A CONCURRENT SIMULATION OF ETHERNET

by

CHARLES M. MEDVITZ

B.S. San Diego State University, 1974

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

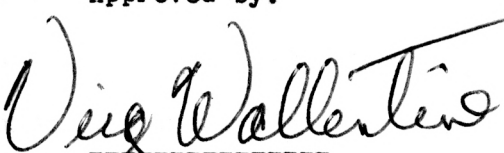
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

Approved by:



Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

This project was initiated to produce a base line simulation of an Ethernet network. This will be used to facilitate the analysis of local network efficiency through comparisons with other broadcast media protocols, and to provide a foundation for further development of Ethernet simulations.

1.2 Objective

The objective was to develop a simulation of the broadcast media of an Ethernet computer network and validate this simulation based on the experimentation results of Shoch and Hupp [1] and the analytical model of Metcalfe and Boggs [2].

1.3 Summary

The simulation model was derived from the Ethernet Data Link Layer and Physical Layer Specifications [3] and executed in a concurrent environment through the use of SIMMON, a concurrent PASCAL based simulation system [4]. This project was limited to physical transmission and Data Link Layer transmission management in a network state of all stations on the network attempting to produce a 100 percent network load. Physical Layer reception and Data Link Layer packet formatting and acknowledgement processing were not incorporated into this model in order to produce a

network state as close to the analytical model of Metcalfe and Boggs as possible. Simulations were conducted for a range of message (packet) sizes varying the number of stations in the network. Results were compared to the finding of Shoch and Hupp and the analytical model of Metcalfe and Boggs. These comparisons indicate the goal of producing a validatable Ethernet simulation was accomplished. This provides a useable base simulation with validated Physical Layer and Data Link Layer protocol implementation for development of more sophisticated Ethernet models.

1.4 Report Structure

This report contains, in order, an Ethernet description, project validation standards, explanation of simulation development, analysis of validation standards, analysis of simulation performance, and conclusions developed from analysis of project results. The appendices contain a copy of the simulation station program, master timer program, and user guide, which are followed by a list of references cited.

CHAPTER 2

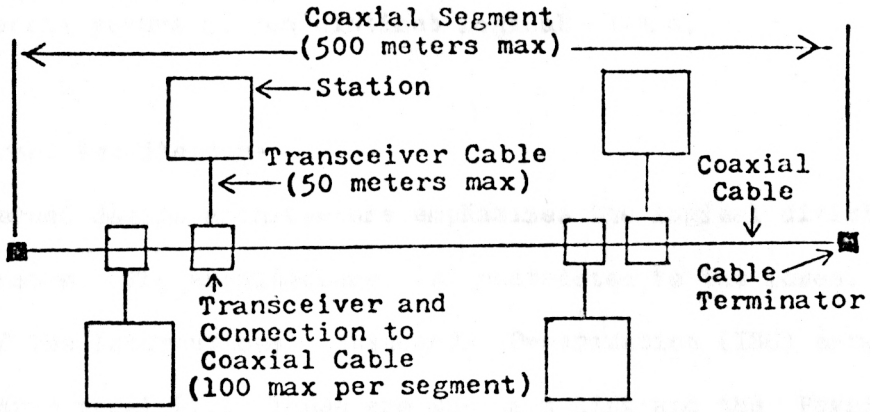
ETHERNET DESCRIPTION

2.1 Ethernet General Characteristics

Ethernet is a distributed computer network control system with the communications media (Ether) shared by all stations on the network. The network can be viewed as "either a local computer network or loosely coupled multi-processors" [5]. A local area network with a maximum station separation of 2.5 kilometers, connecting no more than 1024 stations through a shielded coaxial cable with base band signaling. Ethernet provides for variable size packet transmissions with "best-effort" delivery [6]. The network is an unrooted tree with only one path between any two stations. A network is configured using segments of coaxial cable, which are terminated at each end, as a basic unit. These segments are connected through repeaters providing the ability to tailor a network to a user's needs and allows for expansion (Figure 1). Each station operates with a Carrier Sense Multiple Access (CSMA) protocol with Collision Detection (CD).

CSMA allows each station on the network to sense the Ether and transmit only after it has sensed the absence of a carrier signal on the Ether. CD enables the network stations to recognize a collision (more than one station attempting to transmit at the same time), transmit a "jam" signal to ensure all stations involved in the collision recognize this fact, and initiate a dynamic delaying mechanism (an exponential backoff algorithm) to

A. Minimal Configuration



B. Typical Large Scale Configuration

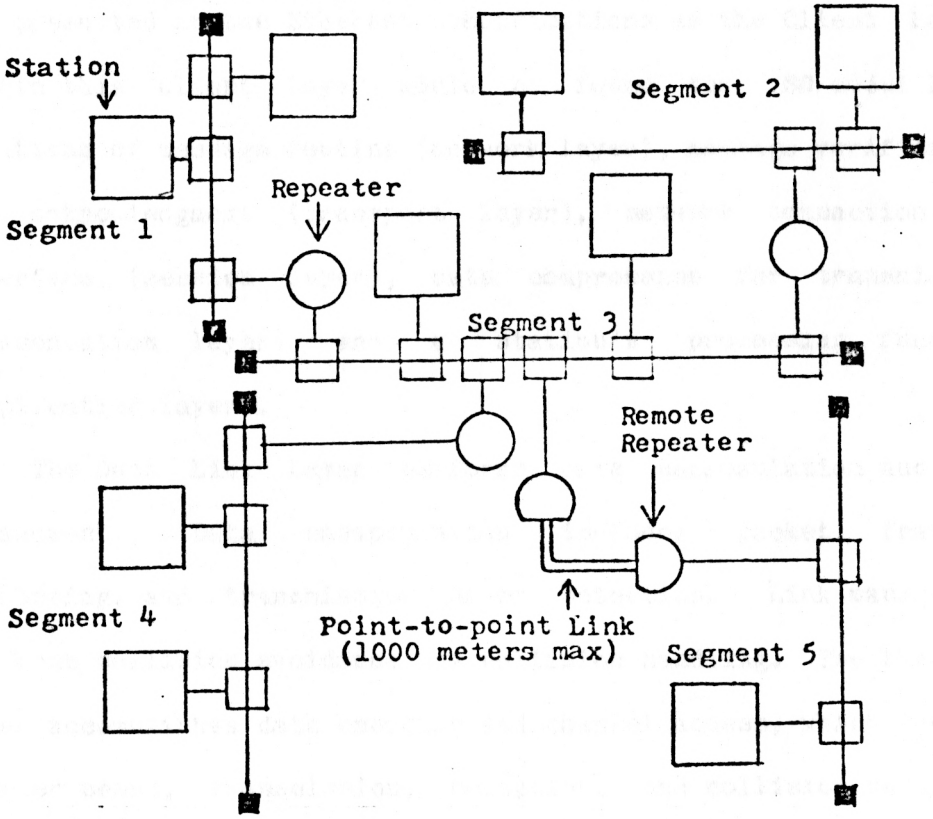


Figure 1

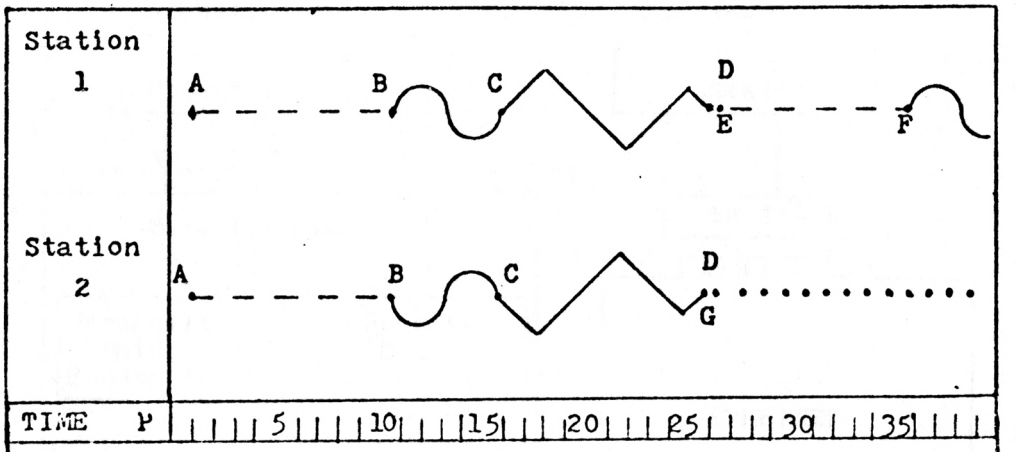
Ethernet Configurations

determine an amount of time to delay before retransmission is attempted (Figure 2). These measures are incorporated into the network architecture at two distinct logical levels.

2.2 Ethernet Architecture

Ethernet design architecture emphasizes the logical divisions of the system. The architecture is restricted to the lowest two layers of the International Standards Organization (ISO) network architecture model [7]. These are the Data Link and the Physical layers. The ISO model layers above this are left untouched, and are presented in the Ethernet specifications as the Client Layer. Within this Client Layer would be found the ISO model layer functions of message routing (network layer), message verification and acknowledgment (transport layer), network connection and interface (session layer), data compression for transmission (presentation layer) and a station's processing function (application layer).

The Data Link Layer performs data encapsulation and link management. Data encapsulation includes packet framing, addressing, and transmission error detection. Link management performs collision avoidance and collision handling. The Physical Layer accomplishes data encoding and channel access, which include carrier sense, transmission, reception, and collision detection functions. The functions of the Data Link and Physical layers and their correspondence to typical implementation hardware are presented in Figure 3. This architecture and its implementation provides distributed network control with each station



Point in Time

A. Stations 1 and 2 sense the Ether clear and each station defers 9.6 microseconds before transmitting.

B. Both stations begin transmitting and monitoring the Ether to insure validity of their signal on the Ether.

C. Stations receive transmissions of both signals, recognize a collision of their transmission with that of another station and begin to "jam" the Ether, in this case for 10 microseconds.

D. Dynamic delaying mechanisms of each station determines time to delay before attempting retransmissions.
Station 1 determines not to delay.
Station 2 determines to delay.

