Modelling High Data Rate Communication Network Access Protocol *

S. Khanna, E. C. Foudriat, F.Paterra,
K. Maly, C.M. Overstreet
Old Dominion University
Norfolk, VA 23529

April 23, 1990
Draft

1 Introduction

Modelling of high data rate communication systems is different from the low data rate systems. Unlike the low data rate systems a message does not fill the whole network structure, so there can be many messages on the system at one time. It implies that more than one event may take place at a time and it is impossible to model the network by treating messages as entities which start and end before other events take place. Previous experience with simulations of networks where only a single message was considered indicated that protocol models are complex and simulations usually take a long time to run on the computer.

Three simulations were built during the development phase of CSMA/RN modelling. The first was a model using Simscript was based upon the determination and processing of each event at each node. The second simulation was developed in C based upon isolating the distinct object that can be identified as the ring, the message, the node, and the set of critical events. The

*This work was supported in part by CIT under grant INF-89-002-01. by NASA under grant NAG-1-908, and by Sun Microsystems under RF596043

third model further identified the basic network functionality by creating a single object, the node which includes the set of critical events which occur at the node. The ring structure is implicit in the node structure. This model was also built in C.

In this paper, we will discuss each model and compare their features. It should be stated that the language used was mainly selected by the model developer because of his past familiarity. Further the models were not built with the intent to compare either structure or language but because the complexity of the problem and initial results contained obvious errors, so alternative models were built to isolate, determine and correct programming and modeling errors. The next section discusses the CSMA/RN protocol in sufficient detail to understand modelling complexities. In the following sections, each model is described along with its features and problems. Following this the models are compared and concluding observations and remarks presented.

2 Description of CSMA/RN protocol operations

The network access controller for CSMA/RN is shown in Figure 1. The incoming signal is split into two streams, one through a delay line or buffer. The node controller, based upon information accumulated in the buffer, is required to make a number of decisions. First, it must detect the presence of incoming data; if it exists, the node must always propagate incoming information as the outgoing signal to the next node on the ring because it would be impossible to recreate the packet unless sufficient storage is provided. If no incoming packet exists, the node is free to place its own data on the ring if its queue is not empty. However, during the time this latter data is being transmitted, if an incoming packet arrives, then the node, within the time limits dictated by its buffer size, must discontinue its transmission and handle the incoming packet. Hence, packets once on the ring take precedence over the insertion of new packets.

Packets are tested at each node to determine if the incoming packet is destined for this node and should be copied to its incoming data buffer (not shown in Figure 1). In addition, to improve the network operation, packets are removed at the destination so the node can use the free space to send

information waiting in its queue. Destination removal improves performance under uniform loading by a factor of two [].

Figure 2 illustrates the events that can occur at each node based upon the travel of empty and full packets of data around the ring as time progresses.

