A MODEL FOR END-TO-END DELAY IN DISTRIBUTED COMPUTER SYSTEMS

BY

John J. Deeds

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

APPROVED:

F. Ricci, Chairman

D. Schaefer

G. Kossakes

May, 1993

Falls Church, Virginia

A MODEL FOR END-TO-END DELAY IN DISTRIBUTED COMPUTER SYSTEMS

by

John J. Deeds

Committee Chairman: Fred Ricci
Electrical Engineering

(ABSTRACT)

Mitchell [1,2] describes end-to-end performance for a LAN-based computer system as the total system throughput and delay for a single-thread transaction. This model is used for a variety of applications. The single-thread transaction might, for example, be a remote database update or a real-time control activity. To model end-to-end performance, one must include the host computers, the network interface units (NIUs), the host-NIU links, and the NIU-NIU links. Based on Jackson's Theorem, total delay for single-thread transaction traversing a computer network can be approximated by the sum of delays in the host computers, the network interface units, the host-NIU links, and the NIU-NIU links.

The host computer performance model can be refined by applying execution path analysis. Execution path analysis examines the structure of each software routine to be executed and provides an expression of time delay as a function of probabilities associated with conditional branches and a function of data input size.

Spreadsheet models provide quick and convenient solutions for purposes of performing computer system tuning and capacity planning as demonstrated by Thomas [10].

This thesis paper extends the typical modeling approach by providing more detailed analysis of host computer delay, more specifically, the execution path analysis. In addition, spreadsheet models are implemented to demonstrate the execution path analysis and to provide comparisons with previously implemented models.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Overview

Mitchell [1,2] describes end-to-end performance for a LAN-based computer system as the total system throughput and delay for a single-thread transaction. This model is used for a variety of applications. The single-thread transaction might, for example, be a remote database update activity or a real-time control activity. Mitchell suggests that to model end-to-end performance, one must include the host computers, the network interface units (NIUs), the host-NIU links, and the NIU-NIU links. Thus, one must model not only the LAN performance, but the computer performance as well. The total delay for single-thread transaction traversing a computer network can be approximated as the sum of delays in the host computers, the network interface units, the host-NIU links, and the NIU-NIU links by using Jackson's Theorem [12].

LAN and computer network performance models are typically constructed based on queuing theory. Mitchell and others at CONTEL Information Systems modeled end-to-end computer network performance as described above using queuing theory. The details of these studies are provided in several publications and are summarized by Stallings [3].

When tracing a single-thread transaction's path through a host computer, delay elements such as the following arise:

1. Time in operating system wait queues.

2. Time executing CPU instructions associated with the application program and with the LAN protocol routines.

3. Time in the terminal I/O queue, and writing to, or reading from a terminal.

4. Time in the disk I/O queue, and writing to, or reading from a disk device.

The above delay elements are analyzed in detail by Leung [9]. In a host computer, the delay elements are often modeled as follows:

1. The execution path length for each software routine is determined by estimating the total number of instructions executed in the path traversed by the single-thread transaction. The time delay for the software routine is computed by multiplying the total instructions by the average delay per instruction, where the average delay per instruction is based on an assumed instruction mix.

2. Terminal I/O delay and disk I/O delay are often addressed by incorporating I/O instructions into the assumed instruction mix.

3. Delay in operating system wait queues is estimated by applying an overhead factor.

The host computer performance model can be refined by applying execution path analysis. Execution path analysis examines the structure of each software routine to be executed and provides an expression of time delay as a function of probabilities associated with conditional branches and as a function of data input size. The host computer delay can then be incorporated into an end-to-end performance model to establish more realistic estimates of total system delay and throughput.

Spreadsheet models provide quick and convenient solutions for purposes of performing computer system tuning and capacity planning as demonstrated by Thomas [10].

This thesis paper extends the typical modeling approach by providing more detailed analysis of host computer delay, more specifically, the execution path analysis. In addition, spreadsheet models are implemented to demonstrate the execution path analysis and to provide comparisons with previously implemented models.