E. Station 1 senses Ether clear and defers.

F. Station 1 begins retransmission.

G. Station 2 delays.

Figure 2

Signal Collision Handling

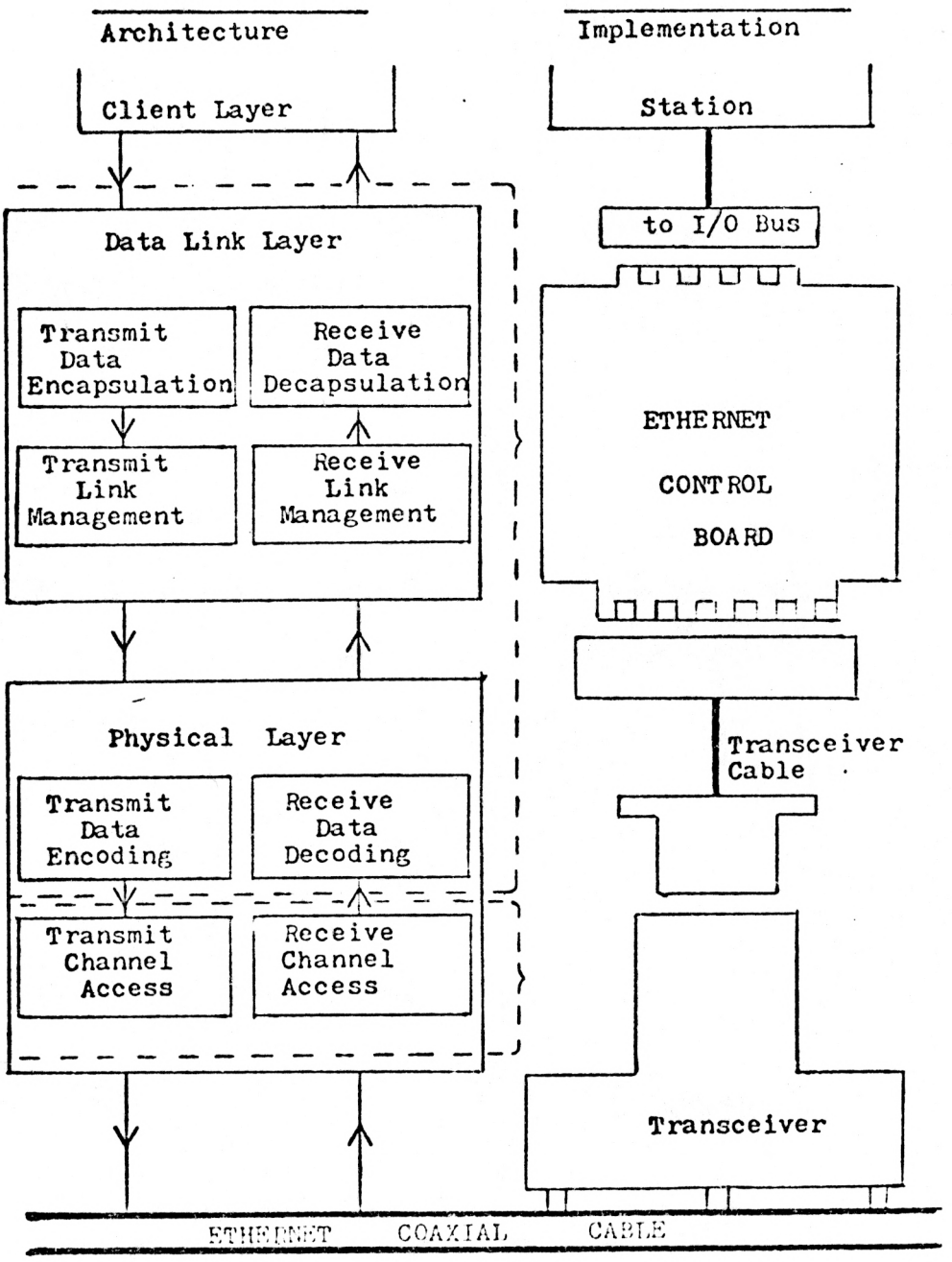


Figure 3

Ethernet Architecture

independently accomplishing message transmission scheduling and routing.

CHAPTER 3

SIMULATION VALIDATION STANDARDS

3.1 Analytical Model

The work of Metcalfe and Boggs presented in "Ethernet: Distributed Packet Switching for Local Computer Networks" [8] provides a firm base for analysis of Ethernet performance. The analytical model they offer is keyed to a collision control algorithm which increases the mean wait time after a collision before attempting a retransmission, based on the network traffic load (Figure 4). This control mechanism approximates the Binary Exponential Backoff Algorithm defined by Metcalfe and Boggs providing adaptive scheduling of retransmission attempts.

To present the analytical model the following are defined:

P- The number of bits in a packet

C- The peak Ether capacity in bits per second (bps)

T- Time in seconds for a slot (the time needed to detect a collision)

Q- The number of stations on the network.

It is assumed that all Q stations are continuously prepared to transmit a packet.

The probability that a station will attempt to transmit when the Ether is sensed clear is 1/Q. The probability that a station will delay is 1-(1/Q). "This is known to be the optimum statistical decision rule" and is approximated by the collision control algorithm [9]. The probability that only one station

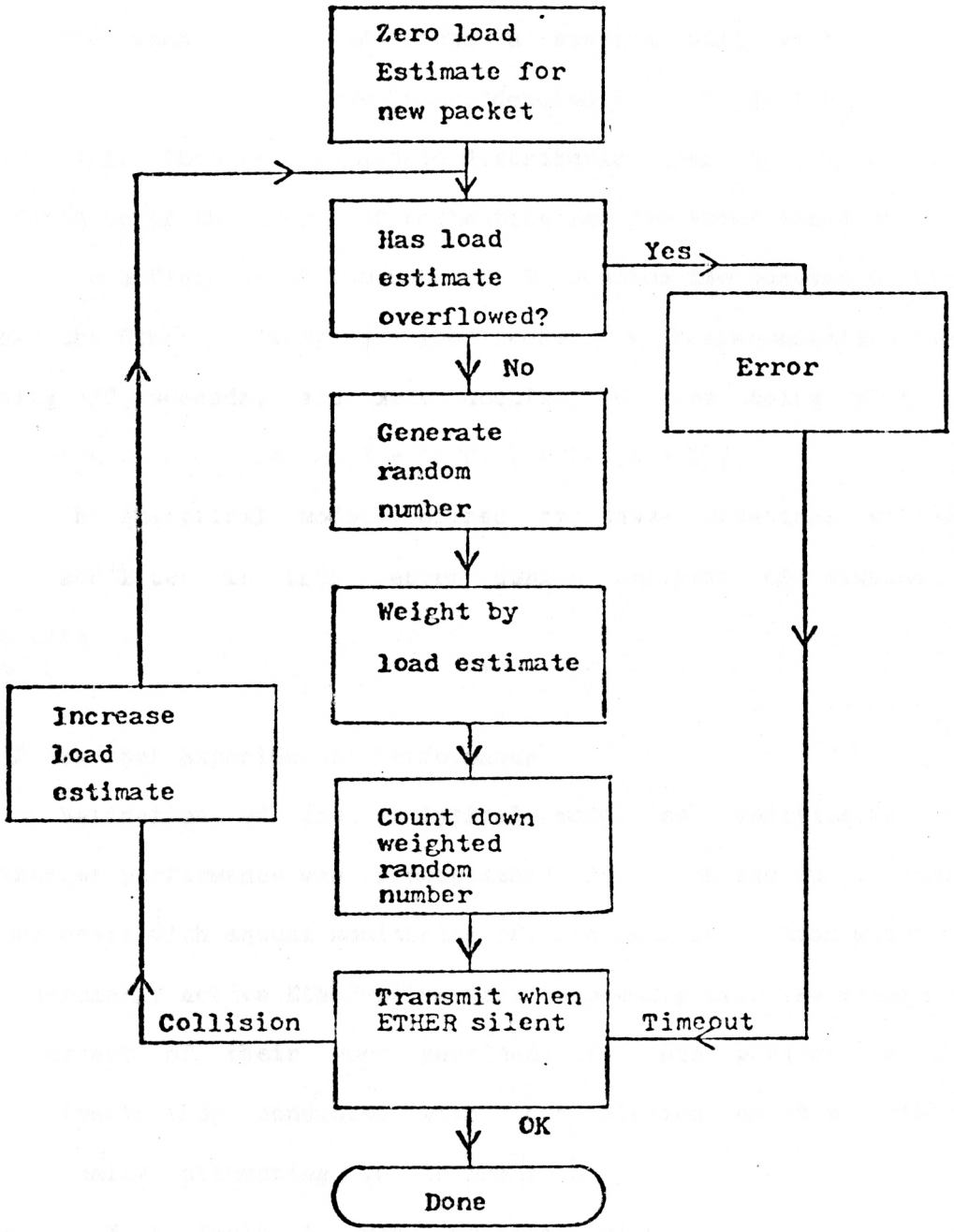


Figure 4

Collision Control Algorithm

attempts to transmit when the Ether is sensed clear, and thus acquires the Ether, is denoted by A. Where $A = (1 - (1/Q))^{(Q-1)}$.

The mean number of slots a station will wait before successfully acquiring the Ether, denoted by W, is defined as $W = (1 - A)/A$. This is a geometric distribution where the mean, W, is a function of the amount of contention for the Ether based on Q.

The efficiency of the network, E, denotes the percent of time that the Ether is carrying a good packet. With transmission time being P/C seconds, and mean acquisition time being W * T, efficiency is defined as $E = (P/C)/((P/C) (W * T))$.

The analytical model defined by these equations will be utilized later in this report during analysis of simulation results.

3.2 Ethernet Experimental Performance

Validation of the analytical model and verification of Ethernet performance was accomplished by Shoch and Hupp. Their work dealt with actual monitoring of, and experimentation with, a commercially active Ethernet network connecting over 120 machines. The aspect of their work pertinent to this project is the experiments they conducted with each station on the network continually attempting to transmit a message. The results, presented in Table 1, were obtained with an Ether capable of carrying 2.94 Megabits per second (Mbps), and were adopted as a validation standard for this project based on Shoch and Hupp's statement that they had validated the analytical model with their experiments. Attempts were made to contact Mr. Shoch in order to

obtain more data concerning the experiments he and Mr. Hupp conducted, but these efforts were unsuccessful until after completion of all phases of this project except preparation of the final report.

STATIONS (Q)	PACKET LENGTH (P) IN BITS				
	4096	1024	512	48	32
5	97%	95%	94%	72%	-
10	97%	91%	89%	68%	58%
32	97%	90%	83%	64%	56%
64	97%	92%	85%	61%	54%

NOTE: This data was originally reported in bytes and converted to bits on an 1 to 8 basis to provide commonality in message size units.