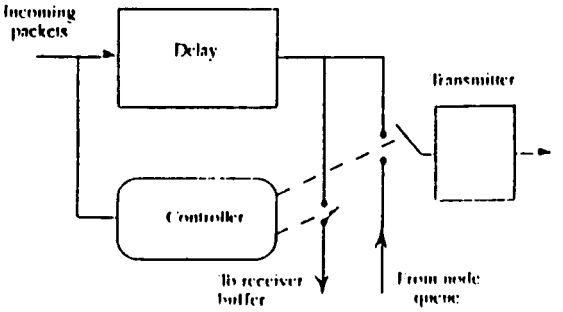

Figure 1 CSMA/RN Access Controller Logic

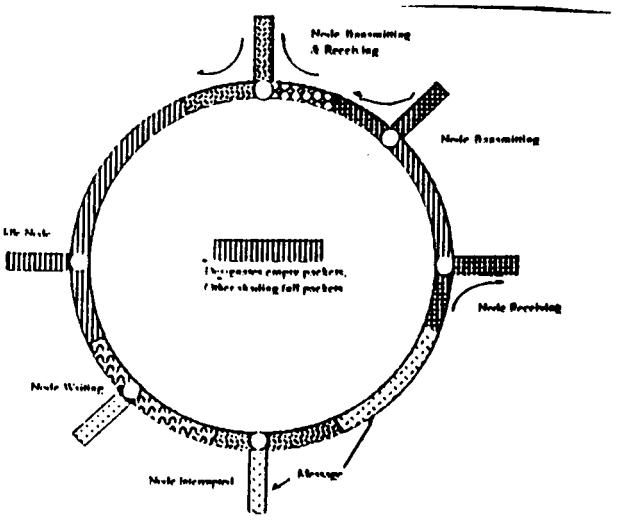

Figure 2. Illustration of Network Operational Conditions

3 Simulator I - Simcript - Node Event List

3.1 Model structure

The first model attempted for the simulation of CSMA/RN was a SIMSCRIPT based model consisting of approximately 900 lines of SIMSCRIPT code. The simulation was written to be event driven, rather than process driven, and has four event types. Because of memory constraints, outgoing messages are only generated when there exists and opportunity to transmit. because of this, all events are initiated by external traffic passing in front of the node. The events are defined as follows:

- start.of.message given node.id, destination, and message.id – executed when a new message is seen passing in front of a node. This event

terminates any internally generated message that is currently being transmitted and places it in a queue for retransmission, sets a busy flag. If the message seen is not for the current node, a **start.of.message** event of posted for the next node in the ring.

- **end.of.message given node.id, destination, length, and message.id** – This event is executed when the end of a messages passes in front of the node. The busy flag set in **start.of.message** is reset and statistics are updated. As with **start.of.message**, an **end.of.message** event is posted for the next node in the ring. After this event completes execution, the current nodes has an opportunity to transmit any data pending.

- **interrupt.message given node.id, destination, length, and message.id** – This event is similar to the **end.of.message** event, but is initiated when a message is interrupted during transmission. As this event is propagated through the nodes, the processing is the same as that of the **end.of.message** event.

- **check.flag given node.id** – When a locally generated message begins transmission at a node, a flag is set to indicated that the node is currently transmitting. This flag inhibits the node from trying to transmit more than one message concurrently, but it must be reset when the message completes transmission. To reset this flag at the proper time, the event **check.flag** is scheduled to execute when the message terminateds transmission. This event resets the flag and, if the message was not interrupted, posts an **end.of.message** event for the next node in the ring.

In the above events, the parameters are defined as follows:

- **node.id** – This is the number of the node that must execute the event.

- **destination** – This is the destination of the message.

- **length** – This is the length of the message, or partial message in the case of an interruption that was sent.

- **message.id** – This parameter is used to insure that all messages are processed in order.

4 Three Object Model

4.1 Model Structure

In this model of CSMA/RN three distinct structures, the ring, the node and the event list, were manipulated by code related to each structure. While they were not identified as objects per se, as in C++, they were separate code blocks in the later versions of the program. In the following sections, we will describe each unit.

4.1.1 Ring Structure

The ring is defined by the number of bits that can exist simultaneously. For example, a 10 km length ring with a 1 Gbps data rate assuming media speed of 5 microsec/km contains 50,000 bits. The ring is modeled as a doubly linked list of data structures containing the necessary data for a packet (a packet is a contiguous portion of a total message). The packet data includes the bit position of its begin and end location on the ring, its length in bits, the packet condition (e.g., free, in use, etc.), message information including source, destination and message number, and left and right pointers to the packets on the ring. Ring operations include updating the packet locations by the increment of time for the next event and linking,delinking,creating and combining packets.

4.1.2 Node Structure

The nodes were defined to be at fixed bit locations on the ring. The nodes were modeled as an array of structures with each succeeding node at a higher bit location. The node data structure contained considerable information including:

1. node number, location and distance to previous node;
2. message details e.g. destination, length, timing data;
3. operational statistics on network performance; and
4. a pointer to the packet presently at the nodes location.

The procedures relating to node operations include collection of operational information, new message generation, updating the packet pointer as packets progressed around the ring and handling the events which occurr at the node.

4.1.3 Event Structure

The event structure and its operations were fairly standard. The event list was a doubly linked list with an event type, pointers to the node and/or packet related to that event, and pointers to complete the list. Procedures related to event processing consisted of creating the event and linking it into the event list either from the head or tail.

The main program is an event handler. The next event is read, the ring and nodes updated as needed, and the event processed. The event can change the ring and/or node conditions. After the event is handled, those nodes which are ready but do not have free packets assigned for their ready message are processed so that if a free packet is available then it can be assigned.

4.2 Experiences

The major difficulties in developing the ring system were in programming the correct wrap around conditions for the ring position and for the various tests relating node locations to packets on the ring. The combining of empty packets is necessary to reduce the number of events as empty spaces are filled more readily. For updating packet pointers, it was found that it had to be checked each time packets were created, removed or moved, since any of these conditions could change the node to which packet pointed.

