Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

SIMULATION OF A LOCAL AREA NETWORK FOR A NEWSPAPER PRODUCTION SYSTEM

J. Agre, I. Shahnawaz, M. Atkinson, D. Joshi* and A.R.K. Sastry

Abstract

A computer simulation model has been developed for a Local Area Network (LAN)-based newspaper production system (NPS). The objective is to develop a model that serves as a performance evaluation and design tool for the NPS operation in a Manufacturing Automation Protocol (MAP) environment. The approach taken involved developing three modules that mimic the seven layers of the MAP/OSI (Open System Interconnection) model from a performance evaluation perspective: (1) the medium access module that is based on the ANSI/IEEE 802.4 Token Passing Bus, (2) a transport module that implements the essential functions of the ISO Transport Protocol, and (3) a user traffic module (representing the top three layers) that uses a general and flexible model to create a realistic message load based on the actual message interactions in a processing plant. While a newspaper production process is considered in this paper for specific application, the user traffic module is capable of portraying any plant with well-defined process interactions in terms of messages. Results are presented from simulation runs with the NPS data.

I. INTRODUCTION

This paper describes a computer simulation model developed to aid in the performance evaluation and design of a local area network (LAN) to meet the real time communication and control needs of a newspaper production system (NPS). The Manufacturing Automation Protocol (MAP) [1] is being widely accepted as a suitable candidate for LAN-based factory automation communications. In view of this, the communication system for the NPS is based on MAP and our simulation model attempts to capture the prominent aspects of MAP's protocols that influence the real time performance of the LAN for carrying NPS messages.

The NPS plant model entities consist of a network of nodes and presses corresponding to the communicating elements and other necessary components in the newspaper production facility subsystems as shown in Figure 1. A node is a generic term denoting a particular point or location in the Plant/Network which may communicate with other nodes. A node may be 'simple' or 'complex' (consisting of a group of simple Nodes). Simple nodes are connected by a *station* to the bus while complex nodes, called *presses*, possess a particular configuration and contain multiple stations.

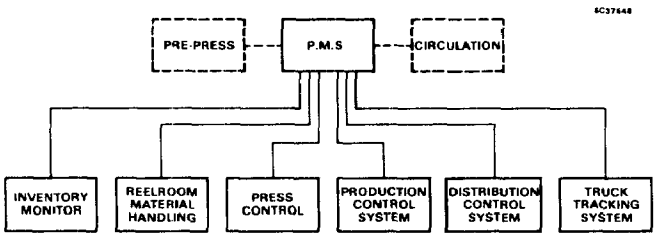

Fig. 1. Plant sub-systems

The simulation attempts to represent actual plant components which are specifiable as inputs using the NPS terminology. Examples of press components are : Automated Roll Loaders (ARL's), Bundle Entry Devices (BED's), Folders, Printing Units, Press Totalizer Interfaces (PTI's), Press Consoles (PC), Streams, etc. Examples of other simple nodes in the plant are: Inventory Monitors, Production Management Systems (PMS's), Material Handling Systems (MHS's), Truck Monitors (TM's), Production Control Host (PCH), Central Press Console (CPC), Information Management System (IMS), Front End Processors (FEP's), Docks, etc. Each of these communicating entities are assumed to be attached to the bus via a station using the MAP protocol as shown in Figure 2.

MAP is based on the 7-layer Open System Interconnection (OSI) model [2],[3] of the International Standards Organization, displayed in Figure 3, and represents a collection of standard protocols that are still evolving. The medium access scheme of the LAN interconnecting the nodes and presses uses the ANSI/IEEE 802.4 Token Passing Bus scheme [4]. The model considered in this work is a single bus (broadcast) system without any cells and bridges. The network layer is thus a null layer. ISO class 4 Transport Protocol [5] is used for layer 4 and, in our model, the Application, Presentation and Session Layers are represented together by a User module that sends and receives messages to and from the transport layer. These

* D. Joshi was with the Graphic Systems Division, Rockwell International, 3100 South Central Avenue, Chicago, Illinois 60650. He is now with Switching Systems Division, Rockwell International, P.O. Box 1494, Downers Grove, Illinois 60515. All the others are with Rockwell International Science Center, 1049 Camino Dos Rios, Thousand Oaks, CA 91360.

messages are generated by a workload model in the User module, in a time sequence based on the NPS plant operational requirements. The workload model accomplishes a mapping of the actual plant data into sequential message sources using a novel approach that is general and flexible so that it permits use of the model for many plant designs and for various operational scenarios or phases of a given plant's operation. The simulator developed has essentially three modules representing (1) the token passing bus scheme, (2) the transport protocol, and (3) the user workload protocol.

