DISCRETE EVENT SIMULATION AND ANALYSIS OF DIS NETWORK ARCHITECTURES

Dr. James W. Dille and Steven D. Swaine
McDonnell Douglas Training Systems
St. Louis, Missouri

ABSTRACT

The Distributed Interactive Simulation (DIS) network architecture and protocol is being developed for the interconnection of large numbers of manned and unmanned simulators. However, little data exists for the performance of this system when implemented on the large scale envisioned. As with most modern communications networks, the complexity of this system is such that traditional analytic techniques such as queueing theory are incapable of accurately predicting system performance. A simulation tool for the analysis of a DIS network is written using Simscript II.5, a discrete event simulation language. The simulation employs models of typical DIS entities such as aircraft, ground vehicles and infantry and can be a valuable tool for evaluating network performance due to different amounts and mixes of entities, dead reckoning error criteria, and various network hardware and protocols. The simulation makes available extensive statistics detailing the performance of the overall system, making it possible to answer such questions as the number of entities that can be supported on particular network hardware, interrupt rates and the transport delays experienced by individual entities. To illustrate the use of this tool, a DIS network composed of high performance aircraft is examined.

ABOUT THE AUTHORS

James Dille received his Ph.D. from Harvard University in 1987 in applied mathematics and computer science, specializing in queueing theory and the analysis and optimization of computer and communications networks. He has been with McDonnell Douglas since 1987, working in the areas of simulation networking and computer architectures, particularly for the low cost reconfigurable trainers of the MASS program. He has been involved in SIMNET or DIS networking demonstrations at every I/ITSC show since 1988, and is active in the DIS protocol standardization effort. Dr. Dille is also a professor at Washington University in St. Louis, where one of the classes he teaches is the simulation and analysis of discrete event systems.

Steven Swaine graduated magna cum laude from the University of Missouri at Rolla in 1986 with a B. S. in Electrical Engineering. He has worked at McDonnell Douglas since 1983 and in simulation and training since 1987. His primary responsibility is in the area of research and development related to low cost simulation systems, particularly in the areas of microprocessor systems and communication protocols. Mr. Swaine is the Principal Investigator for the MASS program at MDTS which has pioneered low cost networked aircraft simulations. He also has been involved in the I/ITSC networking demonstrations and is active in the development of the DIS network standard.

DISCRETE EVENT SIMULATION AND ANALYSIS OF DIS NETWORK ARCHITECTURES

Dr. James W. Dille and Steven D. Swaine
McDonnell Douglas Training Systems
St. Louis, Missouri

INTRODUCTION

Large multi-vehicle simulation and training environments are becoming increasingly important as the focus of training shifts from the individual to the team. The implementation of large networks of simulators is being addressed by the Distributed Interactive Simulation (DIS) standard, which is now in draft form and is being submitted for consideration as an IEEE standard [1]. It has followed directly from the networking protocol developed during the DARPA/U.S. Army SIMNET program.

Simulators on a DIS network communicate through the exchange of Protocol Data Units (PDU), the most common being the Entity State PDU which contains vehicle state information such as position, orientation and appearance. This exchange takes place over standard communication services such as Ethernet or FDDI.

A fundamental aspect of the DIS protocols important to this discussion is the use of dead reckoning. In a typical implementation, a simulator calculates the latest position and orientation of a remote entity using the last known position, orientation, linear and rotational velocities and the time that has passed since that information was known to be valid. Using the same algorithm, the sending simulator calculates this estimate of his vehicle state that other simulators will be using, compares it to his true state and based upon some error criteria, decides whether or not to send a new Entity State PDU. In other words, the sending simulator decides if his position as calculated by the other entities on the network is sufficiently accurate without sending another update.

Since entities such as tanks and aircraft spend significant portions of time moving in relatively constant velocity paths, dead reckoning can make large reductions in the number of entity state updates [3]. What can follow from the reduction in bandwidth is an increase in the number of entities and thus the scale and complexity of the simulated battle that can be supported on a given physical communications network. Empirical evidence from both SIMNET and DIS implementations show significant reductions in network traffic. While substantial benefits can be observed on operating networks, the effectiveness of dead reckoning and its relationship to important parameters such as the error criteria, number and mix of entity types on the network, and the behavior of the simulated entities remains difficult to quantify.