1.2. Organization

Section 2 presents the background theoretical material as well as the new theoretical concepts introduced in this paper. Section 3 summarizes the design approach utilized. Sections 4 and 5 present the implementation and results in the form of spreadsheet models. Section 6 presents the conclusions.

2. CONCEPTS

2.1. Introduction

This section presents the background theoretical material as well as the new theoretical concepts introduced in this paper. The Decomposition Approximation Method for modeling end-to-end delay is extended by incorporating a method for esimating the host application path length.

2.2. Queueing Analysis

A single-thread transaction can be viewed as passing through a series of queues and servers. The servers include the host computers, network interface units, host-NIU links, and NIU-NIU links, providing computing power and data transmission paths. Queues (such as memory) within the host computers and network interface units serve as buffers that prevent loss of messages and data. This approach is the Decomposition Approximation Method.

2.2.1. Decomposition Approximation Method

Maglaris, Lissack, and Austin [11] proved that the Decomposition Approximation Method approximates delay in the open network defined in Jackson's Theorem. Jackson's Theorem [12], later proven by Gelenbe and Mitrani [13], states that under the following conditions each node is independent and that delay can be determined for each node and then summed for an overall result:

1. Messages enter the system with a Poisson distribution rate.

2. Messages are serviced at each node on a first come, first served basis with an exponential distribution rate.

3. Queues are large enough that they are not saturated.

Mitchell [1,2] developed an N stage queuing model to analyze the end-to-end delay for a single-thread transaction traversing a LAN. The LAN model architecture was described as shown in Figure 1 where a host computer could be attached to m network nodes, but a workstation could only be attached to one other node in the network.

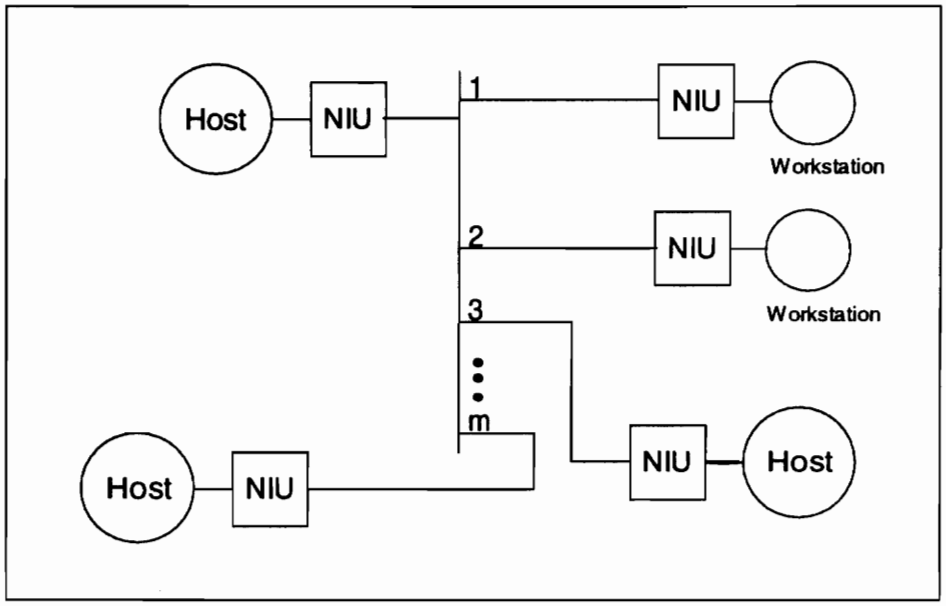

Figure 1. LAN Model Architecture

The decomposition approximation method is used to obtain the model results. It had previously been shown that the total end-to-end delay for the transaction is approximately the sum of the delays calculated at each stage as shown in Figure 2. The delay elements are modeled as a network of tandem queues. The following assumptions were provided for the mathematical model of the LAN:

1. The arrival rate from each set of computers connected to one BIU is uniform over all sets of computers.

2. There is only one transaction type.

3. For each query there is one response.

Michell chose to base his model on the decomposition approximation method rather than exact solution methods because of the limitations they impose on the service disciplines allowed. He determined that the use of the preemptive-resume priority scheme was most appropriate for the LAN model. He reported that since the decomposition approximation method assumes exponential service times, it provides conservative estimates of system performance.