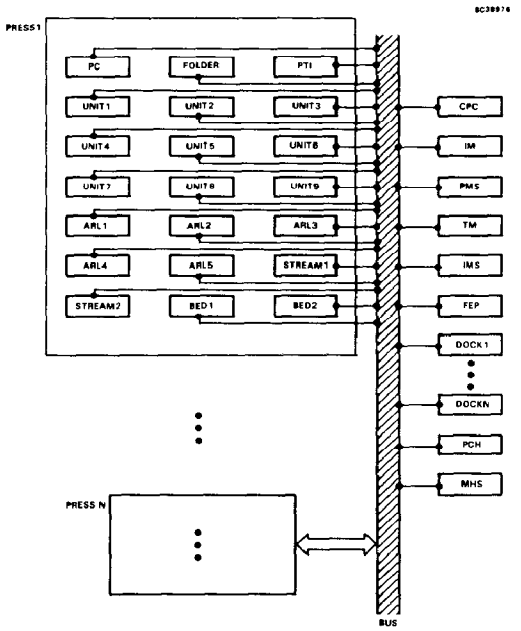

Fig. 2. Press-Station configuration

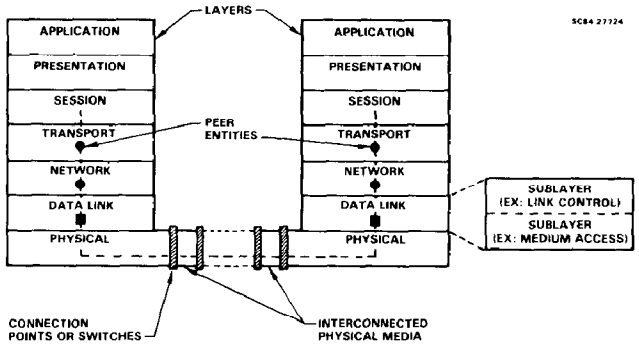

Fig. 3. Layers of the OSI model

Section II describes briefly these three processes. Section III gives detailed descriptions of their implementation as modules in the simulation model. Section IV contains the definitions of the data input parameters as they appear in the input data file, the simulation output statistical variables, a brief description of the usage of the program and its component routines, and some sample outputs of a test simulation runs. Section V gives concluding remarks and some observations.

II. DESCRIPTION OF THE MODEL ELEMENTS

Figure 4 gives an overview of the interaction among the three modules; i.e. the token passing, transport, and user modules. When a station on the LAN initiates transmission of a message to another, the message is handed by the sender station's User protocol layer to its (sender's) Transport protocol layer. After fragmenting the message into smaller transport data units, the transport protocol hands it over to the lower medium access layer (token bus protocol), which in turn creates 'frames' (one for each transport data unit) and transmits them onto the bus. The destination station's medium access protocol layer removes from each arrived frame the overhead bit string corresponding to this layer and passes the remaining transport unit to the Transport protocol layer. Transport units from the same source and belonging to the same message are then regrouped and rejoined in their proper sequence, to form the original message, which is then passed on to the destination station's User protocol layer. The specific operations in each of these modules are briefly described below.

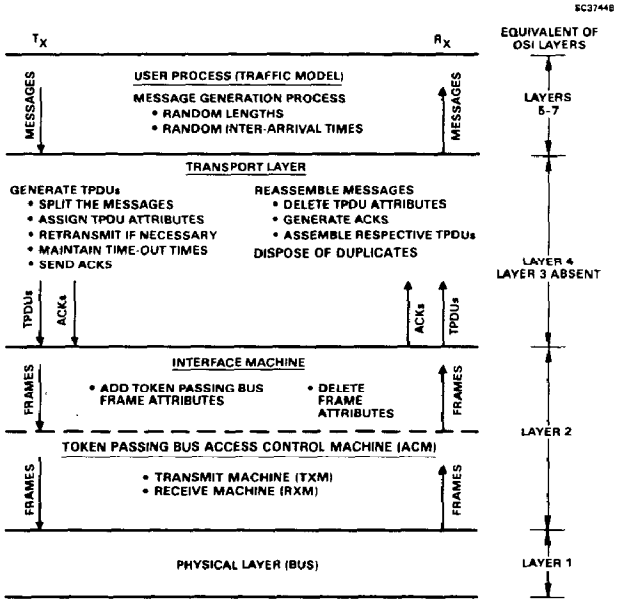

Fig. 4. Overview of the simulation model

A. The Token Passing Bus Mechanism

The IEEE 802.4 Standard token–passing bus scheme [4] is a link level, medium access control procedure for local area networks implemented on a broadcast bus. In this scheme, stations connected to a bus form a 'logical ring' based on the descending order of their addresses (unlike a ring network in which stations would be physically connected serially). A station is allowed to transmit data only after receiving a *token* (control frame), which represents a right to access the bus. After transmitting data, a station hands over the token to the next active station in the logical ring. Thus, the communication on the bus has two phases; i.e., a token–passing phase and a data transfer phase. Detailed procedures have been specified in [4] on the data transfer mechanism and network control.

Each station's protocol is made up of five components: the Access Control Machine (ACM); the Receive Machine (RXM); the Transmit Machine (TXM); the Interface Machine

(IFM); and a station management unit. At each station, there is provision for an optional four-level priority mechanism for traffic handling, with each level having a separate queue. The ability to transmit data frames from each queue is contingent upon the residual value of a *target token rotation timer* (TTRT) for that queue at the time of receiving the token. If the timer expires, transmission of data from that queue is not allowed. After its current value has been transferred to a *token holding timer* (THT), the TTRT is reset with the initial value before beginning transmission of data frames or before passing the token if no data frames are to be transmitted, as the case may be. A guaranteed bandwidth facility is provided by allowing the highest priority queue to transmit for a specified period of time whenever the token arrives.

Timers at different queues may be assigned different initial values depending on the priorities desired. Thus, if a large number of data frames are transmitted from one queue (using up a long time) in a given token rotation, lower priority queues in that station and queues at other stations get relatively less time for transmissions in that rotation, depending on the initial values assigned for the timers of those queues. This interaction among the timers is incorporated in the scheme to enforce fairness in bus access allocations. While considerable flexibility exists in tailoring the priority scheme to the needs of specific applications, implementation of a given priority scheme through selection of appropriate initial values for the timers is complex due to the interdependence among the timers. The complexity grows with increase in the number of stations. Other timing interrelationships determined by such parameters as bus transmission rate, frame sizes, propagation delay, and interface processing delay also become important in determining the achievable throughput [6],[7]. Simulation aids in understanding some of these issues that are difficult to analyze.