THE PROBLEM

The DIS protocol is being developed to be implemented on a scale previously untested by even the largest SIMNET implementation. Every discussion of the goal of DIS indicates that thousands of simulators are envisioned to be interconnected into a single large environment. With the encouraging results of the SIMNET program, and the continuing development of ever higher bandwidth communication services, it is reasonable to assume that an environment this large will someday be possible. At the same time, only rough estimates currently exist for how large such an environment could be using specific existing and emerging communications technologies and at what cost to the training task in degraded network performance.

Without better information concerning the demands of DIS entities on the underlying communication network, a sound engineering approach to the implementation of large DIS environments is impossible. When considering anticipated configurations, it would be desirable to have a tool that could predict such important factors such as network utilization and transport delay. We would like to quantitatively answer such questions as these before the system is built:

- How many simulators will we be able to interconnect on existing networks such as Ethernet?

- How many tank simulators can I run on a single local Ethernet and have no more than 1% of my entity state messages suffer transport delays of over 30 milliseconds?

- Will my anticipated configuration run sufficiently better using FDDI to justify the additional cost over an Ethernet?

- I know that the current hardware that I am using can handle incoming Ethernet packets at a rate of only 1000 per second. Will I be able to use this device in a DIS environment currently supporting 100 high performance aircraft?

- If I use a long-haul connection between two local networks, how far apart can the sites be and still keep the average error between true positions and dead reckoned positions less than 1 meter?

Worst Case and Average Rate Analyses

It is possible to perform a simplistic worst case analysis based on the premise that at crucial moments every entity in the simulation is actively maneuvering and thus sending entity information at their frame rate. For example, consider a network of simulators operating at frame rates of 30 Hertz and using a standard Ethernet and the DIS protocol as currently proposed [1,2]. With an average of 2 articulated parts per simulator, and the standard Ethernet 802.3 headers, each Entity State PDU totals 1520 bits. If 60% of the theoretical Ethernet bandwidth were used for actual data transfer (a reasonable estimate), we could support about 130 simulators. At this network load, contention for the Ethernet would produce significant transport delays. Fewer than 100 simulators would be a better estimate for the number that could be supported on an Ethernet.

This simplified analysis has led to far too conservative an answer by ignoring dead reckoning and the crucial assumption upon which the effectiveness of dead reckoning is based, i.e. that in a *sufficiently* large battle not all of the entities would be actively maneuvering at any one time. This concept of statistical multiplexing is inherent in the operation of many modern networks, such as broad-band packet switching networks. In other words, when large number of independent sources of "bursty" traffic are present, it is unnecessary and unwise to size your network to the worst case.

If either estimates of, or empirical data supplying average entity state broadcast rates are available, the analysis can be improved by bringing in the effects of dead reckoning at an aggregate level. For example, we may observe that tank simulator broadcast packets at a particular rate. We could factor this into the analysis and calculate a more reasonable bandwidth requirement. There are papers available detailing such information [3]. While improved, this analysis still fails in several respects. It can calculate only average bandwidth requirements, assuming constant rates of packet transmission and ignoring the "bursty" nature of the traffic which lead to the occasional network overloads which need to be considered. Like the worst case analysis, it may supply estimates of bandwidth requirements and network utilization but can still give no insight into important statistics such as transport delay or positional errors.

While the worst case analysis overestimates the demands on the network, the average rate analysis underestimates them. With these simple analyses ineffective, we must develop a better model of the system and bring to bear more sophisticated methods of analysis.

Modeling Approaches

A DIS network implementation is a complex example of a general communications network problem. Communications networks in turn are examples of a very broad class of systems known as Discrete Event Dynamic Systems (DEDS).

A Discrete Event Dynamic System is one in which the state of the system changes at discrete points in time rather than continuously with time. In the terminology of queueing systems, which includes communications systems such as DIS, these state changes can be considered to result from the interactions between the two basic elements of a DEDS, customers and resources. In this case, customers (simulators) require service from resources (networks) which possess only limited capacity. Complex rules govern such interactions.

For example, in an Ethernet, the state variable of how many nodes are currently wanting access to the network may jump from 2 to 3. The event of a node having a packet ready for broadcast causes the state change. There are never 2.1 or 2.2 nodes waiting to broadcast; the state of the system is not continuous. Complex rules such as that contained in the collision detection and resolution algorithms of the Ethernet interface hardware and software help drive the evolution of this system.