Table 1 [10]

Shoch and Hupp Experimental Results

CHAPTER 4

SIMULATION DEVELOPMENT

4.1 Selection of Simulation Technique

The key issue in developing an Ethernet simulation is the reproduction of the simultaneous operation of all stations on the network, and the resulting contention on the Ether when more than one station attempts to transmit at the same time. While this environment could be reproduced through several simulation languages available at Kansas State University computing facilities, the basic concurrent nature of SIMMON made it the primary candidate for this project. This position was further reinforced by the fact that the Data Link Layer Procedural Model provided in the Ethernet Specifications is presented in PASCAL, allowing for an almost direct implementation of this procedural model.

SIMMON is a concurrent PASCAL simulation system developed to model computer system operations. Sequential PASCAL programs are written and compiled separately, then executed as concurrent processes synchronized by the SIMMON simulation monitor. This produced an extremely natural environment for preparing the simulation, based on a PASCAL procedural model, and executing the simulation in a concurrent environment.

4.2 Model Development

Having selected PASCAL as the language to be used for simulation implementation and SIMMON as the supporting simulation system, model development flowed naturally from the Ethernet Specifications. Data Link Layer procedural model procedures for Transmission Link Management, Watch For Collision, and Exponential Backoff were transcribed as directly as possible. Adjustments were made to allow for time passage and linkage to procedures simulating the Physical Layer.

The lack of a random number generating function in the specifications did present some difficulties. Two distinct pseudo random number generators were used during initial simulation testing to determine their impact. It was found that efficiency could be affected by up to 15%, for small packets, by the random number generator used. The random number generator selected for incorporation into the simulation model was extracted from Grogono's "Programming in Pascal" [11]. after analyzing it for pseudo randomness based on the considerations presented by Knuth in the "Art of Programming" [12]. This generator was placed in the individual sequential processes with the SIMMON assigned process number used as a seed value.

The Physical Layer transmission and "jam" functions were incorporated in as direct a manner as possible. The transmission media (Ether) was implemented as a variable to which a station would add a discrete value when it transmitted. This variable was checked by the transmitting process after the passage of the mean contention interval and after the passage of each subsequent time

unit (equivalent to 1 microsecond) throughout the duration of the Extended Contention Interval of a transmission (the mean contention interval and Extended Contention Interval are discussed in Sections 5.1 and 5.2). If the value of the variable was not the same as the "transmitted" value, a collision was identified and the transmitting station then "jammed" the media by incrementing the Ether variable by a value greater than the value of any station message value. Transmission duration was established by the packet size, based on the Ether capacity in bps (i.e. a 32 bit packet transmitted on 3 Mbps Ether would have a duration of 10.6 microseconds simulated by the passage of 11 time units).

4.3 Model Characteristics

4.3.1 Network System Characteristics

The system characteristic incorporated into the model were:

1. Line Capacity of 3 Mbps. This was chosen to approximate the system used for experimentation by Shoch & Hupp.

2. Slot Time of 512 bit times. Worst case slot time recommended by Ethernet Specifications to allow for round trip message propagation, synchronization time for intervening electronics, signal rise time degradation, and maximum "jam" time [13].

3. "Jam" Signal Duration of 32 bit times, the minimum jam duration per specification [14].

4. Interframe spacing (Deference Interval) of 9.6 microseconds [15].

5. Maximum attempts to transmit the same message set at 16 [16].

6. Limit on number of times to backoff 10 [17].

A copy of the sequential PASCAL program representing a station in the model is at Appendix A. It should be noted that timing increments were resolved to whole microsecond intervals, and that this impacts on the resolution of simulation results. This was done primarily to reduce simulation execution time.

4.3.2 Steady State Initialization Mechanism

After development and testing of the simulation model a subroutine was added to initialize the stations to a steady state condition. Analysis of the Ethernet collision control mechanism indicated that this was attained when the attempts to transmit a message were uniformly distributed from 0 to 15 among the stations on the network. A uniform random distribution of initial preset attempt values was achieved using the pseudo random number generator and uniform distribution function contained in each station model.

CHAPTER 5

ANALYSIS OF VALIDATION STANDARDS

Prior to analyzing the results of this simulation several disparities between this model and the network utilized by Shoch and Hupp and the analytical model of Metcalfe and Boggs must be discussed.

5.1 Analysis of Experimental Network

The network used by Shoch and Hupp was 550 meters long. The specific maximum distance between stations on this network is unknown and assumed to be 550 meters. Using a signal propagation speed of ".77 c worst case" (c is the speed of light; 300,000 KM per second) [18] signal propagation time over a 550M cable is approximately 2.38 microseconds. This gives a 4.76 microsecond contention interval. For example if a station at one end of the network transmitted a message and a second station, at the opposite end of the network, began a transmission the instant before it received the first station's message, the first station would not recognize a collision until round trip propagation time, 4.76 microseconds, had passed. Assuming an even physical distribution of stations on the network the mean contention interval would be 2.38 microseconds. This simulation implements a 2 microsecond mean contention interval equivalent to a 462 meter network with stations distributed evenly across the network. This difference, while seemingly insignificant, could allow for a

variance of up to 2% in efficiency for small packets. Other network characteristics which are known or suspected to vary from the network used by Shoch and Hupp are the capacity-modeled at 3 Mbps versus 2.94 Mbps; "jam" signal duration-modeled at 32 bit times - is the specification minimum value; and delay slot interval-modeled at 512 bit times - is the specification worst case value. Additionally Shoch and Hupp stated they conducted their experiments during periods of minimal network usage. The impact that normal network usage had on their experimental results is undetermined. The most significant difference, which was not identified until after completion of all project lab work, was that the 2.94 Mbps network used by Shoch and Hupp had no interframe spacing time and was implemented with a different set of Ethernet specifications than those available for this project [20].

5.2 Analysis of Analytical Model

The analytical model of Metcalfe and Boggs is developed with a slot time T. This slot time is defined as "the maximum time between starting a transmission and detecting a collision, one end-to-end round trip delay" [19] and is used as the analytical model delay slot interval. The magnitude of the difference between the analytic model delay interval (4 microseconds for a 462 Meter network) and the 512 bit time interval (170 microseconds for a 3 Mbps capacity network) of this model is rather obvious. Interframe spacing deference time, the 9.6 microseconds a station delays after the Ether is sensed clear before beginning a

transmission, is not included in the analytical model. The impact of these disparities is smoothed, somewhat, by the calculations of the analytical model and the collision detection time encountered in both an actual Ethernet network and this simulation model.

Figure 5 presents a graphic representation of the disparities between the analytical model, the network used by Shoch and Hupp, and this simulation model. Note the Extended Contention Interval depicted which results from station interframe spacing implementation. If a station sensed the Ether clear 9.5 microseconds into a second stations Deference Interval it would begin transmitting 9.5 microseconds into the second stations message. This results in a greatly increased overall contention interval during which a collision could occur.

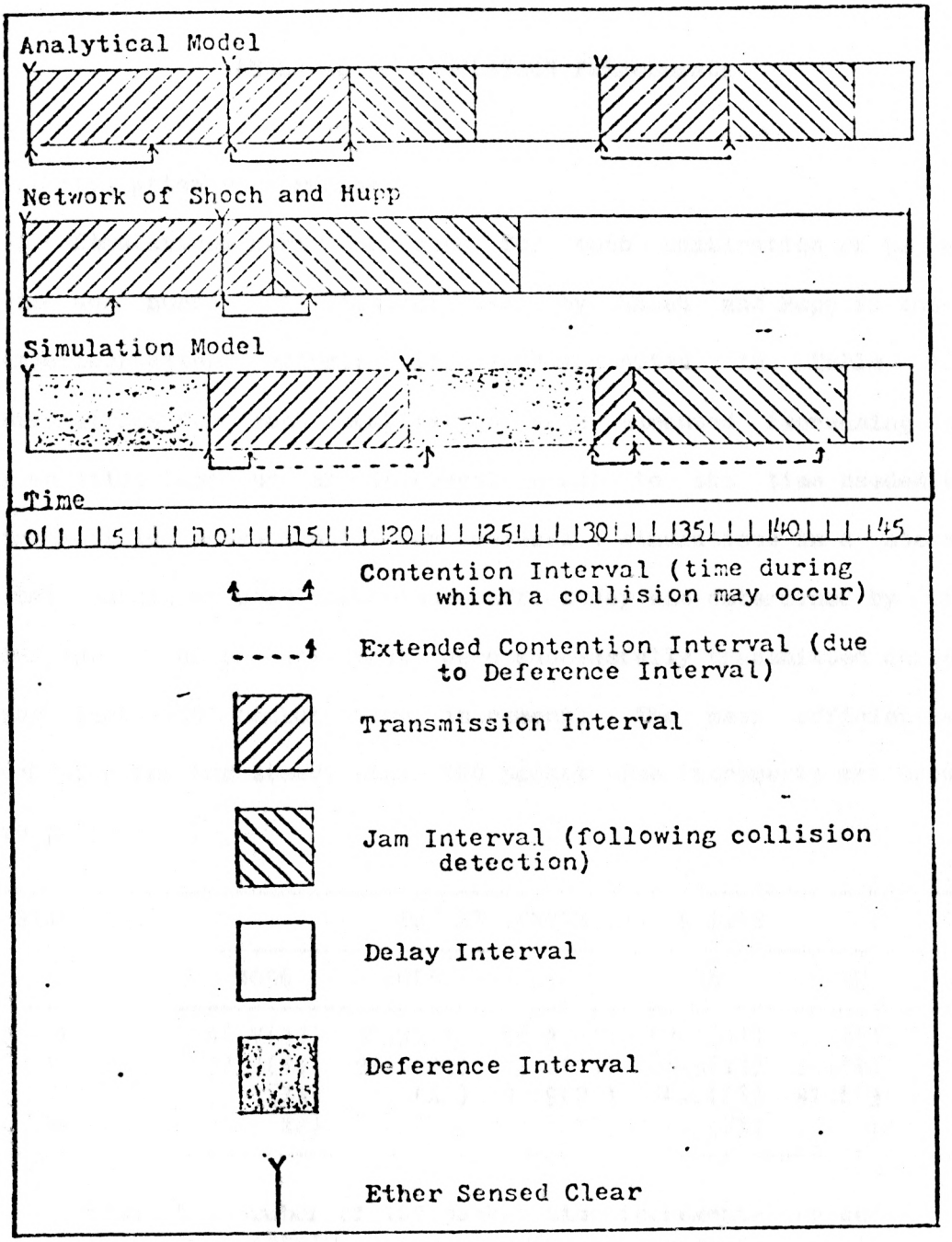


Figure 5

Comparison of Network
Actions for a 32 Bit Packet

CHAPTER 6

ANALYSIS OF SIMULATION PERFORMANCE

6.1 Simulation Performance

Simulations were conducted for each combination of packet size and number of stations used by Shoch and Hupp in their experimentation efforts (previously cited in Table 1). Simulations for each combination were executed increasing the simulation time by an increment equal to the time needed to transmit 100 packets until ten successive simulations in a steady state condition were completed. Efficiency was determined by the percent of 100 packets that were successfully transmitted during the last 100 packet time increment. The mean efficiencies achieved for ten steady state 100 packet time increments are shown in Table 2.