B. The ISO Transport Protocol

The transport layer protocol [5] is concerned with providing end-to-end reliable data transfer between two corresponding transport entities. The specific details involving the three lower layers remain transparent to it, as for example, whether it is a packet-switched or a circuit-switched network, whether error control is employed or not at the second layer, and the type of media involved in the communications. However, there are five 'classes' of transport protocols included in the ISO draft proposal on transport protocol specification [5]. For example, class 2 transport protocol assumes the availability of a highly reliable network specified by the lower three layers and hence makes no provision to include automatic repeat request (ARQ) schemes for error control. Class 4 transport protocol does not depend on the degree of reliability offered by the network (irrespective of whether ARQ is used at the second layer at all nodes or not) and makes provision for ARQ error control as part of its procedures.

Our simulation model incorporates several features of a class 4 transport protocol. The transport layer accepts messages from a user layer and fragments the message into one or more smaller blocks. A fixed overhead bit string for control and error checking are added to each block resulting in *Transport Protocol Data Units* (TPDU's). The TPDU's are then passed (through the null layer 3) to the sender's Link protocol layer (Token passing bus) which then converts each TPDU into a *Frame* by adding to each TPDU a bit string corresponding to the protocol overhead of the Link layer. The Link layer then proceeds to transmit each frame on the bus. The Transport protocol layer sends a new TPDU to the Link layer only after getting an acknowledgment TPDU for the previous TPDU from its counterpart Transport protocol layer at the destination. A *window timer* is set whenever a TPDU is passed to the Link layer and will cause a TPDU retransmission if an acknowledgment is not received within the window time. Thus, any 'lost' TPDU's are retransmitted by the sender and any duplicate transmissions are ignored. This is called a stop-and-wait protocol.

C. The Traffic Model

A workload generation model has been developed for describing the message traffic on the NPS network which can produce traffic inputs for the local area network simulation model of the NPS to determine the network performance characteristics. The objectives of the workload module are to specify the communication requirements using the terminology of the NPS environment and to allow for a diverse collection of message types, parameter distributions, and production run scenarios. The traffic model loads the network's stations with the various messages that are transmitted during the course of a typical newspaper production run. The communication traffic is simulated using the concepts of *jobs*, *phases*, and *message scripts*. Figure 5 illustrates this approach. A job is a set of instructions or 'marching orders' assigned to a particular station or group of stations. In newspaper production terminology a job is a production run. Each job has several components, the most crucial being :

(a) A press configuration comprising one or more stations
(b) A press speed (rate)
(c) A number of newspapers to be produced

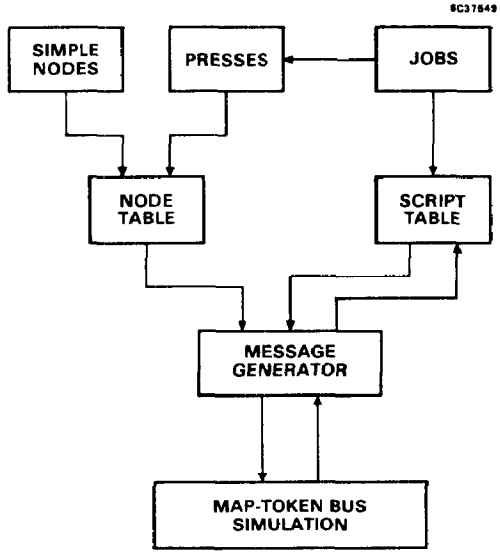

Fig. 5. Workload generation model for the newspaper production system

In addition, a job has three phases associated with it where each phase is a particular state or milestone of the Plant's production run during which certain messages may be transmitted to and from certain stations over the Network. The phases are defined as follows:

Phase 1: MAKE READY - represents the plant start-up procedure and is initiated at the input start time. A zero input value for start time implies that Phase 1 is to commence immediately, while a nonzero input value (in seconds) implies a certain delay period before the plant start-up. Phase 1 executes for a period specified by the input value of the MAKE READY duration.

Phase 2: RUN - represents the normal production run and is initiated upon the expiration of the phase 1 (Start Time + Make Ready) duration. Phase 2 may execute for a duration specified by the input value for the RUN duration.

Phase 3: END RUN - represents the shut down operations and is scheduled upon the expiration of phase 2 (Start Time + Make Ready + Run) duration. Phase 3 executes for a duration specified by an input value for the END RUN duration.

Upon the expiration of Phase 3, the job is complete.

Each phase causes the execution of a corresponding *message script*. The Message Script is a description of all the messages to be generated during a particular Phase. The Message Script details the message contents, the (variable) time interval between successive messages, the (variable) message length, message source station, destination station, message type (i.e., whether the message has been initiated by the source station, or is a response to a previously received message), whether a response message is desired from the destination, the execution time for the message, and finally the total number of such messages in case this message is not generated with constant frequency. Several jobs may be directed by a single script or each job may have its own unique script.

III. DESCRIPTION OF THE SIMULATION MODEL

The following is a description of the SIMSCRIPT programming language model of the simulation of the IEEE 802.4 Token Passing Bus Medium Access, Transport and User layers. The User layer model consists of the Application, Presentation and Session layers lumped together, the Transport protocol layer mimics the MAP Transport protocol layer, and the Link layer consists of a detailed model of the MAP Data Link layer lumped with the Physical layer. The Transport and Data Link layers have been modelled in detail as they have been deemed significant for the network's performance evaluations. Each station on the LAN has associated with it one copy of each of the above mentioned layers. Figure 6 represents the basic workings of these layers and displays the functions provided as messages travel from the source (user) layer, through the transport and medium access protocol layers, and then over to the destination station where they must travel up the layers to the receiving user layer. These functions will be further explained in this section.