This is fundamentally different from a continuous physical process such as a chemical reaction or the flight of an aircraft, where a state variable such as the concentration of an element or the aircraft's position changes continuously with time. It is important to understand the difference between these two different types of physical processes. In particular, the simulation of a discrete event system such as a communications network is *not* the same as the simulation of an aircraft or tank where a digital computer simulates the evolution of a continuous system using discrete time steps. The way dynamic systems can be analyzed also differs greatly. With a continuous system, the evolution of the system can be described mathematically (however difficult that may be) and analyzed using tools such as calculus. With the evolution of a DEDS described by a complex set of rules or algorithms, standard analysis techniques are not applicable.

A specialized field of mathematics known as queueing theory offers some tools to analyze such systems but are limited in their application. As with most modern communications networks and protocols, the complexity of a DIS network is such that traditional analytic techniques such as queueing theory are incapable of accurately predicting system performance. The necessary assumptions for the distributions of message traffic and message sizes, and for the behavior of the underlying communications network that would have to be made in order to fit an analytic model render the outputs of that model of little use.

An accurate quantitative performance analysis of a system as complex as a DIS network must therefore rely on observation and experimentation, either on an existing system or on a simulation of such a system. With the goal of being able to predict performance, we have no appropriate existing systems and must consider simulation. A flexible tool based upon discrete event simulation of DIS network architectures has therefore been developed for this purpose.

DIS SIMULATION MODEL

The DIS network simulation model is implemented in Simscript II.5, a discrete event simulation language. Simscript is a software product that has been widely used for the analysis of complex discrete event dynamic systems. It has been successfully used to analyze communications networks at Hughes Aircraft, Rockwell International, and NASA.

Simscript uses what is known as the process approach to DEDS simulation. The behavior of each of the active entities or processes in the simulation is contained in a single software description. Resources are defined and then the interactions between various processes and between processes and resources are handled by the simulation language. The programmer does not have to be concerned with simulating these complex effects. The collection of statistics is automated and the English-like syntax yields code that is virtually self-documenting.

Software Design

The primary consideration when developing the DIS network simulation was to create an effective analysis tool. Several goals were identified.

- The program should be data file driven to avoid having to alter the source code in order to merely change the configuration of the network being considered, allowing the use of the tool by people who do not possess the Simscript compiler.

- A modular design is aided by the Simscript language itself, but extra care was taken to ensure that enhancements to the program such as adding an FDDI communications network to replace the Ethernet network or an improved entity model would be as simple as replacing a single routine.

Communication Architecture Models

The first communications service to be modeled is an Ethernet 802.3 protocol. This is a common network, used for the SIMNET local area networks and for the 1992 I/ITSC DIS networking demonstration. It uses carrier sense multiple access with collision detection (CSMA-CD).

Multiple access means that all stations share the same channel and carrier sense means that no station will begin to transmit a packet if it detects data from another station on the channel. Because two or more stations could begin to transmit at the same time, they listen while they are transmitting and if they detect such a collision, they retry at some random time in the future.

Every detail of this process as described in the IEEE 802.3 specification [2] was transformed into the Simscript language, modeling every access, collision, backoff and retry, etc. While the intent of this simulation tool is not to build a model of an Ethernet, the accurate representation of the actions of the underlying network is critical to the *evaluation of the effectiveness of a DIS* implementation.

Some of the parameters in the 802.3 specification are not firmly defined, but only recommended and some parameters are related to the length of the network. The following parameters were used:

Bandwidth	10 Mbits per second
Slot Time	512 bit times
Jam Size	32 bits
Min Frame Size	64 bytes
Max Frame Size	1518 bytes
Interframe Gap	9.6 microseconds
Station Addresses	48 bits

All of these values constitute the most common Ethernet implementation. Another parameter is the time it takes for a signal to reach other nodes. This is a function of the physical network and is set to half of the slot time, modeling an Ethernet which is half of the maximum 500 foot length.

Entity Models

It is in the design of the entity models and their network interfaces that the actual DIS protocol is reflected. The DIS simulation supports three different types of entity models as discussed below. More than one type of model can be used in a particular run of the simulation. The entities generate requests to their network interfaces for the broadcast of entity state packets. At this time, only the entity state packet has been modeled. Packets such as fire and detonate packets occur so relatively rarely that their effect is negligible, but packets such as radar and emissions may make significant contributions to bandwidth requirements. Such packets will be added as enhancements to the simulation as they become better defined in the DIS standard.