STATIONS (Q)	PACKET LENGTH (P) IN BITS				
	4096	1024	512	48	32
5	98.7(11)	97.3(11)	98.4(11)	63.1(11)	53.9(11)
10	98.1(11)	97.0(21)	93.8(11)	61.3(11)	51.7(11)
32	(XX)	(XX)	92.9(21)	54.6(26)	47.8(33)
64	(XX)	(XX)	(XX)	(XX)	(XX)

NOTE: The number of 100 packet time increments needed to obtain ten successive steady state simulations are indicated in parenthesis.

Table 2

Simulation Efficiency

6.2 Analysis of Performance

Simulation performance (reported in Table 2) when compared to the experimental results of Shoch and Hupp (presented in Table 1) facilitates the development of variances in network efficiency for each given packet size and station number combination. These variances are presented in Figure 6.

Comparison to the analytical model first requires the presentation of the analytical model efficiencies calculated for a network with a 6.5 microsecond mean collision detection time and a capacity of 3 Megabits per second (Table 3). Mean collision detection time was used in lieu of the round trip propagation time due to Metcalfe and Boggs emphasis that T in the analytical model reflected the time required to detect a collision. The value 6.5 microseconds is based on the 4 microsecond round trip propagation time and the 9 microsecond deference interval which make up the Extended Contention Interval. Assuming the stations are evenly distributed across the network and that delayed stations awaken in a uniformly distributed manner during other stations Deference Intervals, the mean collision detection time is equal to 1/2 the network Extended Contention Interval. The variances between this simulation and the analytical model are shown in Figure 7.

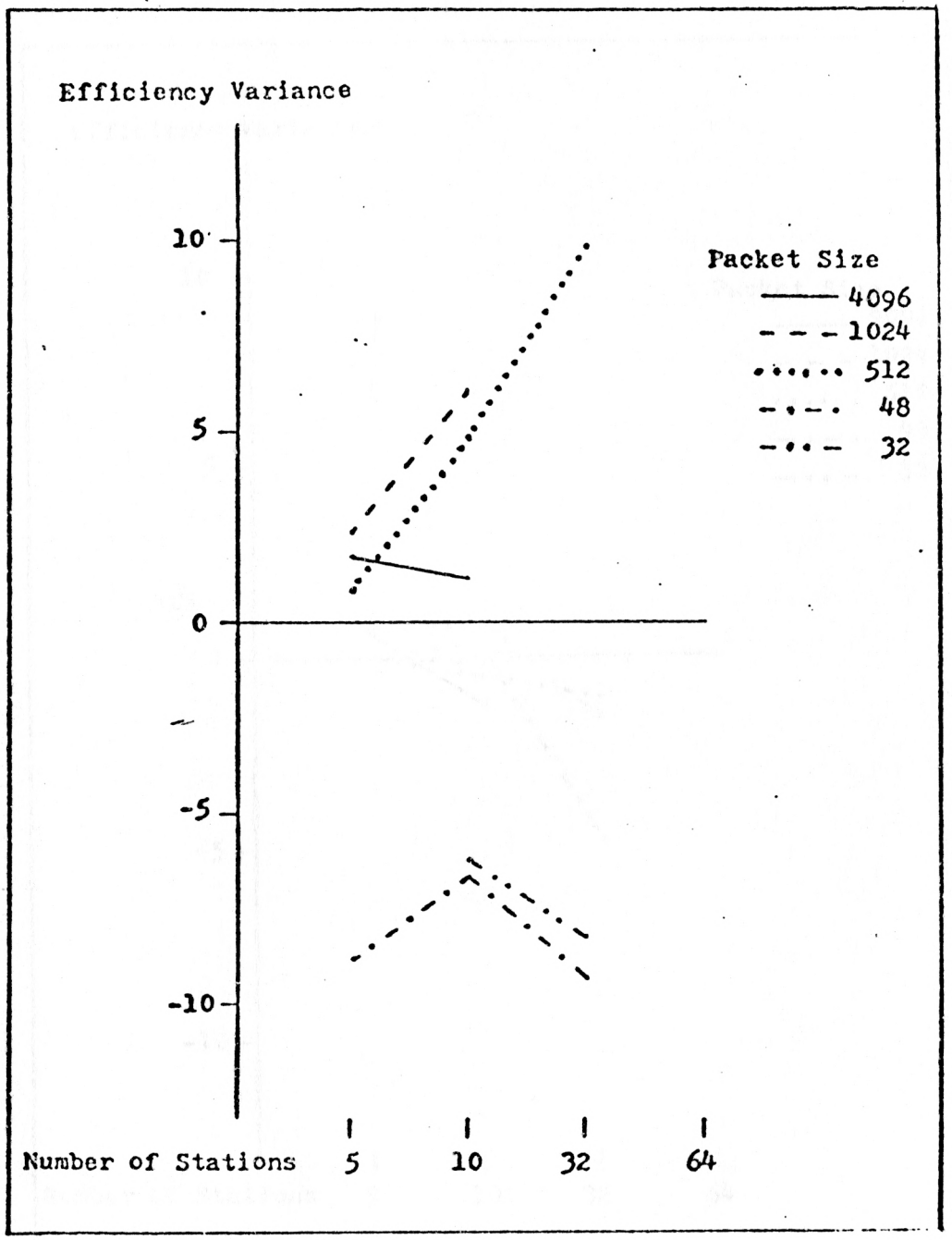


Figure 6

Efficiency Variance Between
Simulation and Shoch and Hupp
Experimental Results

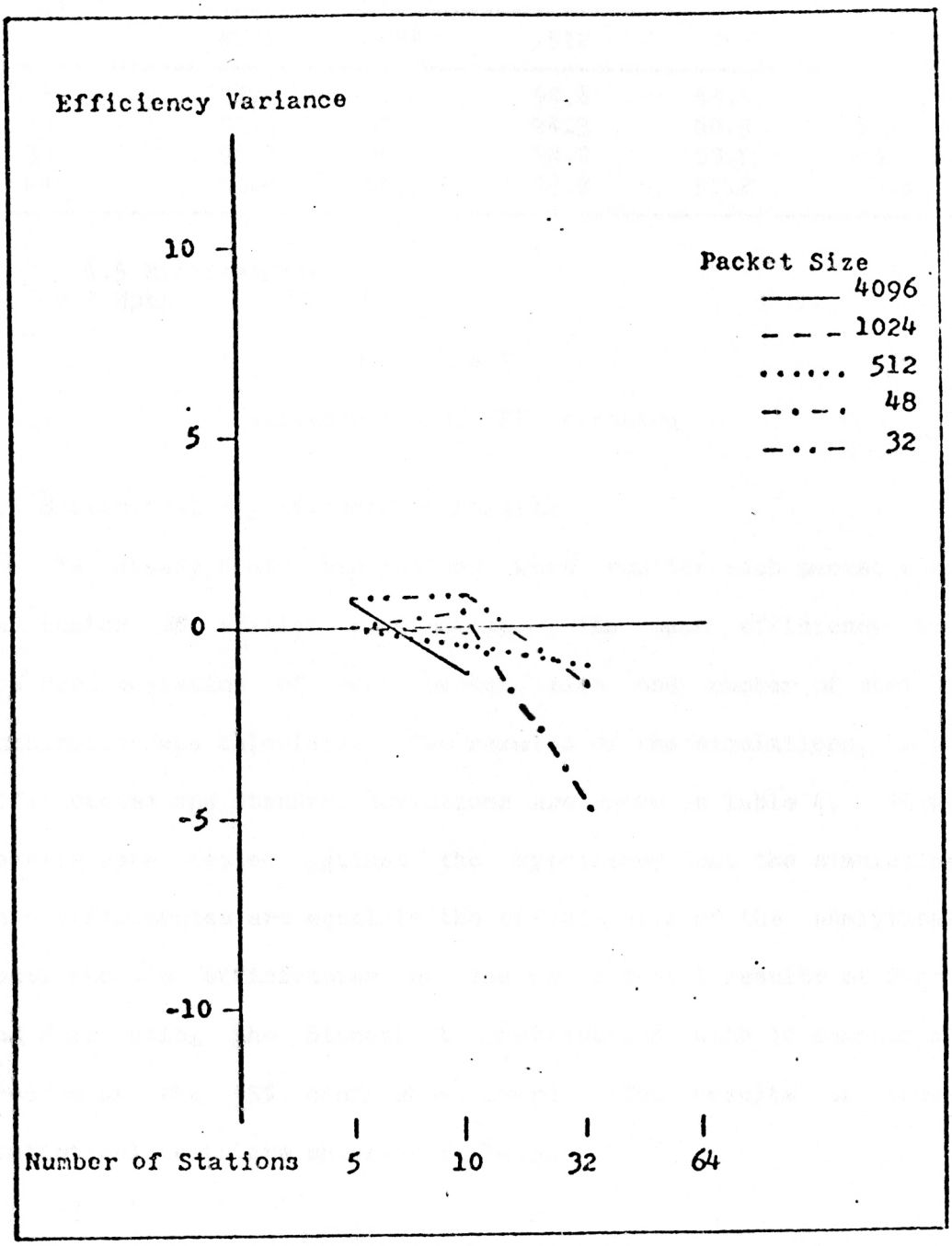


Figure 7

Efficiency Variance Between
Simulation and Analytical Model

STATIONS (Q)	PACKET LENGTH (P) IN BITS				
	4096	1024	512	49	32
5	99.3	97.2	94.8	63.1	53.2
10	99.3	97.0	94.3	60.9	50.9
32	99.2	96.9	94.0	59.5	49.5
64	99.2	96.9	93.9	59.2	49.5

T = 6.5 Microseconds
C = 3 Mpbs

Table 3

Analytical Model Efficiencies

6.3 Statistical Significance of Results

Ten steady state simulations were run for each packet size and number of station combination. The mean efficiency and standard deviation of each packet size and number of station combination was calculated. The results of the simulations, mean efficiencies and standard deviations are shown in Table 4. These results were tested against the hypotheses that the simulation mean efficiencies are equal to the efficiencies of the analytical model and the efficiencies of the experimental results of Shoch and Hupp using the Student t Distribution with 10 degrees of freedom at the 95% confidence level. The results of these statistical tests are shown in Table 5.