A. SIMSCRIPT - A Discrete Event Simulation Language

The model is developed using SIMSCRIPT, a discrete-event simulation language that is comparable in power to Fortran, but with added list processing (queue handling) and discrete-event simulation capabilities. It is English-like in appearance and contains semantics whose world-view assists the modeller in making the correspondence necessary between the elements of the model and the concepts in communications network models. The ability to model the concurrent functioning of the components and the traffic is provided by the software mechanisms in SIMSCRIPT which manage the time clock and switch from process to process without programmer intervention.

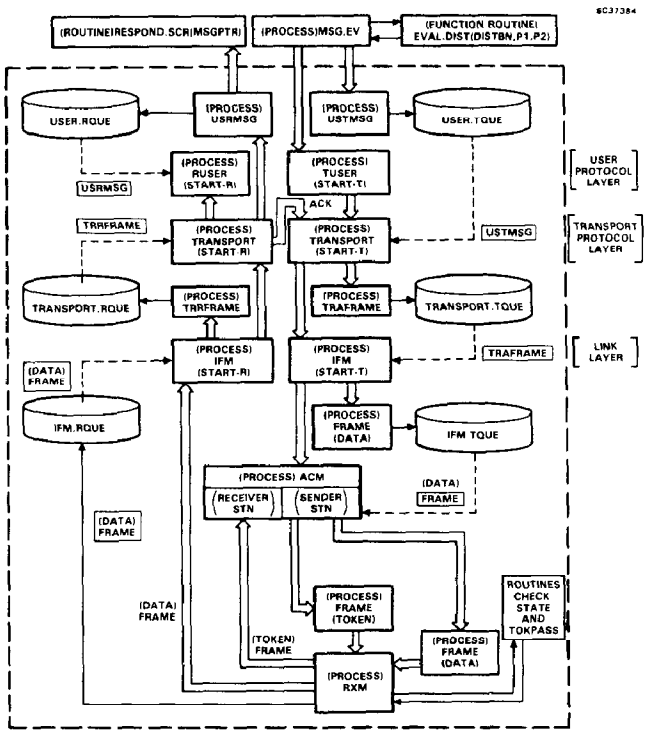

Fig. 6. Link and Transport layer protocols

Various network components (their activities) are modelled as SIMSCRIPT *processes* such as those which control the movement of the frames on the bus. Usually, several replications (instances) of these processes are necessary and the language provides this capability while requiring the programmer to specify only one prototype. Other components are modelled as *resources* for which these processes can contend (eg. buses, buffers). Because the number of some of the problem elements is usually not known *a priori*, dynamic storage allocation and deallocation is provided so that elements of the traffic (messages, TPDU's, frames) can be created and destroyed at will. SIMSCRIPT provides software mechanisms for the collection of statistics as the simulation proceeds over time, which in the study of networks would include performance measures such as throughputs and delays, Node and Link utilization, and maximum and average queue sizes. A complete description of the language is contained in [8].

B. The Link/Medium Access Control Layer

Figure 6 gives a representation of the simulation model for the link and transport modules. The Link/Medium Access Control Layer (MAC) uses token passing bus scheme which is divided into four functional parts:

(1) The Interface Machine (modelled as process IFM)
(2) The Access Control Machine (process ACM),
(3) The Receiver Machine (RXM) and its monitor (RXMON), and
(4) The Transmitter machine (resource TXM).

Each station's Link layer in the model is made up of combinations of the four machines. The replication ability of SIMSCRIPT is used to produce instances of each at each station. In general, when in transmitting mode, the lower protocol layer is responsible for removing from the higher layer's outgoing queues; and when receiving, the lower layer places into the upper layer's incoming queues.

A station has an IFM, which is responsible for withdrawing TPDU's from the corresponding Transport layer queue. The IFM creates a frame from each TPDU by increasing the total length by an extra header length and then inserting these lengthened frames into their respective priority (one of four) queues. Following this, the IFM will notify the ACM that there are frames available.

The intelligence of a station resides in the Access Control Machine (process ACM), which determines when and what to transmit by following the token passing rules. This machine may be thought of as a finite state machine, that is, it may be in any one of ten possible states at any instant. The ten possible states are as follows : IDLE, DEMAND IN, DEMAND DELAY, CLAIM TOKEN, USE TOKEN, AWAIT IFM RESPONSE, CHECK ACCESS CLASS, PASS TOKEN, CHECK TOKEN PASS, AWAIT RESPONSE. The present model has implemented into it four of the ten states, namely IDLE, CLAIM TOKEN, USE TOKEN, PASS TOKEN. The conditions that cause the transition from one state to another are formally presented in the IEEE 802.4 document [4].

The Receiver Machine (process RXM) receives frames (data or token) broadcast on the bus and performs the appropriate frame processing depending upon the frame type, token or data. It also generates a report containing the performance statistics of its bus.

The Transmitter Machine (TXM) is modelled as a SIMSCRIPT resource which is requested and then relinquished by its station according to the station's broadcasting needs. The TXM is held for a time determined by the frame length and the bus transmission rate. A separate process, FRAME, is used to model a frame (data or token); where the appropriate type is parametrically specified. The bus is modelled implicitly in the FRAME process which 'consumes' time as the frame travels on the bus.