All three models share the common fundamental problem of making sure that their behavior is representative of the entity behavior that will be taking place in the actual network. They address this issue in different ways.

High Fidelity Model. This model calculates the position and orientation of the entity every frame, essentially implementing the entity simulation itself. With this information available, the simulation then uses the DIS dead reckoning algorithms and error criteria to determine whether to send the packet. An advantage of this model is that extensive statistics can be collected as will be discussed below.

However, a separate piece of code must be developed for each kind of entity; tanks, airplanes, foot soldiers, etc. It is easy to make entities such *tanks drive in a circle or a square, but the* complexity grows rapidly as you try to generate vehicle movements that are representative of those that take place in the real world. Validation that you have a reasonable model is also difficult. Another problem is the impact these calculations have on the computational requirements of the DIS simulation itself. As with all Monte Carlo simulations, and particularly considering both the detail and size of this one, excessive computation should be avoided if possible.

Distribution Model. The distribution model describes the entity state broadcast behavior of an entity in terms of a mathematical distribution. This can then be used in the simulation to generate the instances when that entity sends an entity state PDU.

The simplest representation would be the mean rate at which entity messages are sent. By observing an existing simulation running a DIS protocol, this rate could be measured and used as a distribution parameter within the DIS simulation. Unfortunately, a simple mean rate statistic has lost the "bursty" nature of entity motion and is therefore unsatisfactory for reasons discussed earlier. What is used instead is only a slightly more complicated distribution based upon the premise that an entity has two rates of entity state transmission, high and low, corresponding to an actively maneuvering state and relatively constant velocity motion. Four parameters are thus required; the low and high rates of transmission and the average amount of time spent in each of those two states before transitioning to the other. This data can be collected from operating simulators with little more work than collecting a simple average rate.

This model requires considerably less computer power than the full fidelity model and is more easily validated as representative of true entity behavior since observed data is the source of the distribution parameters. Another advantage of this approach is the fact that for entities whose simulation has not yet been implemented, educated guesses about their broadcast rates can be used in the evaluation of a prospective network.

A disadvantage of this approach is that position data is not available, so statistics such as the distance between the true and dead reckoned positions cannot be collected. Another disadvantage of this approach is that the actions of the dead reckoning algorithms are contained in the distribution data and not explicitly performed in the simulation. Experimentation with different dead reckoning rules and error criteria would necessarily involve new observed data and the generation of new distribution parameters.

Data Model. The third entity model that the simulation supports is the data model, which uses *raw data collected from existing simulators*. Data files are supplied to the DIS simulation which contain frame by frame position and orientation and their associated velocities and accelerations. With entity motion specified in this way, as with the High Fidelity model, the simulation itself performs the dead reckoning and allows for explicit experimentation with error criteria and dead reckoning models. The model is as representative of the real world as the data collected. Since one data file is likely to be used for many entities of a particular type in a simulation, their starting points within that file are selected randomly.

Sample Input and Output

The program inputs are supplied in a data file, an example of which appears in Figure 1. Comment lines begin with an asterisk. The example shown is for a DIS network consisting of 10 aircraft and 50 tanks. First the length of the simulation. data output intevals and a statistics interval (discussed below) are defined.

```
************************************************************
************************************************************
* Run Length      (seconds)
* Interim Result Interval    (seconds)
************************************************************
 100.0
  20.0
************************************************************
* Statistics Interval    (seconds)
************************************************************
 0.2
************************************************************
*  Dead Reckoning Algorithm
*  Translational Error Criteria   (meters)
*  Rotational Error Criteria      (degrees)
************************************************************
 1P1R
 0.5
 2.5
************************************************************
*  Entity Declarations
*
*  Formats
*                                           Low         High
*   Name  DIST_MODEL   Frame_Rate   Time  Rate  Time  Rate
*
*   Name  DATA_MODEL   Data_File_Name
*
*   Name  HF_MODEL     Frame_Rate   Routine_Name
*
************************************************************
AIRCRAFT   DIST_MODEL  30Hz   30.0   4.0     15.0   25.0
TANK       DATA_MODEL  tank.path
************************************************************
*  Entities
*
*  Format
*    Name      Number     Site Host Entity
*                         (0 0 0 assigns them automatically)
************************************************************
AIRCRAFT     10           0 0 0
TANK         50           0 0 0
************************************************************
************************************************************
* Options
************************************************************
Individual
Network Summary
System Summary
Position Errors
************************************************************
```

Figure 1 -- Input Data File

Next, the entity types in the simulation are defined, in this case a 30 Hz aircraft using the distribution model with the parameters as indicated, and a 15 Hz. Tank as a data model with the information located in the file "tank.path". The number of each of these entities is defined and then a series of statistics and output options are specified.