Packet Size	Stations	Mean	Standard Deviation	Trials 1	2	3	4	5	6	7	8	9	10
32	5	53.9	1.97	54	55	50	50	55	55	55	55	55	55
	10	51.7	4.62	45	55	54	45	55	55	44	54	55	55
	32	47.8	5.10	54	40	45	51	44	42	46	54	47	55
	64	XX.X	X.XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
48	5	63.1	2.39	64	56	64	64	64	64	64	63	64	64
	10	61.3	3.71	64	56	63	64	55	64	64	56	64	63
	32	54.6	7.28	51	47	64	55	56	64	43	47	55	64
	64	XX.X	X.XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
512	5	94.8	0.40	94	95	95	95	94	95	95	95	95	95
	10	93.8	1.07	92	94	93	92	94	95	95	94	94	95
	32	92.9	1.58	94	94	92	89	92	93	93	95	94	93
	64	XX.X	X.XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1024	5	97.3	0.46	98	97	97	97	97	97	97	98	97	98
	10	97.0	0.68	97	97	97	96	98	96	98	97	97	97
	32	XX.X	X.XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
	64	XX.X	X.XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
4096	5	98.7	1.00	99	99	99	99	100	99	99	99	98	96
	10	98.1	1.30	99	99	99	99	98	95	99	99	97	97
	32	XX.X	X.XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
	64	XX.X	X.XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Table 4

Simulation Results

Hypotheses for each packet size and station number combination:

H_0: Mean efficiency of simulation = validation standard efficiency.

H_1: Mean efficiency of simulation ≠ validation standard efficiency.

Student t value for 95% confidence level with 10 degrees of freedom = 2.26.

Student t formula: $$t = \frac{\bar{x} - p}{\hat{s}/\sqrt{n}}$$

Packet Size	Stations	Results vs analytical model $\bar{x}$	p	$\hat{s}$	t	Accept H_0	Results vs experimental results $\bar{x}$	p	$\hat{s}$	t	Accept H_0
32	5	53.9	53.2	1.97	1.12	yes	53.9	--	1.97	--.--	
	10	51.7	50.9	4.62	.17	yes	51.7	58	4.52	4.30	no
	32	47.8	49.5	5.10	1.05	yes	47.8	56	5.10	5.09	no
	64	XX.X	49.2	X.XX	X.XX	YYY	XX.X	54	X.XX	XX.XX	YYY
48	5	63.1	63.1	2.38	0.00	yes	63.1	72	2.38	11.79	no
	10	61.3	60.9	3.72	0.34	yes	61.3	68	3.72	5.70	no
	32	54.6	59.5	7.28	2.12	yes	54.6	64	7.28	4.08	no
	64	XX.X	59.2	X.XX	X.XX	YYY	XX.X	61	X.XX	XX.XX	YYY
512	5	94.8	94.8	0.40	0.00	yes	94.8	94	0.40	6.32	no
	10	93.8	94.3	1.08	1.46	yes	93.8	89	1.08	14.09	no
	32	92.9	94.0	1.58	2.20	yes	92.9	83	1.58	19.84	no
	64	XX.X	93.9	X.XX	X.XX	YYY	XX.X	85	X.XX	XX.XX	YYY
1024	5	97.3	97.3	0.46	0.00	yes	97.3	95	0.46	15.81	no
	10	97.0	97.1	0.63	0.50	yes	97.0	91	0.63	30.00	no
	32	XX.X	96.9	X.XX	X.XX	YYY	XX.X	90	X.XX	XX.XX	YYY
	64	XX.X	96.9	X.XX	X.XX	YYY	XX.X	92	X.XX	XX.XX	YYY
4096	5	98.7	99.3	1.00	1.88	yes	98.7	97	1.00	5.34	no
	10	98.1	99.3	1.30	2.92	no	98.1	97	1.30	2.68	no
	32	XX.X	99.2	X.XX	X.XX	YYY	XX.X	97	X.XX	XX.XX	YYY
	64	XX.X	99.2	X.XX	X.XX	YYY	XX.X	97	X.XX	XX.XX	YYY

Table 5

Statistical Inference Results

CHAPTER 7

CONCLUSION

The statistical analysis presented in Chapter 6 infers that the objective to validate this simulation by the analytical model was accomplished. A simulation model based on the Ethernet specifications was developed and validated at a 95% confidence level by the analytical model of Metcalfe and Boggs. The inability to statistically infer validation by the experimental results of Shoch and Hupp, particularly considering the apparent discrepencies between the Ethernet specifications used for this project and those that pertained to the network used by Shoch and Hupp, is not surprising.

The inference that an Ethernet implemented with interframe spacing conforms to the efficiencies projected by the analytical model is very provocative. Further simulations are needed over a range of network capacities and lengths to verify that the correlation found between this simulation model and the analytical model is not simply a manifestation of the network characteristics implemented in this simulation model.

Development of a simulation model based on the specifications for the network used by Shoch and Hupp, and the testing of this model to assertain the correlation between the work of Shoch and Hupp and the analytical model would significantly supplement the efforts presented here.