The stations on the bus form a logical ring, that is, normally the token is passed from station to station in descending order of logical station addresses. At Initialization, all processes representing the station's components are activated after all their associated resources have been created and exist in the 'idle' state. The TUSER, RUSER, TRANSPORT, the IFM, the ACM, the RXM and TXM associated with each station are defined and logically connected. It should be pointed out that upon completion of the initialization phase all the processes proceed to perform their individual tasks asynchronously with respect to each other. A more detailed discussion of Figure 6 follows.

The IFM:

When the IFM is set to the 'start transmitting' mode, START-T, by its associated TRANSPORT process, process IFM checks the outgoing TDPU queue, TRANSPORT.TQUE and if it is empty, the IFM sets its mode to IDLE and suspends. Otherwise, process IFM removes the first outgoing TPDU, called TRAFRAME, converting it into a data FRAME process with the same characteristics as the TRAFRAME by increasing the information length of the FRAME by a fixed overhead along with some synchronization bits. IFM then places FRAME into its the transmitting queue IFM.TQUE corresponding to the priority specified in the TDPU and notifies the ACM. In this manner, the IFM processes each TPDU in the TRANSPORT.TQUE until it is emptied.

When set to the 'start receiving' mode , START-R, by its associated RXM process, IFM checks its queue IFM.RQUE and removes the first FRAME from IFM.RQUE converts it into an incoming TPDU, TRRFRAME, by removing the frame overhead bits. IFM places TRRFRAME into the receiving queue TRANSPORT.RQUE of its associated TRANSPORT process, and notifies TRANSPORT by setting its status to START-R.

The ACM:

It is the function of the ACM, when it obtains the token, to control the removal of frames from its station's IFM queues and to transmit them onto the bus and then to activate the FRAME process. When an ACM obtains the token it enters the USE TOKEN state where the ACM services (extracts dataframes one at a time from) each of the four IFM queues in the order of their priority: (4,3,2,1). The ACM cannot extract a new frame until the previous one has been transmitted to the destination RXM.

Each of these queues has associated with it a token hold timer variable, THT. This timer is assigned a value representing the residual value of a target token rotation timer, TTRT. Before commencing transmission, the value of TTRT is transferred to THT as soon as the token is received by the station and the TTRT is then reset to the specified initial value. The highest priority queue is always allowed to transmit for a specified period of time independent of the THT. Each queue holds the token (transmits dataframes) until either its THT counts down to 0, or until the queue is empty, whichever comes first. If all four queues of a particular station are completely serviced, it passes the token to its successor station. When a token has visited all the stations and returns to a given station, a *rotation* is said to occur with respect to the given station. The station then copies the contents of TTRT (which was also counting down during the previous rotation) to THT, resets TTRT to the initial value, and proceeds with transmission, as before, commencing a new rotation. When the ACM has completed use of the token it enters the PASS TOKEN state and transmits the token to the RXM of its successor station.

Utilization of the transmitter (TXM) is done during the ACM's data FRAME transmission and when the token is passed to the successor station. The TXM is a resource seized for an amount of time equal to the frame transmission time plus the propagation time of the bus. Upon activation by the ACM, The FRAME process simulates time spent on the bus and then reactivates the destination station receiver RXM.

The RXM:

Upon receiving a frame, the RXM checks the frame's final destination address and if the frame is not at its station destination, then the frame is destroyed and the RXM goes to the idle state. If the frame is addressed to it, then the frame is further checked for its type (data or token). If the frame is a data frame, it is delivered to the IFM incoming queue IFM.TQUE, and the IFM is notified. If the frame is a token frame, the ACM is notified that the token has arrived.

Simulation Speed-up

It was found that removing the token passing events when there was nothing to transmit greatly increases the efficiency of the simulation. After the Token has completed each rotation several conditions are checked and statistics collected. The routine CHECK.STATE is used to identify the 'Idle Rotations', defined as those rotations during which there was no data frame transmission by any station.

By examining the contents of the simulation Event set, specifically for the message arrivals, phase changes, and message responses, CHECK.STATE determines the 'Idle' duration from present simulation time till the next scheduled message transmission event. The magnitude of this duration is divided by the time taken by the token to make an Idle rotation, to compute the the truncated integral value of the number of idle rotations. The simulation program execution is made more efficient by advancing the simulation clock by this corresponding time interval. The statistics are appropriately adjusted which match the simulation results without the speed-up.

C. The Transport Layer

The messages created by the workload traffic model interact with the transmitting user protocol process called TUSER. When the station receives a message, USTMSG, from the workload model, process TUSER places the USTMSG into its outgoing queue, USER.TQUE, and then activates its Transport layer process, TRANSPORT. The TRANSPORT process can be activated by either the arrival of messages from the User layer or upon receiving an acknowledgment TPDU from another station in the IFM's incoming queue (IFM.RQUE). Upon activation, TRANSPORT removes the next USTMSG from USER.TQUE fragments the USTMSG into one or more blocks whose lengths are the maximum length allowable (specifiable by input data), except for the last which may have a variable length. An overhead length is added to these blocks, creating the TPDU's, which are referred to as transmitting TRAFRAME processes. Each TRAFRAME is assigned a sequence number to enable the reordering, and also the various source and destination characteristics of its parent USTMSG. These TRAFRAMES's are then placed into the intermediate, ready to transmit queue, TRANSPORT.PREP. This queue will be used to hold the TRAFRAME's that are blocked by the stop-and-wait protocol, as the IFM will only remove from the outgoing queue, TRANSPORT.TQUE.