The output file corresponding to this input file is shown in Figure 2. One section that appears in the file has been deleted in the interest of space. At the top of the output file, the input file is echoed so that every output file can be used as an input file if the experiment needs to be rerun and the input file is no longer available. Also in the interest of space, much of the individual entity statistics have been cut out.

```
########################################################################
                            FINAL RESULTS
  Total Simulation Time =  100.000000 seconds
########################################################################

########################################################################
   NETWORK UTILIZATION STATISTICS
########################################################################
 Local Network        Utilization %    Arbitration %     Free %
_____________________________________________________________________
       1                2,1038            0,1238          97.7724

########################################################################
   MESSAGE COUNT AND RATE STATISTICS
########################################################################
                                  SEND                       PROCESSED
  Entity               Total       Rate            Total          Rate
_______________________________________________________________________

  1   1:  1:  1        1213     12.1300001         40862     408.6199951
  2   1:  2:  1        1170     11.6999998         40907     409.0700073
  .   .   .   .          .           .               .            .
  .   .   .   .          .           .               .            .
 59   1: 59:  1         596      5.9600000         41472     414.7200012
 60   1: 60:  1         587      5.8699999         41474     414.7399902
_____________________________________________
 Total                42077    420.7699890

########################################################################
   Instantaneous Send Rates
########################################################################
        Entity              Minimum         Average       Maximum
_____________________________________________________________________
  1   1:  1:  1        10.7000000     12.1888889     13.1000000
  2   1:  2:  1        10.7000000     11.7111111     13.0000000
  .   .   .   .             .              .              .
  .   .   .   .             .              .              .
 59   1: 59:  1         5.5000000      6.0666667      7.1000000
 60   1: 60:  1         5.3000000      5.9555556      6.7000000
_____________________________________________________________________
 Total                  3.9000000      7.5296292     13.6999998

########################################################################
   Instantaneous Receive Rates
########################################################################
        Entity              Minimum         Average       Maximum
_____________________________________________________________________
  1   1:  1:  1       401.1000000    409.5555556    415.1000000
  2   1:  2:  1       401.5000000    410.0555556    415.5000000
  .   .   .   .             .              .              .
  .   .   .   .             .              .              .
 59   1: 59:  1       406.5000000    415.5555556    420.1000000
 60   1: 60:  1       407.2000000    415.5000000    421.2000000
_____________________________________________________________________
 Total                399.2000122    414.6004944    422.2999878

########################################################################
   TRANSPORT DELAY
########################################################################
        Entity              Minimum         Average       Maximum
_____________________________________________________________________
  1   1:  1:  1          .0000500       .0168913       .0334333
  2   1:  2:  1          .0000500       .0170508       .0334833
  .   .   .   .             .              .              .
  .   .   .   .             .              .              .
 59   1: 59:  1          .0000500       .0352799       .0667667
 60   1: 60:  1          .0000541       .0347827       .0667667
_____________________________________________________________________
 Total                   .0000500       .0306052       .0668167

########################################################################
   DEAD RECKONING
########################################################################
 83.4 % of Entity State messages eliminated by Dead Reckoning

########################################################################
   POSITION ERRORS
########################################################################
                        Average         Maximum
______________________________________________
        X               0.17547         0.99689
        Y               0.07062         0.88989
        Z               0.14419         0.83203

     Distance           0.28680         1.15323
```

Figure 2 -- Output Data File

Statistics

One of the powerful features of Simscript is its ability to collect statistics. Network statistics include utilization broken down into the bandwidth required for actual data communication and the bandwidth lost in arbitration for the Ethernet.

Other statistics are entity based and are collected both for individual entities and averaged across entities. They include the number and rate at which entity state messages were sent and received. These are simple long term average rates calculated as the number sent or received divided by the total time of the simulation run.

Another calculation is used for the instantaneous send and receive rate statistics. These rates are calculated over an interval defined in the input file. Since an arrival of a PDU may generate an interrupt, it is important to know at what rate these may occur to know if a simulator can handle the flood of packets. We are not interested in the overall average rate, but an instantaneous rate, for we need to be able to handle the occasional high rate of arrivals. To collect this statistic, the user selects the appropriate value for this interval in the input file.