Table 1 shows the breakdown of subroutine code for each major structure. The general code includes I/O, initialization routines and generic procedures.

Item/Code Unit	Ring	Node	Event	General
Procedure Count	5	7	3	5
Lines of Code	100	332	121	252

Table 1. Procedure Code to Support Network Simulation System

The major problem of coding and debugging was the identification of the event interface between the packets on the ring and the nodes. Initially, two

events were described, the insertion of a new packet at a ready node and the removal of a packet which had reached its destination. In order, to schedule packet insertion, the node, when ready, looks at the packet at its location. If it is empty, a new packet is created starting at the present location; if the packet at the location is full, the node searches arriving packet to identify an empty packet. If an incoming empty packet is found, an event is created for its arrival time at the node. Arriving full packets are emptied and the node checked to see if it has a ready message.

The major event handling difficulties arise due to potential interaction of these events with adjacent up stream nodes. First, if an arriving full packet length encompasses the previous node(s) the packet can not be completely emptied or else the previous nodes may incorrectly identify the packet as empty and use it. Thus, the packet must be truncated at the previous node and a pseudo arrive event created to take care of new event. For very long packets a number of subsequent pseudo events may be necessary.

Very similar problems occur for filling empty packets and searching for empty packets for a node to use. Packets can not be created for a length greater than the previous node since that node may become ready and occupy a part of the empty packet. Alternatively, a node can not identify for use an empty up stream packet which a prior node may use before the packet arrives at the node in question. This creates handling problems which make the originally simple events quite complex and account for much of the code and most of the programming problems.

5 One Object Model

The model is based on the fact that the carrier sensing is local to a node and the nodes implicitly refer to the ring. The main events which result from the local carrier sensing are :

- A node receives an upstream message not destined for itself and gets interrupted and starts reposting the upstream message.
- A node is transmitting a message which is on the head of its queue.
- A node is idle i.e. node has an empty queue.

The unit of measurement in the model is a bit.

5.1 Node Structure

The ring is modelled as an array of nodes and broadly speaking each node has the following structure -

- The Node Status: Transmitting, Idle or Reposting;
- Head of the Queue: Information about the message which is on the head of the imaginary queue at a node. For example, arrival time of the message, message length, its destination etc. In this model a queue never exists but a new arrival is scheduled whenever a message is fully transmitted which may be well in future or way back in past;
- The Interrupt Timetable: A linked list of interruption times for a node and the duration of interruption. It lists the time a node will pre-empt the current message if it is transmitting (and resume later) and start reposting for the duration mentioned in the list;
- Statistics: Network performance satistics.

5.2 Interaction

The operation of the model is mainly goverened by a schedular,Next Time Generator, which looks at local conditions at all the nodes and then decides the next time when a set of events will fire at different nodes. The resulting flow is depicted in the figure 3.

5.3 Experiences

Two or more events may occur simulatneously on the ring. The schedular has a tight job to decide about the next time it will return. Also, the interrupt timetable has to be maintained carefully enough else the packets on ring will start cramping on each other.

The ring under heavy load may have a state where all the messages can be in a bumper to bumper situation. If a node finds this state of the ring in its interrupt timetable, it will continue to repost till it finds a hole on the ring.

For packets of sizes equal to the bitlength between two consecutive nodes, a proxy-completion of the message is caused which is identical to $buf = 0$ state in the flow given in fig.3. The proxy-completion avoids a phenomenon where each node checks for its transmit complete and next node's interruption before interrupting and finishing transmission.