In the transmission mode, TRANSPORT gives priority to acknowledgment TPDU's by placing all acknowledgment TRAFRAMES at the front of the TRANSPORT.TQUE, setting the high priority field so that the IFM will place them in its high priority queue. This effectively gives a transport level priority to the acknowledgments. Following this, TRANSPORT will remove one data TRAFRAME from TRANSPORT.PREP and place the TRAFRAME in TRANSPORT.TQUE for service by the IFM. When the IFM removes a data TRAFRAME from the queue, the TRANSPORT places a copy of the TRAFRAME in the retransmission queue, TRANSPORT.RTQ, and initiates the Transport Window Timer. If no acknowledgment frame is received within this duration the TRAFRAME's copy is retransmitted (the copy resides in another queue called TRANSPORT.RTQ). TRANSPORT attempts two retransmissions before abandoning the transmission of this TRAFRAME.

In the receiving mode, START-R, TRANSPORT checks for received frames, TRRFRAMES, by examining the received frame queue, TRANSPORT.RQUE. If TRANSPORT.RQUE is not empty, TRANSPORT removes the first TRRFRAME and checks whether this is an acknowledgment corresponding to a transmitted TRAFRAME. When an acknowledgment is received, the corresponding TRAFRAME copy is removed from the TRANSPORT.RTQ, the Window timer is disabled to suppress retransmission, and the TRANSPORT is enabled to remove the next TRAFRAME from the TRANSPORT.PREP. If the arrived TRRFRAME is a data TRRFRAME, TRANSPORT generates an acknowledgment TRAFRAME which is immediately placed into the TRANSPORT.TQUE. The TRRFRAME is then checked to determine whether it is a duplicate of a previously arrived TRRFRAME stored in the holding buffer, TRANSPORT.RPRQ, and if so, it is discarded. For nonduplicates, the TRRFRAME is placed in TRANSPORT.RPRQ and the message reassembly procedure is started. Attributes such as source station address, 'parent' address, sequence number, and number of 'sibling' TRRFRAMES are all checked against the corresponding identification characteristics of other previously arrived TRRFRAMES. If this TRRFRAME completes a message, then all the TRRFRAMES are removed and reassembled in their proper sequence to form a reconstructed message, called process USRMSG, which is passed on to the User layer receiving process, RUSER, through the USER.RQUE.

D. Workload (Traffic) Model

The workload generation module for creating the message traffic follows the control flow shown in Figure 7. The workload generation program (user module) has several data structures which provide the necessary information for the simulation. The system description consists of the press, the job, and the script definitions. The system is divided into two types of resources: simple nodes (e.g., inventory monitor) and press-component nodes (e.g., press console). Both types of system nodes may depend on the specific job definitions (e.g.. streams, docks, beds, and trucks). A job specifies a particular configuration of resources, defines the phase duration times, and gives the number of copies to be printed. The simple nodes, the press definitions, and the job definitions are used together to compute the set of active stations on the communication network. As shown in Figure 8, the structure of a press contains job information (PJOB.ID, NCOPIES), job phase information (PRPHASE, START.T, MAKEREADY.T, RUN.T, END.T), statistical variables for interval reports, (MSG.CNT, BYT.CNT, MSG.DEL, BYTE.DEL), and pointers to the associated press component nodes. Some of the component nodes are stored as sets to allow the user to specify a variable number of components as with UNIT's, BED's, ARL's, and DOCK's.

Each job will execute the message descriptors defined in the Script Table shown in Figure 9. Each entry of this table consists of a high level description of the message (MNAME, MTYPE, PAGEID), message sources (SNAME) and destinations (DNAME), (e.g., console to units), the phase (PHASE.ID), and the message generation parameters. In addition, a separate response entry in the Script Table, RESP.PTR, is identified and associated with the message. Other parameters of the message specify probability distributions and include the initial message delay (INIT.DELAY), the intergeneration time (IGT), the processing delay (PROC.DELAY), and the message length (LENGTH). These

are specified by choosing a distribution and appropriate parameters. The distributions currently implemented include constant (with parameter .P1), uniform (between .P1 and .P2), and exponential (with mean .P1). The MAXNUM variable gives the number of messages to be generated each time the script is executed. If MAXNUM is zero, the message is repeatedly generated throughout the phase.