APPENDIX A

LISTING OF STATION SIMULATION

```
250
251
252     "simprefix here"
253     "
254     ******************
255     *   ETHER JOB    *
256     ******************
257     ";
258
259
260       CONST TIME_SLOT = 170;"512 BIT TIME DELAY INTERVAL"
261             C = 3000000;     "LINE CAPACITY IS 3 MBPS"
262             P = 512;         "PACKET SIZE IS 512 BITS"
263             ETHER = 1;       "THIS COUNTER IS USED TO
264                                SIMULATE THE COMMUNICATION
265                                MEDIA"
266             JAM_COUNT = 2;
267             PACKETS_SENT = 3;
268             JOB_COUNT = 4;
269             ATTEMPT_LIMIT = 16;
270             BACK_OFF_LIMIT = 10;
271             INITIAL_STEADY_STATE_LIMIT = 16;
272             CONTENTION_INTERVAL = 4;" PROPAGATION ROUND
273                                      TRIP TIME OF NET "
274             JAM_LENGTH = 11;        " 11 MICROSECONDS
275                                       FOR 32 BIT TIME
276                                       JAM "
277             DEFERENCE_DURATION = 9; " 9 MICROSECONDS
278                                       MODELED FOR 9.6
279                                       MICROSECONDS "
280
281       VAR   TRANSMIT_SUCCEEDING : BOOLEAN;
282             TRANSMISSION_TIME,
283             EXPO_BACKOFF_NUMBER,
284             MESSAGE,
285             JOB_INDEX,
286             RANDOM_NUMBER,
287             ATTEMPTS:INTEGER;
288
```

```
289    PROCEDURE START_TRANSMITTING ( TRANSMISSION_SIGNAL :
290                                          INTEGER );
291
292            " THIS PROCEDURE SIMULATES THE PHYSICAL
293              LAYER ACTION OF INITIATING A TRANSMISSION."
294
295
296      BEGIN
297        TRANSMIT_SUCCEEDING := TRUE;
298        INC_COUNT ( ETHER , TRANSMISSION_SIGNAL )
299      END;
300
301
302    PROCEDURE STOP_TRANSMITTING ( TRANSMISSION_SIGNAL :
303                                        INTEGER );
304
305            " THIS PROCEDURE SIMULATES THE PHYSICAL
306              LAYER ACTION OF TERMINATING A
307              TRANSMISSION.                        "
308
309
310      BEGIN
311        INC_COUNT ( ETHER , - TRANSMISSION_SIGNAL )
312      END;
313
314
315    PROCEDURE JAM_ETHER;
316
317      CONST JAM_SIGNAL = 999;
318
319      VAR   JAM_DURATION:INTEGER;
320
321            " THIS PROCEDURE BROADCASTS A JAM SIGNAL ON
322              THE ETHER OF 32 BITS,  THIS CONSUMES 10.7
323              MICROSECONDS OF TIME AND IS MODELED AS
324              11 MICROSECONDS.                       "
325
326      BEGIN
327        START_TRANSMITTING ( JAM_SIGNAL );
328        FOR JAM_DURATION := 1 TO JAM_LENGTH DO
329            WAIT_TIME (1);
330        INC_COUNT ( JAM_COUNT , 1 );
331        STOP_TRANSMITTING ( JAM_SIGNAL )
332      END;
333
334
```

```
335     PROCEDURE WATCH_FOR_COLLISION;
336
337       VAR   TRANSMISSION_CONTINUES:BOOLEAN;
338             TRANSMISSION_DURATION:INTEGER;
339
340             " THIS PROCEDURE MONITORS A TRANSMISSION
341               IN ORDER TO DETECT COLLISIONS WHEN THEY
342               OCCUR.  WHEN A COLLISION IS DETECTED
343               THE ETHER IS JAMMED WITH A 32 BIT SIGNAL
344               AND THE TRANSMISSION IN PROGRESS IS
345               TERMINATED.  WITHIN THIS SIMULATION THE
346               PHYSICAL LAYER TRANSMISSION OF A PACKET
347               IS REPRESENTED BY THE WHILE TRANSMISSION
348               CONTINUES LOOP OF THIS PROCEDURE.      "
349
350       BEGIN
351         WAIT_TIME ( ( CONTENTION_INTERVAL DIV 2 ) );
352         TRANSMISSION_DURATION := ( TRANSMISSION_TIME -
353                       ( CONTENTION_INTERVAL DIV 2 ) );
354         TRANSMISSION_CONTINUES := TRUE;
355         WHILE TRANSMISSION_CONTINUES DO
356           BEGIN
357             IF ( TEST ( ETHER ) = MESSAGE )
358               THEN
359                 BEGIN                      " NO COLLISION "
360                   IF TRANSMISSION_DURATION <
361                             ( TRANSMISSION_TIME -
362                             ( CONTENTION_INTERVAL DIV 2 )
363                             - DEFERENCE_DURATION )
364                     THEN
365                       BEGIN  " STATION ACQUIRED THE NET "
366                         WAIT_TIME
367                               ( TRANSMISSION_DURATION );
368                         TRANSMISSION_CONTINUES := FALSE;
369                         INC_COUNT ( PACKETS_SENT , 1 )
370                       END
371                     ELSE
372                       BEGIN
373                         WAIT_TIME ( 1 );" PASSAGE OF TIME
374                                     IN EXTENDED CONTENTION
375                                     INTERVAL            "
376                         TRANSMISSION_DURATION :=
377                               TRANSMISSION_DURATION - 1
378                       END
379                 END
380               ELSE
381                 BEGIN            " COLLISION DETECTED "
382                   TRANSMISSION_CONTINUES := FALSE;
383                   JAM_ETHER;
384                   TRANSMIT_SUCCEEDING := FALSE
385                 END
386           END "WHILE TRANSMISSION_CONTINUES"
387       END;
```

```
388
389
390     PROCEDURE CARRIER_SENSE;
391
392       VAR   ETHER_CLEAR,
393             DEFERRING:BOOLEAN;
394             DEFERENCE_TIME:INTEGER;
395
396             " THIS PROCEDURE SENSES THE ETHER CARRIER
397               SIGNAL AND WAITS UNTIL THIS SIGNAL GOES
398               OFF BEFORE BEGINNING INTERFRAME
399               SPACING DEFERENCE OF 9.6 MICROSECONDS.
400               AT THE END OF THIS DEFERENCE PERIOD
401               THE ETHER IS SIGNALED AS CLEAR TO THE
402               TRANSMISSION MANAGEMENT ROUTINE
403               REGARDLESS OF THE PRESENCE OF A CARRIER
404               SIGNAL ON THE ETHER.                    "
405
406       BEGIN
407         ETHER_CLEAR := FALSE;
408         DEFERENCE_TIME := DEFERENCE_DURATION;
409         WHILE NOT ETHER_CLEAR DO
410           BEGIN
411             WAIT_UNTIL ( ETHER );"WAIT UNTIL ETHER CLEAR"
412             DEFERRING := TRUE;
413             WHILE DEFERRING DO
414               BEGIN
415                 WAIT_TIME ( 1 );
416                 DEFERENCE_TIME := DEFERENCE_TIME - 1;
417                 IF DEFERENCE_TIME = 0
418                   THEN
419                     BEGIN
420                       DEFERRING := FALSE;
421                       ETHER_CLEAR := TRUE
422                     END
423               END "WHILE DEFERRING"
424           END "WHILE NOT ETHER_CLEAR"
425       END;
426
427
428     PROCEDURE INITIALIZE;
429
430             " THIS PROCEDURE INITIALIZES BOOLEAN FIELDS
431               INORDER TO INITIATE PROCESSING.  IN THIS
432               SIMULATION THE BOOLEAN FIELDS PRESENTED IN
433               THE ETHERNET SPECIFICATIONS ARE NOT USED.  "
434
435       BEGIN
436         WAIT_UNTIL ( ETHER )     " ETHER CLEAR "
437       END;
438
439
```

```
440
441     FUNCTION UNIFORM_DISTRIBUTION (I , J : INTEGER) :
442                                             INTEGER;
443
444       CONST MULTIPLIER = 25173;
445             INCREMENT  = 13849;
446             MODULUS    = 65636;
447
448             " THIS FUNCTION PRODUCES UNIFORMLY
449               DISTRIBUTED PSEUDO RANDOM NUMBERS BASED ON
450               THE VALUES PASSED INTO THE FUNCTION.  THE
451               INITIAL VALUE OF THE VARIABLE RANDOM NUMBER
452               IS SET TO THE STATION NUMBER IN THE LINK
453               MANAGEMENT MAIN ROUTINE.  THE RANDOM NUMBER
454               GENERATOR IS FROM GROGONOS PROGRAMMING IN
455               PASCAL [11].                              "
456
457       BEGIN
458         IF J <= I
459           THEN UNIFORM_DISTRIBUTION := I
460             ELSE
461               BEGIN
462                 RANDOM_NUMBER := ( MULTIPLIER *
463                             RANDOM_NUMBER + INCREMENT )
464                             MOD MODULUS;
465                 UNIFORM_DISTRIBUTION := I  +  ( (J + 1 -
466                             I) * RANDOM_NUMBER );
467               END "FI";
468       END "UNIFORM";
469
470
471
472     PROCEDURE RANDOMIZE ( MAXIMUM_BACKOFF : INTEGER ;
473                         VAR SLOTS_TO_DELAY : INTEGER );
474
475        VAR    RANDOM_NUMBER : INTEGER;
476
477             " THIS PROCEDURE GENERATES A RANDOM NUMBER
478               WHICH IS DIVIDED BY THE CURRENT MAXIMUM
479               TIME SLOT BACKOFF VALUE.  THE REMAINDER
480               IS USED TO DETERMINE THE NUMBER OF TIME
481               SLOTS FOR A STATION TO DELAY BEFORE
482               ATTEMPTING TO RETRANSMIT A MESSAGE.  THE
483               VALUE RETURNED IS:
484               0 <= SLOTS_TO_DELAY < MAXIMUM_BACKOFF.  "
485
486       BEGIN
487         RANDOM_NUMBER := UNIFORM_DISTRIBUTION ( 0 ,
488                                       MAXIMUM_BACKOFF );
489         SLOTS_TO_DELAY := RANDOM_NUMBER MOD
490                                       MAXIMUM_BACKOFF;
491       END;
492
```

```
493     PROCEDURE BACKOFF ( ATTEMPT_NUMBER:INTEGER );
494
495       VAR    RAND_SLOT,
496              WAIT_LENGTH:INTEGER;
497
498              " THIS PROCEDURE CALCULATES AND EXECUTES THE
499                BACKOFF TIME DELAY WHEN A COLLISION IS
500                ENCOUNTERED.                              "
501
502        BEGIN      "EXPONENTIAL BACKOFF"
503          IF ATTEMPT_NUMBER = 1
504            THEN
505              EXPO_BACKOFF_NUMBER := 2
506            ELSE
507              IF ATTEMPT_NUMBER <= BACK_OFF_LIMIT
508                THEN
509                  EXPO_BACKOFF_NUMBER :=
510                                    EXPO_BACKOFF_NUMBER * 2;
511          RANDOMIZE ( EXPO_BACKOFF_NUMBER , RAND_SLOT );
512          WAIT_LENGTH := ( RAND_SLOT * TIME_SLOT );
513          WAIT_TIME ( WAIT_LENGTH )
514        END;         "EXPONENTIAL BACKOFF"
515
516
517     PROCEDURE SET_STEADY_STATE_ATTEMPTS;
518
519       VAR    COUNT,
520              WORK_VALUE:INTEGER;
521
522              " THIS PROCEDURE USES THE RANDOM NUMBER
523                GENERATOR ROUTINE TO PRESET THE ATTEMPT
524                NUMBER TO A STEADY STATE CONDITION WERE
525                THE ATTEMPT NUMBERS FOR THE STATIONS IN
526                A GIVEN SIMULATION ARE UNIFORMLY
527                DISTRIBUTED FROM 0 TO THE DEFINED
528                INITIAL_STEADY_STATE_LIMIT - 1.          "
529
530       BEGIN
531         WORK_VALUE := UNIFORM_DISTRIBUTION ( 0 ,
532                                INITIAL_STEADY_STATE_LIMIT );
533         ATTEMPTS := WORK_VALUE MOD
534                                 INITIAL_STEADY_STATE_LIMIT;
535         IF ATTEMPTS = 0
536          THEN
537            EXPO_BACKOFF_NUMBER := 0
538          ELSE
539            BEGIN
540              EXPO_BACKOFF_NUMBER := 1;
541              FOR COUNT := 1 TO ATTEMPTS DO
542                EXPO_BACKOFF_NUMBER :=
543                                    EXPO_BACKOFF_NUMBER * 2
544            END
545       END;
```

```
546
547
548
549
550         " THIS IS THE BEGINNING OF THE MAIN TRANSMISSION
551                       MANAGEMENT ROUTINE.                 "
552
553
554     BEGIN
555       ALIVE (JOB_INDEX);
556       INC_COUNT ( JOB_COUNT , 1 );
557       MESSAGE := JOB_INDEX;
558       RANDOM_NUMBER := JOB_INDEX;    " SETS RANDOM SEED "
559       TRANSMISSION_TIME := ROUND ( 1000000.0 * CONV ( P )
560                                       / CONV ( C ) );
561       SET_STEADY_STATE_ATTEMPTS;
562       INITIALIZE;
563       REPEAT
564
565            " THE MAIN ROUTINE OF THIS PROGRAM IS A
566              REPRESENTATION OF THE TRANSMIT LINK
567              MANAGEMENT ROUTINE PROVIDED IN THE ETHERNET
568              SPECIFICATIONS.                             "
569
570         BEGIN
571            "ATTEMPTS := 0; MOVED TO END OF ROUTINE TO
572             ALLOW PRESETTING OF ATTEMPTS TO STEADY STATE
573             CONDITION.                                   "
574
575           TRANSMIT_SUCCEEDING := FALSE;
576           WHILE ( ATTEMPTS < ATTEMPT_LIMIT )
577                 AND ( NOT TRANSMIT_SUCCEEDING ) DO
578             BEGIN
579               IF ATTEMPTS > 0
580                 THEN
581                   BACKOFF ( ATTEMPTS );
582               CARRIER_SENSE;
583               START_TRANSMITTING ( MESSAGE );
584               WATCH_FOR_COLLISION;
585               STOP_TRANSMITTING ( MESSAGE );
586               ATTEMPTS := ATTEMPTS + 1
587             END; "WHILE"
588           ATTEMPTS := 0
589         END "REPEAT"
590       UNTIL ( 1 <> 1 )
591     END.
```