Transport delay is defined as the time elapsed from when the data was valid at the sending simulator to the time the PDU was processed by the receiving entities. The effect of the asynchronous frame rates of the simulators is seen in this statistic in addition to delays due to the network.

As mentioned earlier, if the Data or High Fidelity Model is used, more statistics are available, since position and orientation data and the dead reckoning is being performed explicitly. The percentage of the frames where dead reckoning has prevented an entity state broadcast is calculated. Also, the average and maximum error between the true position and orientation of the entities and that estimated by the other entities on the network is calculated. These error statistics are computationally expensive to collect and the user can request them in the input file if desired.

EXPERIMENTAL RESULTS

To illustrate the use of the DIS simulation tool, a set of experiments analyzing the performance of a DIS network of high performance aircraft were performed. It is important to stress the fact that this data is not to be interpreted as a definitive answer for the performance of DIS networks in general, but is presented only to show possible uses of the simulation tool. Specific results are highly dependent on the application; the number and mix of entities, frame rates, dead reckoning algorithms and criteria, entity behavior, and the accuracy with which that behavior is modeled.

In this example, we look at a network consisting solely of high fidelity aircraft. The entity model used is the data model with 15 minutes of representative flight data captured from an F-15C simulator. The dead reckoning algorithm selected is a first order algorithm operating on both position and orientation. The error criteria used was 0.5 meters for translational error and 2.5 degree for rotational error. A number of simulation runs were performed in order to show the effect of the number of simulators on the various performance parameters. Then for a particular configuration of 500 aircraft, the dead reckoning error criteria are adjusted to investigate their effect on network performance.

Network Utilization

Figure 4 shows network utilization as a function of the number of aircraft on the network. The total utilization is the percentage of time the network is busy, i.e. being accessed. The useful utilization is the percentage of the time that the network is actually moving DIS data. The difference between the two curves is the loss due to the packet collisions that are used to arbitrate access to the Ethernet.

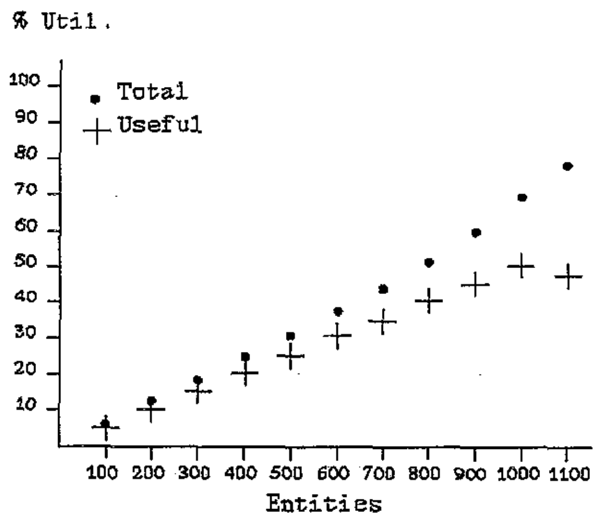

Figure 3 -- Network Utilization

The utilization results show no hard limit to the number of entities that can be supported, and that is the case with all of the results. What is shown is the decreasing performance of the network in response to the increased number of entities. The figure also shows a drop in the useful bandwidth above 1000 entities. At that network load, almost 20% of the packets are failing to get access to the network and are giving up as per the Ethernet protocol. The sending entity is informed of the failure and then attempts a send in its following frame, but there is no higher level retransmission policy in effect. This produces the dip in actual data transferred, while the overall utilization continues to rise.

Transport Delay

Transport delay as a function of the number of aircraft is shown in figure 4. Transport delay includes the normal effects due to the asynchronous simulators. Although all of the entities are running at 20 Hz, they are randomly distributed in time. Therefore, without any network delays, the transport delay statistic can be as large as twice the frame time, or 50 milliseconds. The average would be 25 milliseconds. The effect of the increasing number of entities and the larger network load is seen through the increase above these base values and it is this component due to the network loading which is plotted.