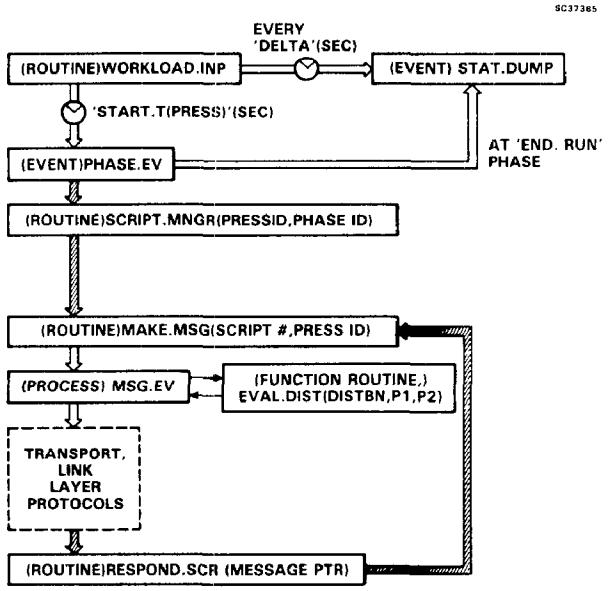

Fig. 7. Flowchart of the workload model

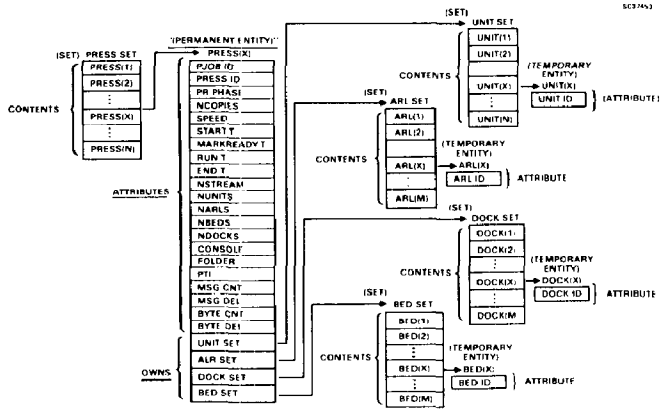

Fig. 8. Data Structures

The workload generation model is integrated with the other simulation modules by interacting only with the user layer protocol routines, TUSER and RUSER. As shown by Figure 7, the workload is initially specified by a user input routine called WORKLOAD.INP which sets up the necessary tables and data structures. The job phase changes for each press, PHASE.EV, drive the simulation events which in turn generate messages. At each phase change, all message descriptors that match the script of the current phase are used to generate the appropriate messages by the script manager, SCRIPT.MNGR. These messages are given to the source station user queues for transmission. When a station has received a message from the user process, the RESPOND.MSG routine examines the script to determine the response and, if a response is required, generates the response message using the script descriptor of the response message, which is then given to the station user process for transmission. Before the response is generated, a user specified processing delay distribution is introduced which represents the time needed by a process to formulate the response message.

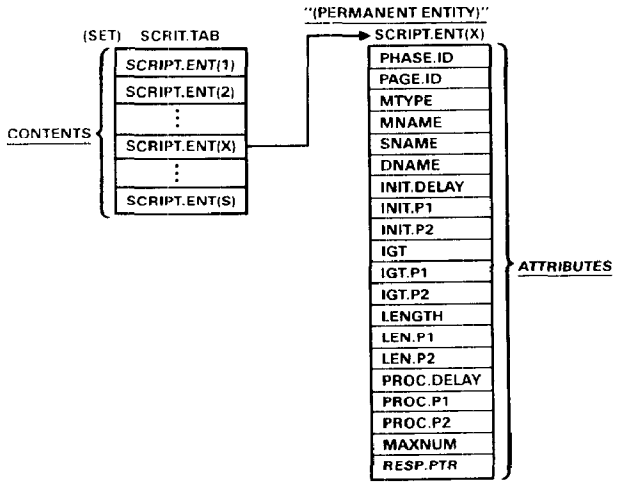

Fig. 9. Data Structures (contd.)

IV. INPUT/OUTPUT PARAMETERS AND SIMULATION RUN RESULTS

In order to allow the modeling of many plant configurations and scenarios, the simulation program inputs supplied by the analyst describe the entire system. There are four classes of inputs: communication hardware, protocol, workload, and simulation control. The communication hardware inputs define the bus properties, such as transmission rate and propagation time. Inputs concerned with protocols involve maximum frame and TPDU length, overhead lengths, and initial TTRT values per station queue. Description of the workload requires definition of the simple nodes and press configurations, the jobs and their phases, and all of the message entries of the script table. Lastly, the simulation control inputs specify the run length, reporting intervals, and the levels of statistical output desired.

The simulation outputs are designed to provide statistics on user-oriented performance measures, protocol-layer options or variables, and LAN operation. Most values are captured per priority queue, per station, per bus and for the overall network. For each protocol layer, an appropriate set of the simulation statistics are collected. The job phase changes also provide statistical dumps at the end of each phase. In addition, the program user has been provided with the capability of setting a DELTA time interval such that it is possible to obtain accounting statistics, such as the number of messages, data bytes, and overhead bytes generated at regular intervals

of DELTA during the entire simulation. It is possible to obtain a lengthy, step by step printed record of each station's transactions at each of its protocol layers, for each message sent by the station, by the setting of appropriate input flags before a simulation run.

A simulation output, called *total message delay* is defined to be the time beginning from the instant when a message leaves the sending station's User protocol layer, until the final instant just before the message enters its destination station's User protocol Layer, and is the primary user performance measure. This is divided into queueing delay and transmission time. Similar measures are provided for TPDU's at the transport level and frames at the data link level. The statistics used to assess the bus performance are the bus utilizations and throughputs per station and per bus in terms of the actual and useful (only data) bytes. In addition, average token rotation time is measured as well as the average residual TTRT values. The number of idle token rotations and the number of consecutive idle rotations are also acquired as additional measures of the idle periods.

Figures 10–12 show the corresponding traffic generated and some LAN performance outputs for a sample production run. This example uses a nine press configuration running a staggered, three job scenario on a single 10 Mb/s bus requiring 159 stations. Figure 10 shows the phase durations for each job and the usage of the nine presses. Through the input data for the workload layer, this type of scenario can be easily specified. The message traffic generated by the specified scripts is plotted in Figure 11 where the highly transient nature of the workload is apparent. It is possible to identify the heavy traffic periods for more detailed simulation analysis using this type of graph. In Figure 12, some of the performance values are plotted for various initial TTRT values. As the low values of data throughput indicate, the loading of the bus is very light, indicating a tremendous amount of spare capacity on the bus. In addition, the statistics are not very sensitive to TTRT values at these light loads. In Figure 12, the active traffic period is a measure of the token rotations that contained at least one busy station. The difference between the throughput and this value is an indication of the token passing overhead, which in this case is substantial due to the large number of stations (159). It is possible to have a very large variance in the token rotation times caused by the arrival pattern of the messages.