APPENDIX B

LISTING OF SIMULATION MASTER TIMER

```
240
241     "simprefix here"
242     "
243     **********
244     * MASTER *
245     * ETHER  *
246     **********
247     "
248       ; VAR I : INTEGER
249         ; FAC_SET : MAX_FAC
250       ; BEGIN
251           ALIVE
252         ; DCL_COUNTERS ( 4 )
253         ; TRACE ( FALSE , FALSE , FALSE )
254         ; WAIT_TIME ( 17100 )
255
256            " WITH THIS AMOUNT OF TIME AND THE GIVEN
257              LINE SPEED THE PERFECT UTILIZATION OF
258              THE LINE IS 100 PACKETS.  THE MAX VALUE OF
259              COUNTER NUMBER 3 WILL GIVE A WHOLE INTEGER
260              THAT IS EQUAL TO THE % UTILIZATION
261              OF THE LINE.  THIS TIMER IS FOR 512 BIT
262              PACKETS.  THE WAIT TIME IS CALCULATED BY
263              THE FOLLOWING FORMULA.
264
265
266                 PACKET SIZE / CAPACITY  = MESSAGE TIME
267
268                ( MESSAGE TIME * 1000000.0 )  ROUNDED =
269                       MESSAGE TIME IN MICROSECONDS
270
271                  MESSAGE TIME IN MICROSECONDS * 100 =
272                        100 PACKET SIMULATION TIME
273                             IN MICROSECONDS          "
274
275         ; CANCEL ( FALSE )
276         ; REPORT_COUNTERS ( PRINTER )
277         ; CANCEL ( TRUE )
278         END
279     .
280
```

APPENDIX C

SIMULATION USERS GUIDE

C.1 SIMMON

This concurrent simulation of Ethernet is supported by a version of SIMMON, a Simulation Monitor written in CONCURRENT PASCAL. The modified version of SIMMON used for the simulations reported in this document is stored with the label SIMON.PAS in account 73. Four compiled versions of SIMON are stored with the labels SIMON.TSK, SIMON2.TSK, SIMON3.TSK and SIMON4.TSK. These compiled versions have been adjusted to support 5, 10, 32 and 64 station simulations. If a different number of stations are desired line 120 and 126 of SIMON must be changed. There are comments on those lines which assist in determining the changes to be made.

C.2 SIMBATCH.CSS

"SIMBATCH" is a Command Substitution System (CSS) file which may be used interactively or to initiate a background execution of a simulation. The SIMBATCH file contains the necessary system commands to invoke a SIMMON version, request the necessary core space to execute that appropriate SIMMON version and initiate an output file for the simulation being run. Table 6 lists the SIMBATCH versions currently stored, the SIMMON version they invoke, the task size required and the number of stations for that simulation.

SIMBATCH VERSION	SIMMON VERSION	TASK SIZE	NUMBER STATIONS
SIMBATCH	SIMMON	100K	5
SIMBATC2	SIMON2	140K	10
SIMBATC3	SIMON3	200K	32
SIMBATC4	SIMON4	400K	64

Table 6

SIMBATCH.CSS Versions

C.3 Input Files

The SIMBATCH CSS requires an input file with a suffix of ".IN". This input file consists of the label for the simulation master timer program followed by labels for the stations included in the simulation. Each of these labels must be on a separate line. After signing on the system this file can be created in EDIT mode. An example of how an input file can be created follows.

Command	Response
EDIT , Filename.IN "Return"	System creates File with label Filename.IN and Displays "Buffer Empty" message.
CR "Return"	Places user in create mode
MasterTimerLabel "Return"	Enters Master timer program label into file
StationProgramLabel "Return"	Enters station programs label into file
StationProgramLabel "Return" Note: The Station Program Label must be entered once for each station desired on the network. This example has only 2 stations on the network.	
"Return" "Return"	Exits create mode
EN "Return"	Ends EDIT Session and saves newly created file

Input files for the packet size and station number combination simulations reported in this document are stored with the label "ST" followed by the Packet Size and Number of Stations (i.e. the 32 bit packet size, 5 station simulation input file label is "ST325.IN").

C.4 Master Timer Program

The master timer program establishes the duration of the simulation. A copy of the master timer for 100 packets of 512 bits is displayed in APPENDIX B without the SIMMON Prefix. The WAIT-TIME value in line 254 of this program dictates the simulation duration. A master copy of this program is stored

under the label EMASTER.PAS. Copies of this file can be generated through the PEDIT mode. An example of how this may be done follows.

Command	Response
PEDIT EMASTER,Newlabel.PAS "Return"	Displays line one of file EMASTER.PAS. Changes made during this PEDIT session will appear only in file Newlabel.PAS which is generated at the end of the session.
PL 254 "Return"	Causes Line 254 to be displayed.
CH/17100/34200 "Return"	Changes the number 17100 to 34200 in the file named Newlabel.PAS.
EN "Return"	Causes the PEDIT session to end and stores the new file with changes named NewLabel.PAS.

After changing the duration of a simulation the new master timer must be compiled and stored for access by the SIMBATCH CSS. Compilation is accomplished with the command "PAS32 NewLabel,,NS 'Return'." At the end of the compilation a "Compilation Complete" message is displayed. A "RELOCATE NewLabel 'Return'" command generates and stores a SEQ suffixed file NewLabel.SEQ which is accessed by the SIMBATCH CSS. In order to reduce file space usage the intermediate files generated while changing the master timer should be deleted. This is accomlished by entering the commands "DE NewLabel.PAS 'Return'" and "DE NewLabel.OBJ 'Return'".

Master timers currently stored are labeled by the convention "EM" followed by packet size (i.e. the timer for 32 bit packets is labeled EM32) and are for the last simulation run for the cited

packet size. The duration of the timer cannot be checked and it is recommended that a new master timer be created before any additional simulation is conducted. CAUTION- Make sure the name of your master timer is the first label in the input file to be used.

The master timer controls several other functions in addition to the simulation duration. The SIMMON Simulation System reference pamphlet (Kansas State Computer Science Department TR-79-05) should be referenced before using any of these functions.

C.5 Station Program

A master copy of the station program listed in APPENDIX A is stored under the label "EDFAST.PAS". Copies are stored for access by the SIMBATCH CSS under the label convention "ED" followed by packet size (i.e. the station program for a 32 bit packet is labeled "ED32") for 32, 48, 512, 1024, and 4096 bit packet stations. A copy of this program may be generated and changed in the same manner that was described for the master timer program. All values that depict the characteristics of the network being modeled are constants declared in lines 260 through 280. Lines 263 through 268 are name declarations for the four SIMMON counters declared in the master timer. The packet size is declared on line 262.

CAUTION- Ensure that the label for the station desired is in the input file. Remember it must appear in the input file once for each station of that type to be included in the simulation

(i.e. for a 5 station simulation the station program label must be listed 5 times).

C.6 Background Job Stream

Background job stream CSS files are stored under the labels RUN, RUN2, RUN3, RUN4, RUN5. These CSS files make it possible to execute the job stream listed within them without being signed on at the terminal during their execution. They may be referenced and changed through the EDIT mode. The following is an example of a background job stream CSS stored under the name RUN3.CSS.

```
$ BUILD RUN3, JOB
SIGNON RUN3, "Account Number", "Password"
LOG RUN3, LOG
D T
SIMBATCH ST325
D T
SIMBATC2 ST3210
D T
ME "UserName" JOB RUN3 IS DONE
SIGNOFF
$END B
ALLONEW RUN3,LOG,133
SUBMIT RUN3, JOB
D T
$EXIT
```

The entries that pertain to a specific simulation are the SIMBATCH CSS and Input File line and the following "D T" entry. Job stream files may be changed through the EDIT mode. For example, if it was desired to change the RUN3 CSS file listed above and delete the SIMBATCH ST325 job, change the other job to use input file ST4810, rename the changed file RUN7.CSS, and add job SIMBATC3 ST5132 to the job stream, the following would accomplish this.

Command	Response
EDIT RUN3.CSS,RUN7.CSS "Return" RUN3. this session.	Displays line 1 of file CSS. Changes made during Edit session will appear only in file RUN7.CSS which is generated at end of the
CH/RUN3/RUN7/*"Return" "RUN3 to	Changes all instances of RUN7" and displays lines changed.
PL 5 "Return" displayed.	Cause line 5 to be This is "SIMBATCH ST325."
DL "Return"	Cause line 5 to be deleted & line 6 to be displayed, this is "D T".
DL "Return"	Cause line 6 to be deleted & line 7 to be displayed, this is "SIMBATC2 ST3210".
CH/32/48 "Return"	Causes line 7 to read "SIMBATC2 ST4810" and displays changed line.
"Return"	Causes the next line to be displayed, this is "D T".
IL SIMBATC3 ST51232 "Return"	Causes this line to be inserted into the file following the line previously displayed.
IL D T "Return"	Causes this line to be inserted into the file following the line previously displayed.
PL1 "Return"	Causes Line 1 of the file to be displayed.
TY23 "Return" displayed	Causes the line being and the following 23 lines of the file to be displayed. It would look like this

```
1       $BUILD RUN7, JOB
2       SIGNON RUN7, "AccountNumber", "Password"
3       LOG RUN7, LOG
4       D T
7       SIMBATC2 ST4810
8       D T
8.01    SIMBATC3 ST51232
8.02    D T
9       ME "UserName" JOB RUN7 IS DONE
10      SIGNOFF
11      $END B
12      ALLONEW RUN7, LOG, 133
13      SUBMIT RUN7, JOB
14      D T
15      $ EXIT
```

```
EN "Return"                          Causes the EDIT session to
end
                                     and the file RUN7.CSS to be
                                     stored.
```

This job stream can be executed by entering the command "RUN7 'Return'". Once the system displays a date and time the user may SIGNOFF and the job stream will continue to execute.

NOTE- The "Account Number" and "Password" have to be for the account that the SIMMON, SIMBATCH, master timer, station program, input, and background job stream files are stored in. The "UserName" specifies the user to whom the message "JOB RUN7 IS DONE" is to be provided when the job stream terminates. This will only be done if the user is signed onto the system at a terminal with the cited "UserName".

C.7 Output Files

The SIMBATCH CSS causes an output file with a suffix of ".OUT" to be generated for the associated job being run. The full name of this output file is the same as the job input file except the suffix is ".OUT" instead of ".IN." An output file can be

viewed on a terminal by using the command "EDIT OutputFile.OUT 'Return'". This causes the first line of the "OutputFile" to be displayed. The complete file may be viewed with the command "TY12 'Return'". The "EN 'Return'" command will terminate this edit session. A printed copy of the output file may be obtained with the command "COPYA OutputFile.OUT, PrinterID 'Return'", where the Printer ID is the device you desire the output to be printed on (i.e. the PrinterID for the high speed printer in he Mini Lab is "PR:", for the printer in room 14 "PA3A:"). When you have finished with an output file it is recommended that you delete the file with the command "DE OutputFile.Out 'Return'".

CAUTION- If a job stream contains more than one interation of the same job and its associated input file, and each of these input files have the same name, the output file for this job which is generated by the first occurrence of the job will be over written by the second and any subsequent occurrences. This will result in the loss of simulation results.

