system, so the second system was designed so the data reached the user not in near real time, but within an hour, which would reduce the amount of queuing in the system. The three models (truth, expanded packet, and data stream) were again constructed and executed. Figure 5 shows that the percentage of difference when comparing the expanded packet and the data stream model models to the truth model is not as significant in the one hour data delivery system as the near real time system. However, in both systems the data stream model had less error than the expanded packet model.

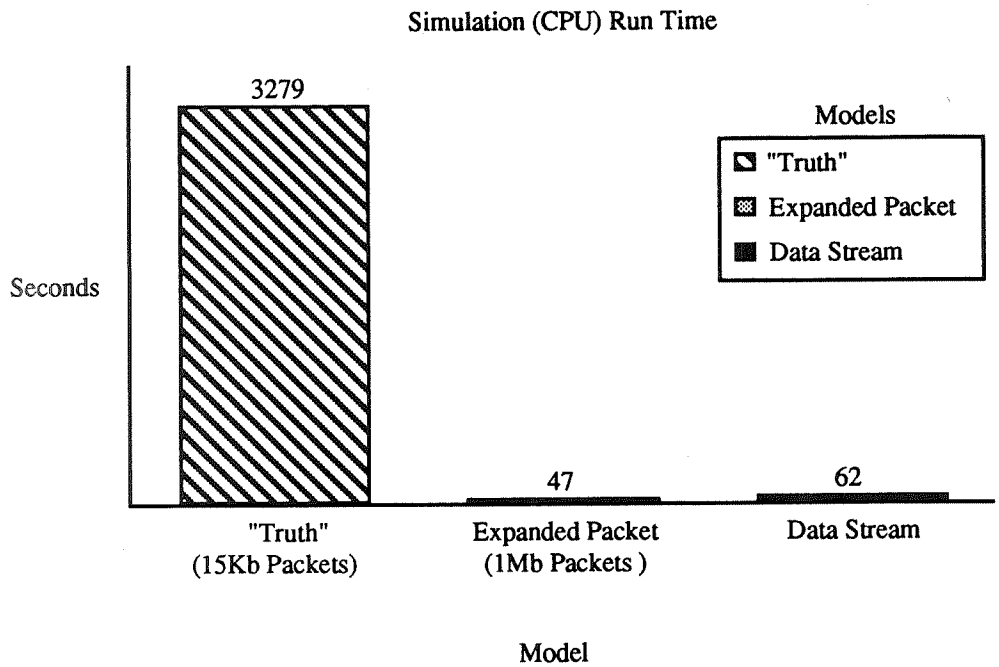

Figure 3: Comparison of simulation times of three different models.