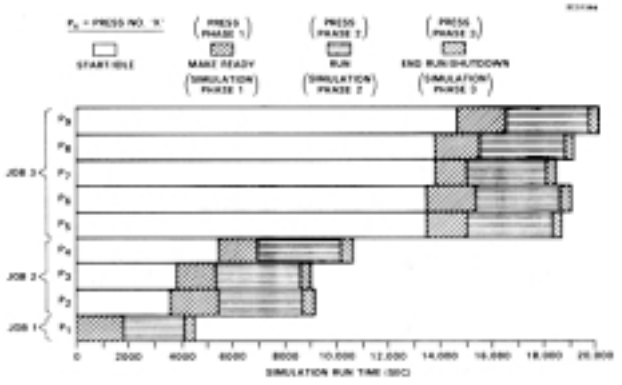

Fig. 10. Job Profiles

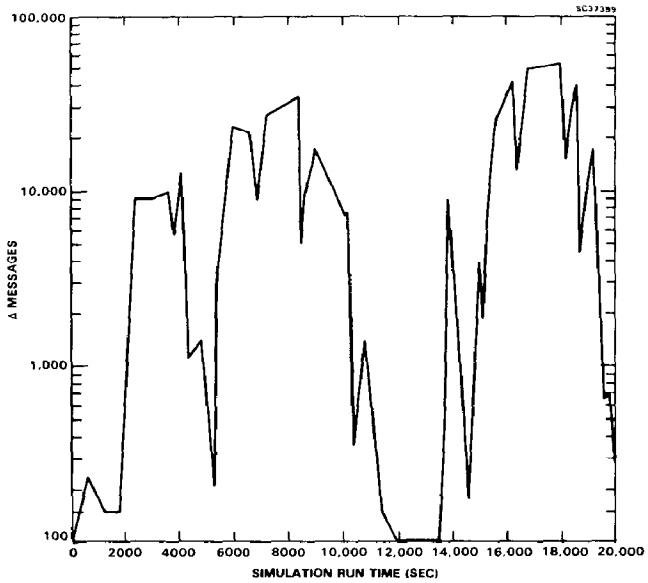

Fig. 11. Full load traffic profile

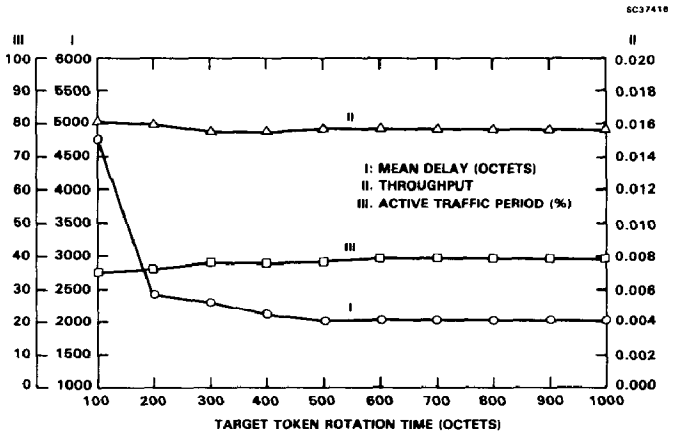

Fig. 12. Simulation results

V. CONCLUDING REMARKS

A simulation program which models the performance of a MAP–based local area network with a realistic workload of a newspaper production system has been developed. The program is flexible in its ability to specify the transient behavior of message generation in terms of the newspaper production environment through an underlying workload model. The characteristics of the MAP environment which are simulated are the transport layer mechanisms and the medium access layer. Statistics which show user, system, and protocol performance are measured and reported.

The simulation model can be used to demonstrate the feasibility of interconnecting the various communicating components of the production systems. This is an important step in eliminating the myriad point-to-point connections typically employed. A production system where all components are connected to a number of common buses would allow for a

high degree of factory integration and optimized control. For most of the communicating components there are real-time command messages that must be delivered within a maximum allowable delay. The current simulation will report on the average delays, but more specific information may be needed to assure correct operation. It is planned to extend the model to measure the number of times a message of a given type experiences a delay greater than a specified value.

Once a system has been implemented, it is possible to use the model to test various strategies for assigning initial values to the target token rotation timers (TTRT's). As the loading of the network increases, the proper selection of timers can yield lower delay values. The timers can be adjusted based on requirements for real-time constraints. Currently, the problem of how to assign timer values and to develop automatic, adaptive timer assignment algorithms is under investigation using the simulation model.

REFERENCES

[1]. *Manufacturing Automation Protocol (MAP)*, Version 2.2, General Motors, August 1986.

[2]. International Standards Organization (ISO), "Information Processing Systems - Open Systems Interconnection - Basic Reference Model," Draft Standard ISO/DIS 7498, 1982.

[3]. Proceedings of the IEEE, (Sp. Issue on Open System Interconnection model), December 1983.

[4]. IEEE Project 802, Local Area Network Standards, "Token-passing Bus Access Method and Physical Layer Specifications," IEEE Standard 802.4, 1985.

[5]. International Organization for Standardization (ISO), "Transport Protocol Specification," ISO/TC97/SC16, Draft Proposal ISO/OP 8073, June 1982.

[6]. A.R.K. Sastry, "Maximum Mean Data Rate in a Local Area Network with a Specified Maximum Source Message Load," Conf. Rec., IEEE INFOCOM, pp. 216–221, March 1985.

[7]. A.R.K. Sastry and M.W. Atkinson, 'Simulation of the IEEE 802.4 Token Passing Bus Protocol Using SIMSCRIPT,' NBS Workshop on Analytic and Simulation Modeling of IEEE 802.4 Token Bus, Gaithersburg, MD, April 1985.

[8]. SIMSCRIPT II.5 *Programming Language*, CACI Inc., 1983.