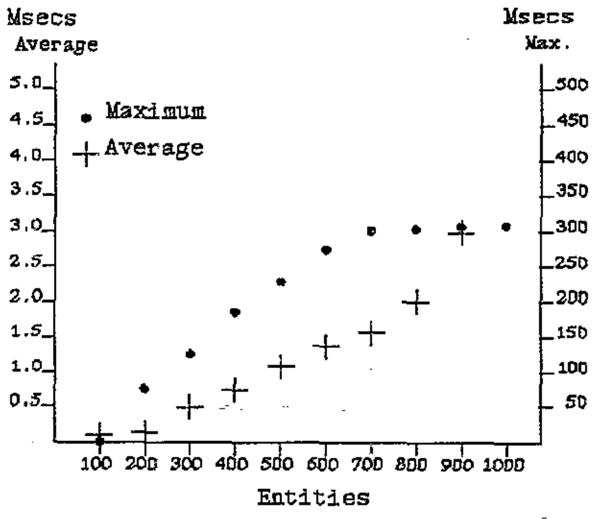

Figure 4 -- Transport Delay

The figure shows a steady increase in the average transport delay, up to approximately 700 entities, and then a rapid increase until the average delay is over 8 seconds with 1000 entities. The maximum delay also rises steadily, but it is then limited to about 300 msecs as packets that might take longer than that are being thrown out due to too many collisions.

As the figure indicates, for this network with fewer than about 600 entities, the average effect of the network delay is negligible compared to the effect of the asynchronous frames. In fact, the errors due to the transport delay are probably negligible in the face of the dead reckoning errors that are going to be allowed to build up once this data has been received. At reasonable network loads, transport delays due to network loading could only become significant where high frame rates and extremely small error criteria are used.

Position Errors

Possibly the most significant statistic is the error in the position and orientation of the entities, for supplying this data accurately and efficiently is the primary purpose of the DIS network. Figure 5 shows the average distance between true entity positions and the estimates used by remote entities.

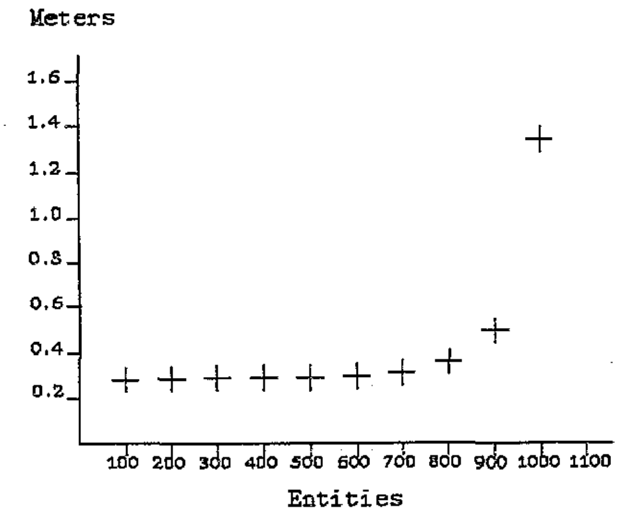

Figure 5 -- Position Errors

As the data shows, the error is steady at a little over 0.25 meters for less than 700 entities, a reasonable average for an error criteria of a cubic meter. Above 700 entities it climbs rapidly due to lost and delayed packets until at 1100 entities, the average error exceeds 15 meters.

Interrupt Rates

The average packet arrival rate or interrupt rate is shown in Figure 6, again as a function of the number of aircraft on the network. This is an important parameter, for it helps indicate the kind of ethernet interface and processing power that will be necessary to handle the flood of incoming packets. The statistics shown are the average and maximum instantaneous receive rates calculated over a statistics interval of 200 milliseconds.

The smaller the statistics interval, the larger the difference between the maximum instantaneous rate and the simple long run average rate. However, too small an interval and the statistic is useless. If you consider two packets arriving almost simultaneously, the arrival rate over that small an interval is obviously extremely high, but the internal buffering capabilities of network interface hardware would handle those two packets without difficulty. Too large an interval, and this statistic is no better than the long run average which could have been obtained through much simpler analysis, but again ignoring the bursty nature of the traffic. The statistics interval used here is 200 msec.