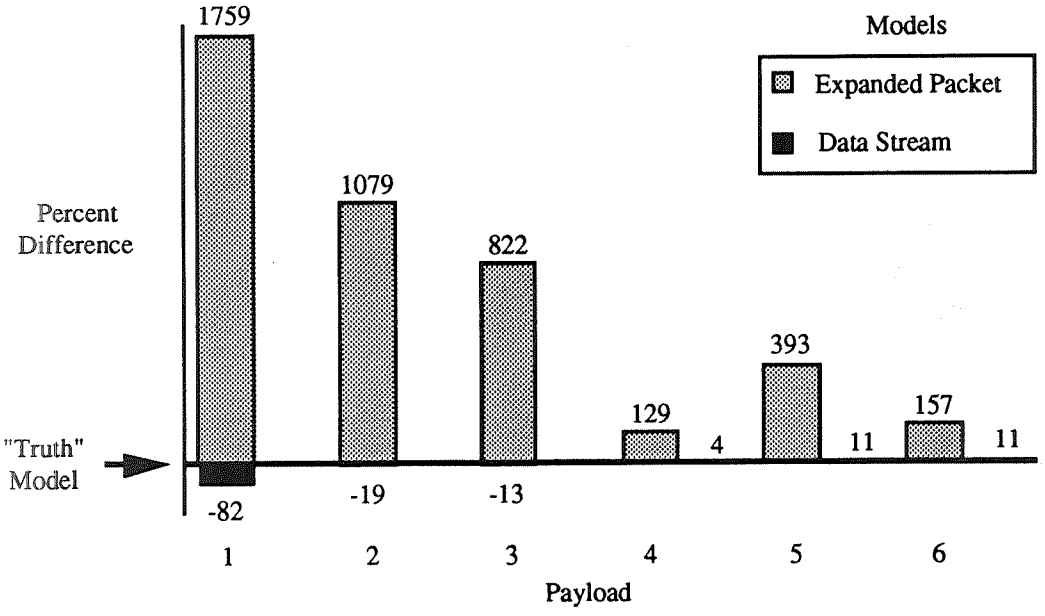

Figure 4: Mean Transit Time Difference From "Truth" Model

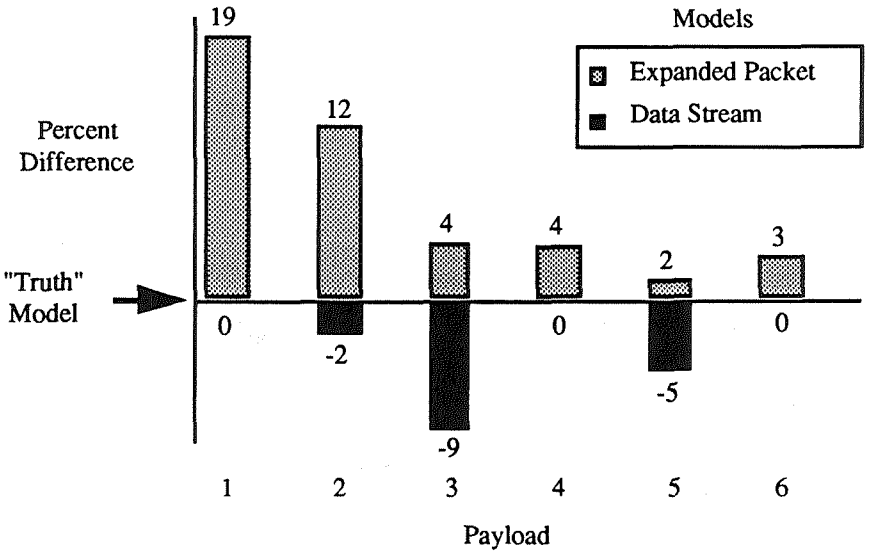

Figure 5: Mean transit time difference for one hour data delivery system.

COMMERCIAL POTENTIAL

Modeling has always been a valuable method for determining and validating requirements and designs for network-based systems. This is becoming even more important today. Competition in bids for commercial, military and other government contracts and a decrease in available make simulations of networks during the requirements and design phases of systems more important. In addition, because the systems are becoming larger and more complex, they are becoming harder to understand, and it is more difficult to predict their performance. By modeling these systems early problems can be detected. Modeling saves time and money in the long term by helping to identify potential problems (e.g. bottle necks). Lastly, detecting problems before development has the potential for large savings. The earlier errors are detected in a design, the less expensive it is to fix.

The commercial potential of DSDS is evident when compared with recent releases of other modeling tools. Other simulations use only the packet model simulation and not the data streams method. These systems require more simulation time and are more prone to error for the modeling of large end-to-end systems. Though the current commercial tools do not support the data streams methodology, vendors have expressed a desire to incorporate it into their products.

One commercial application of data streams is obvious: they can be used to simulate data management systems, whether NASA related or not. Packets can be used to represent entities in the general sense; people, cars, planes, boxes, bags or pencils to name a few. Similarly, data streams can be used to model the flow of people, cars, planes, boxes, bags and pencils. Data stream modeling is not intended to replace packet modeling. However, data streams can be used effectively as an alternative to artificially inflating the size of packets.

CURRENT WORK

Work on DSDS is still being performed. We are currently adding a graphical user interface, investigating an expanded use of data streams, and developing of an integrated simulation system. The following paragraphs discuss each of these areas.

Graphical User Interface

The current user interface for DSDS is a menu-based system where users type in the name of each element and the names of other elements that the first is connected to. A new graphical user interface is being developed that will allow users to draw the elements of the model and physically connect them together on a bitmapped screen. This will allow modelers to see their model as they develop it. The new interface will also support hierarchies so that the model can be developed in a structured, top-down or bottom-up fashion. Figure 6 shows an example of a model drawn using the new graphical user interface.