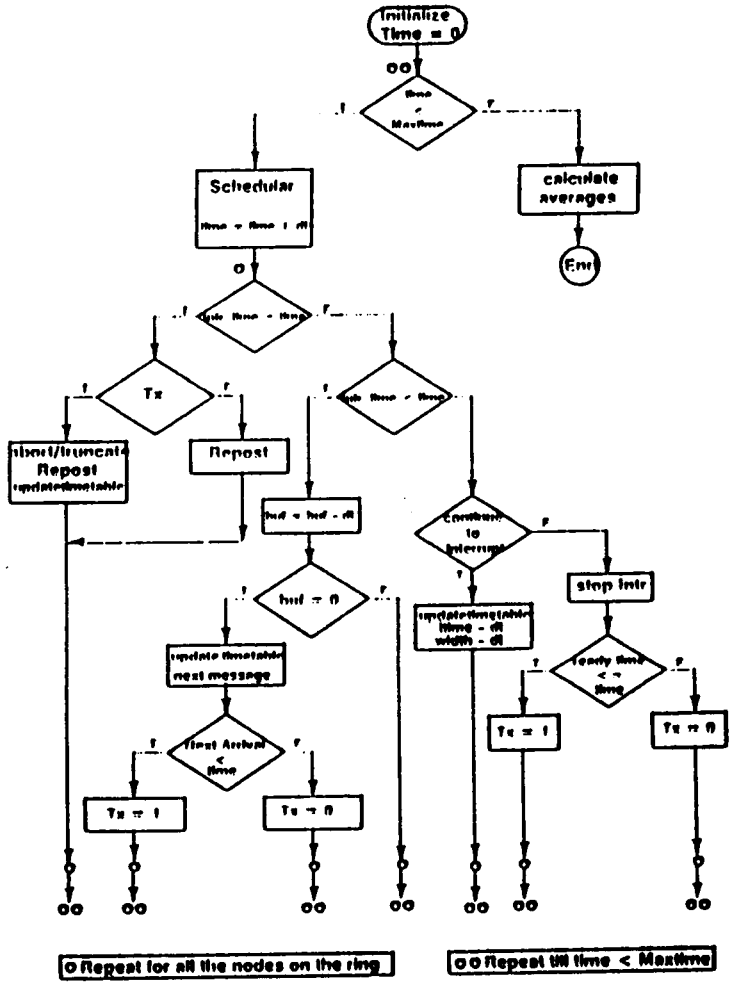

Fig.3 Flowchart for one object model

6 Discussion

Decisions are based on the local conditions at a point in the network and not on global conditions. Even if the decisions are simple, complexity may occur because prior decisions when propagate, influence the present network condition at some other point in the network. Conditions on the network evolve. Each bit or indivisible block of network should be modlled so that it and its effects on the network conditions can be developed.

Various types of runs were made to study simulator confidence. Data were collected for intervals during a run and compared as to their variability and to the mean of all data collected for the run. We plotted wait time, the most sensitive of the variables, taken at the end 10 intervals, and the cumulative average taken of the active period of the run. Load fractions of 1.0 and 1.5 were used since at the higher loads, fluctuations tend to be greater. First, it was found that the ring tended to reach steady state values rather quickly, but its results still varied considerably between intervals. It was found that in order to obtain data with a 90% confidence in the mean accuracy, the ring had to cycle a number of times, where a cycle is the time for information to completely traverse the ring. In general, about 1000 - 5000 cycles was found to be sufficient elasped time. Figure 4 shows a comparitive result for wait time analysis for a 10 nodes, 10 Km, 1Gbps, and 2 Kbits messages.

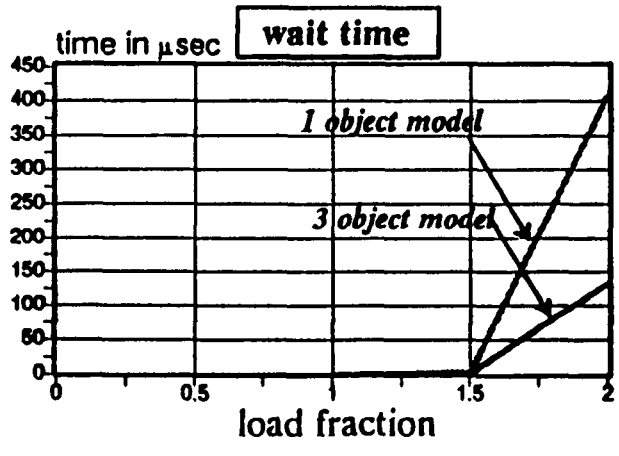

Fig.4 Comparitive results for wait times

7 Concluding remarks

To date, CSMA/RN studies have been limited to simple asynchronous data operational conditions. Additional study is required to document its performance for messages with variable lengths, for non-uniform load conditions, for conditions where ring domination by a few nodes can occur, and for large node count conditions where message fracture is most likely. Protocol procedures must be developed and studies must be done for CSMA/RN to effectively handle integrated traffic, i.e., synchronous traffic consisting of voice and video data in conjunction with asynchronous messages. It means that the model's capability to handle complex decisions needs expansion as operational features of the protocol become known; thus adding capability to the model further increases its complexity.