C.8 Interpreting Output Results

Output results consist of an 8 line SIMMON statistical report. The following is an example.

THE CURRENT TIME IS 136500

COUNTER	TIMES SET	AVE VALUE	MAX VALUE	MIN VALUE
1	245	124.204	2003	0
2	12	6.500	12	1
3	98	49.500	98	1
4	5	3.000	5	1

SIMULATION TERMINATES AT TIME 136500

The following is a brief explanation of the most pertinent information provided by this report:

1. "THE CURRENT TIME IS" value indicates the simulation time unit at which the simulation terminated.

2. "COUNTER 1 TIMES SET" value indicates the number of times the ETHER variable was changed. This value should equal 4 times the counter 2 value plus 2 times the counter 3 value. If the simulation terminates during a transmission, but before a collision is detected, the Counter 1 value will equal 4 times the counter 2 value plus 2 times the counter 3 value plus 1.

3. "COUNTER 2 TIMES SET" value indicates the number of collisions that occured during simulation.

4. "COUNTER 3 TIMES SET" value indicates the number of good transmissions that were completed during the simulation.

5. "COUNTER 4 TIMES SET" value indicates the number of stations that were simulated as being on the network.

6. The job that an output report is for can be determined from the simulation time value and the number of stations listed in the output report.

Network efficiency for a 100 packet simulation is equal to the number of good messages transmitted in the time interval that 100 packets could be transmitted. If it is desired to run a series of simulations, as was done for this report, the efficiency for a given 100 packet interval is obtained by subtracting the number of good transmissions in the previous simulation from the number of good transmissions in a subsequent simulation that is run for a duration 100 packet transmission time intervals longer

than the previous iteration. If a series of 3 efficiency values is desired run 4 simulations, the first for a duration equal to the simulation time needed to transmit 100 packets, the second for a duration equal to the simulation time needed to transmit 200 packets. The efficiency value for the second 100 packet time interval is equal to the total number of good transmissions in the second simulation minus the total number of good transmissions in the first simulation. The second efficiency value is obtained by running a third simulation for 300 packet transmission time intervals and subtracting the total number of good transmissions in the second simulation from the total number of good transmissions in the third simulation. To obtain the third efficiency value increase the simulation by another 100 packet time interval, and subtract the total number of good transmissions in the third simulation from the total number of good transmissions for this, the fourth simulation.

C.9 100 Packet Time Interval

The simulation time needed to transmit a packet at a resolution where one time unit equals one microsecond is obtained with the following formulas.

((Packet Size/Ether Capacity) X 1000000)Rounded = Time interval needed to transmit one Packet.

One Packet Time Interval X 100 = 100 Packet Time Interval.

C.10 Checking Job Stream Status

The progress being made by a background job stream may be checked only if all the output files from previous simulations

have been deleted prior to starting the current job stream. If this is the case, issuing the command "D F,.OUT 'Return'" after signing on, will cause a display of output file names, the number of lines they contain (NLR column) and the date the file was created, to appear on the terminal. When the number of lines for a file is zero this is the normal indication that the job creating that file is currently executing. If by comparing the order jobs appear in the job stream to existing ".OUT" files it is determined that a job is executing that appears in the job stream after a job which has zero lines on the display, the job with zero lines that is not currently executing, reached an abnormal termination.

The reason for this abnormal termination may be listed in the LOG file which will be discussed later. After the job stream has terminated the LOG file may be printed to determine the cause of an abnormal termination.

A cursory check may be made by entering the command "MA 'Return'". You do not have to sign on before doing this. The "MA" command causes the tasks currently executing on the system to be displayed. Your job stream (i.e. "RUN3") is considered a systems task. If it is executing it will be included in the resulting display which indicates the present task size and status (i.e. active, wait, rolled) of your job stream. Unfortunately, the only significant status information provided by this display is whether your job stream is executing or not. If it is not listed in the display your job stream has finished processing.

C.11 Accessing the Log File

The LOG file created for each job lists the job "IN" label, date and time the job starts, each SIMMON master timer and input Station program request line, several lines indicating job termination, the stop date and time, and the elapsed time. Below is an example of LOG file entries for one job, which was for a 5 station, 32 bit packet simulation.

```
-  1     D T
-  2     12/08/81  01:10:22
-  3     SIMBATCH ST325
-  4     M_TIMER =
-  5     ENTER FILE NAME
-  6     ENTER FILE NAME
-  7     ENTER FILE NAME
-  8     ENTER FILE NAME
-  9     ENTER FILE NAME
- 10     ENTER FILE NAME
- 11     SIMULATION FINISHED CANCEL TASK
- 12     INFINITE WAIT ERROR
- 13     ERROR IN KERNEL
- 14     TASK PAUSED
- 15     RUN3    -END OF TASK CODE= 255
- 16     ELAPSED TIME=00:04:16
```

The LOG file for a job stream has a full file name of "'JobStreamId'.LOG" (i.e. "RUN3.LOG"). When a job stream has terminated it contains a set of entries for each job in the job stream. This LOG file may be accessed in the same manner as an ".OUT" file, through either an EDIT or COPYA command. If the "SIMULATION FINISHED CANCEL TASK" and "INFINITE WAIT ERROR" lines are not in the LOG file for a given job, that job did not terminate normally and no output information will be in the ".OUT" file for that job. When this occurs, if no error condition is listed, termination resulted from a system failure.

NOTE- The LOG file is inaccessable while a job stream is executing. If you attempt to EDIT the file it will result in a scratch file error. This may be corrected with the command "DE ED.SCR 'Return'" when the EDIT session is complete.

The elapsed time of a simulation may be used to estimate subsequent execution times. This provides a basis for estimating the run time of a job stream executed in background and enables the user to schedule his access needs to the system to retrieve output and initiate subsequent simulation job streams.

C.12 Executing Without An Input File

A CSS labeled SIM is stored in account 73 that facilitates simulation with up to 5 station processes in an interactive manner. This CSS is executed by entering the command "SIM". This causes the SIM CSS to invoke SIMMON which display "M_TIMER =", at which time the user enters the master timer program name. SIMMON will then display "ENTER FILE NAME". The user then enters the station program name desired. This exchange continues until the user has entered the number of station programs he desires to (not more than 5). After the user has entered all the stations into the simulation that he desires, the next time SIMMON displays "ENTER FILE NAME" the user hits the "Return" key.

The user must stay signed on throughout the duration of the simulation. When the simulation terminates the output is displayed and "SIMULATION FINISHED CANCEL TASK", "INFINITE WAIT ERROR", "ERROR IN KERNEL", and "TASK PAUSED" messages are diplayed. At this time the user enters "CAN" then "Return". An

end of task message is displayed including elapsed time. NOTE- No ".OUT", or ".LOG" file are created using the SIM CSS.

C.13 The Null Process

The Null Process program ("NULLPROC") maintained in account 73 is needed for the proper execution of SIMMON when all simulation stations are in a wait state. The SIMMON versions have a maximum process number greater than required for the master timer and station programs needed for simulation to ensure the inclusion of 1 Null Process program per simulation.

C.14 Station Simulation Versions

Three versions of the station simulation program are stored in account 73. They are labeled EDNEXT, EDSTEADY, and EDFAST.

EDNEXT is the basic station simulation program. Each time unit is equivalent to 1 microsecond, and the passage of each time unit for each station, requires a call on SIMMON. It is very slow. The initial attempt number for a message transmission is set to zero.

EDSTEADY is the same as EDNEXT except a steady state initialization routine is added which sets the initial message attempt number to a value between zero and 15 using the Uniform Distribution Function of the simulation.

EDFAST, is the same as EDSTEADY except that the passage of time in a station that acquires the network is facilitated by one "WAIT_TIME" call on SIMMON for the transmission duration remaining after the Extended Contention Interval has passed rather than

passing time in one microsecond intervals in the WATCH_FOR_COLLISION routine.

APPENDIX D

REFERENCES CITED

1. John F. Shoch and Jon A. Hupp, "Measured Performances of an Ethernet Local Network", Communications of the ACM, Vol 23, Number 122, (December 1980) 711-721.

2. Robert M. Metcalfe and David R. Boggs "Ethernet: Distributed Packet Switching for Local Computer Neworks", Communications of the ACM, Volume 19, Number 7 (July 1976) 395-403.

3. The Ethernet, A Local Area Network, Data Link Layer and Physical Layer Specifications, Version 1.0, Xerox Corporation, September 30, 1980.

4. V. E. Wallentine, W. J. Hankley, R. A. McBride, SIMMON - A Concurrent Pascal Based Simulation System, TR-79-05, (Manhattan, Kansas: Department of Computer Science, Kansas State University) April, 1978.

5. Metcalfe and Boggs, p. 395

6. Ethernet Specifications, p. 1.

7. Andrew S. Tanenbaum, Computer Networks, (Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1981) pp. 15-21.

8. Metcalfe and Boggs.

9. Ibid. p. 401.

10. Shoch and Hupp, p. 719.

11. Peter Grogono, Programming in PASCAL, Revised Edition, (Reading, Massachusetts: Wesley Publishing Co. 1980), p. 118.

12. Donald E. Knuth, Seminumerical Algorithms, Vol. 2, The Art of Programming, (Menlo Park, California; Addison-Wesley Publishing Co., 1969) p. 155.

13. Ethernet Specifications, p. 24.

14. Ibid.

15. Ibid. p. 23.

16. Ibid. p. 25.

17. Ibid. p. 35.

18. Ibid. p. 49.

19. Metcalfe and Boggs, p. 400.

20. John Shoch, Telephone Interview, 14 Dec. 1981.

A CONCURRENT SIMULATION OF ETHERNET

by

CHARLES M. MEDVITZ

B.S., San Diego State University, 1974

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNVERSITY

Manhattan, Kansas

1982

ABSTRACT

A Concurrent Simulation of Ethernet

Ethernet is a local area network utilizing a Carrier Sense Multiple Access (CSMA) protocol with Collision Detection (CD). All stations on the network share the broadcast media (Ether) which is a coaxial cable. The simulation of Ethernet presented in this document is an attempt to model this network environment and validate the simulation implementation of the CSMA/CD protocol utilized. Validation of this simulation model consists of achieving a steady state condition on the network in the special case of all stations on the network continuously prepared to transmit a message, and comparing the efficiency of the simulation network to the experimental results of Shoch and Hupp attained on an active Ethernet [1] and the analytical model of Metcalfe and Boggs [2].