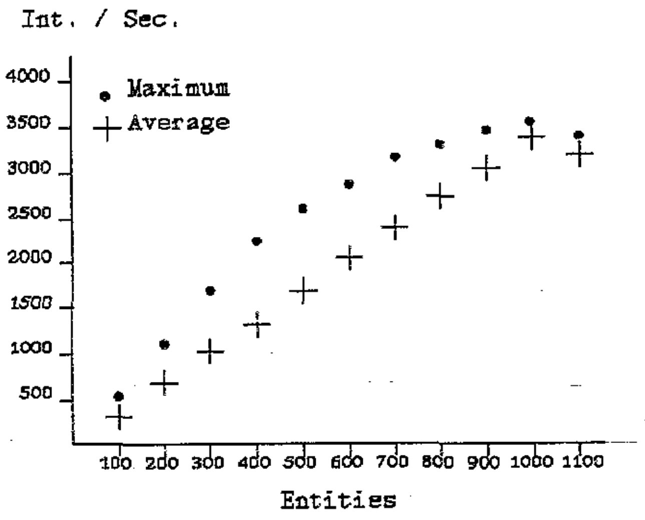

Figure 6 -- Interrupt Rates

As can be seen in the figure, packet arrival or interrupt rates can be quite high, and the instantaneous rates can be significantly higher than the simple long run averages. It is obvious that significant attention must be placed on this aspect of a DIS network interface. In fact, it is probable that the processing power required to handle the incoming packets and dead reckon remote positions will far exceed the processing power required for the local entity simulation itself.

Dead Reckoning Error Criteria

A second set of experiments was performed to investigate the effect of changing the dead reckoning criteria. The network was the same as before, but composed of a fixed number of 500 aircraft with the translational and rotational error criteria scaled linearly upward and downward from the nominal value of 0.50 meters and 2.5 degrees used in the first set of experiments.

Figure 7 shows the average distance error and the percentages of possible entity state transmissions that were eliminated due to the actions of the dead reckoning algorithms.

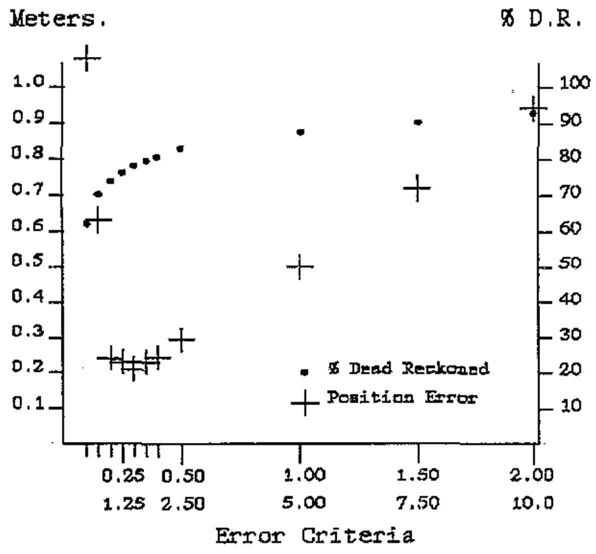

Figure 7 -- Dead Reckoning Error Criteria

As expected, as the error criteria is loosened, the percentage of entity state packets that can be kept off of the network increases. The average error stays ar about half the error positional error criteria value.

As the error criteria are tightened, the observed position error exhibits an important effect, declining to a point and then increasing rapidly as the network becomes increasingly overloaded. Additional runs were performed to determine the minimum value, showing the ability of the tool to help optimize performance of a DIS network configuration. The minimum distance error occurs when the criteria are approximately 0.3 meters and 1.5 degrees.

CONCLUSION

This simulation tool can provide valuable information to the designers of both individual DIS network entities and the network itself. A good design must consider a wide variety of factors such as the number of entities, dead reckoning algorithms and error criteria, network technology and costs, and the required fidelity for the training task. This simulation allows the user to experiment with many of these parameters and obtain quantitative performance measures to aid in decision making. The issue is complex and the DIS standard continues to evolve. We will continue to enhance the simulation as the standard develops. Currently underway are an FDDI communications layer and long haul networking.

As industry and government look forward to the large scale implementation of DIS environments, we hope that this tool can contribute quantitative performance data to aid in a sound engineering design of such networks.

REFERENCES

1. Standard for Information Technology, Distributed Simulation Applications, Process and Data Entity Interchange Formats. IEEE Project P1278.

2. IEEE Standard 802.3 CSMA/CD Access Method and Physical Layer Specifications, The Institute of Electrical and Electronics Engineers, Inc.

3. Harvey, E. and R. Schaffer, *"The Capability of the Distributed Interactive Simulation Networking Standard To Support High Fidelity Aircraft Simulation"*, Proceedings of the 13th Interservice/Industry Training Systems Conference, 1991.