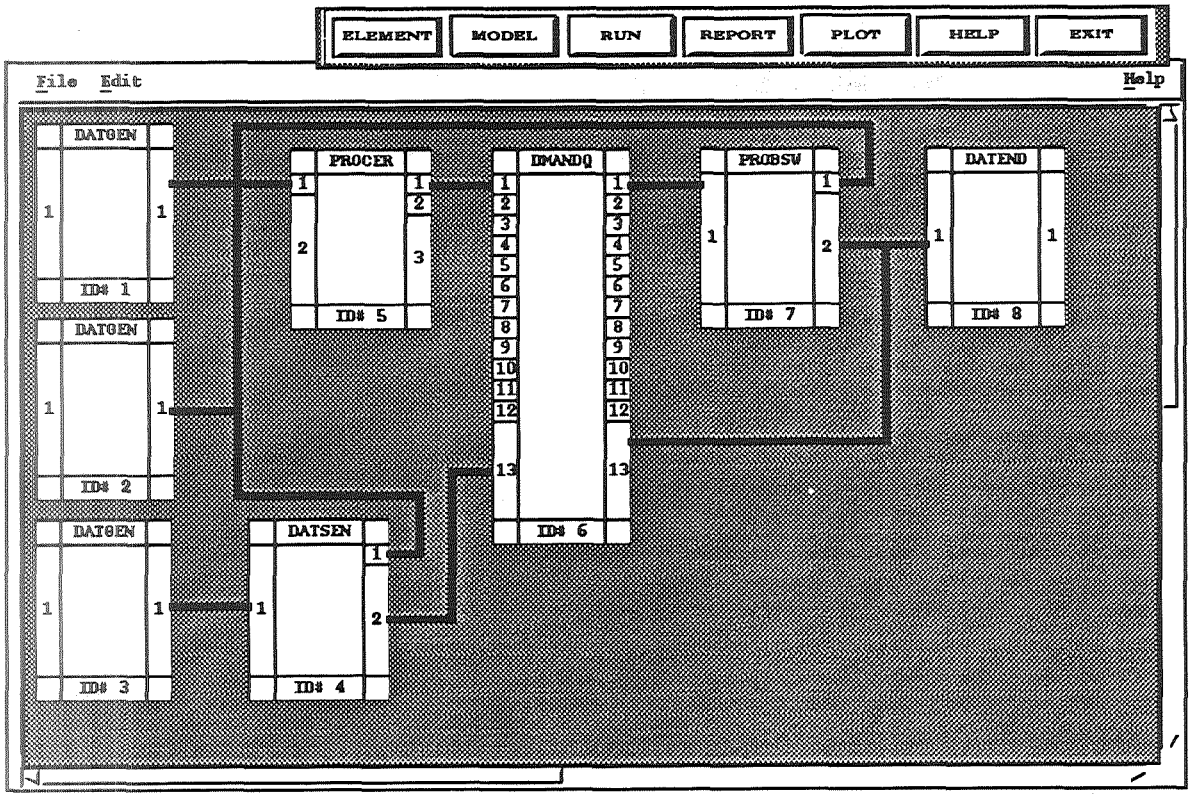

Figure 6: New graphical user interface for DSDS.

General Purpose Modeling

To date, the use of data streams has been limited to modeling of large, high-level, end-to-end data systems. These models have been characterized by duty cycles which produce millions of packets in a continuous mode. A recent study by Code 520 at GSFC has also demonstrated the potential of data streams for general purpose modeling. The study concentrated on the suitability of data streams for non-linear arrival patterns. Simulations were run with varying numbers of packet arrivals being simulated as streams. As expected, the streams which emulated the least number of arrivals gave the best results. The study reinforced the conclusion that data streams, when used for the purpose of optimizing CPU run time, are more accurate than artificially-inflated packets. In any case where the real packet size must be violated, the data stream option should be considered.

Signal Analysis Modeling System (SAMS)

An integrated modeling environment called the Signal Analysis Modeling System is also being developed to support pre-mission specification, storage, archival and analysis of spacecraft signal data. To achieve this, SAMS contains a relational database system (Oracle), an Experiment Scheduling System (ESS), a simulation system (DSDS), and a desktop publishing system (Asterix). The Oracle database contains thousands of records describing the commands to be sent to the spacecraft, together with the measurements to be collected by the onboard sensors in response to the command stimulus. This database information is exported to the scheduling system, which generates a timeline of operations by scheduling the use of the

onboard activities in such a way as to resolve conflicts. The ESS timeline is then exported to DSDS and used as a driver to populate a simulation model of the end-to-end NASA data system. The DSDS model then yields insight into the performance of the individual and aggregate data processing and data communication sub-systems. Finally, the Asterix desktop publishing system is used to capture the input data and output products generated by the other tools, such that a single, comprehensive report can be produced documenting the results of all the analyses performed.

CONCLUSION

DSDS has been used for a number of years on NASA projects but has not been released to the general public. The combination of the packet modeling, data streams modeling and extensibility makes the tool versatile and suitable for many different modeling situations. The data streams optimization technique allows accurate modeling of high data rate systems that could not be modeled accurately using the expanded packet optimization technique. As more high data rate systems are used in government and commercial applications, a technique such as data streams will be come essential for modeling their performance in a timely and accurately manner.