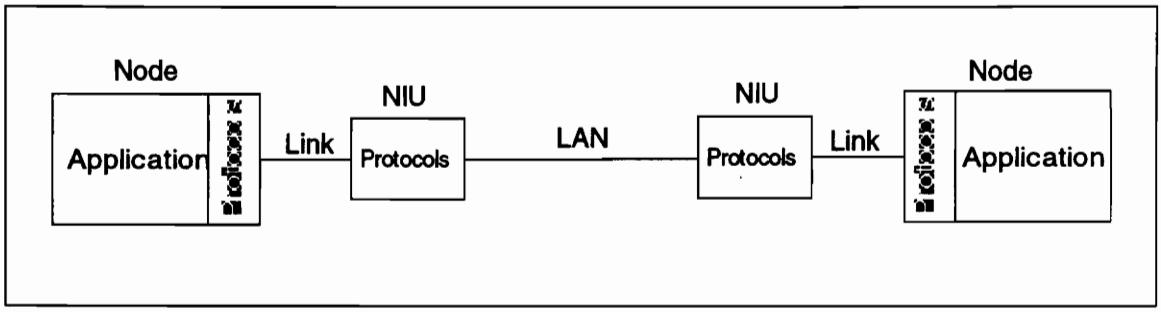

Figure 2. Single-Thread Path Through the LAN

The LAN model is divided into four types of components that are referred to as the network servers. These are:

1. The nodes (host computers and/or workstations).
2. The node/NIU interfaces.
3. The NIU processors.
4. The NIU-NIU links (employing CSMA/CD discipline).

2.2.2. Node Delay

Delay is analyzed for two classes of processors, host computers and workstations. Although host computer delay is the primary focus of this analysis, an analysis of workstation delay will be provided as well. Both host computers and workstations are modeled as single resource, multiple class servers.

A processing overhead factor is used to account for system overhead, such as delay contributed by the operating system. Our model will incorporate an overhead factor to account for this delay. Thus, the host computer capacity is

$$NEPC = (1 - NOH)\, NIPS \qquad (1)$$

where *NEPC* is node effective processing capacity, *NOH* is node overhead, and *NIPS* is node average machine instructions per second.

Five classes of activity are served at each node on a preemptive-reserve basis:

1. Link-in. Link level protocol processing for inbound transactions.

2. Protocols-in. High level protocol processing for inbound transactions.

3. Application. Application level processing.

4. Protocols-out. High level protocol processing for outbound transactions.

5. Link-out. Link level protocol processing for outbound transactions.

The following service priority order is assigned to activities on host computers:

1. Link-in.

2. Link-out.

3. Protocols-in.

4. Protocols-out.

5. Application.

In addition, the following service priority order is assigned to activities on workstations:

1. Link-in.

2. Protocols-in.

3. Application.

4. Link-out.

5. Protocols-out.

Node service time is computed by dividing the execution path length in machine instructions by the processing capacity, *NEPC*, expressed in machine instructions per second. The following assumptions are made regarding the node model:

1. A single input queue and single server are provided.

2. Each class of activity is represents an independent Poisson distributed source with mean arrival rate λ_i.

3. Each class of activity is served on an exponentially distributed basis with mean service time S_i.

4. Queue sizes are of sufficient size such that no data is lost.

Stallings [3] explains that for the single-thread path described by Mitchell with an arrival rate λ, mean service time s, and utilization ρ, each queue is stable if

$$\rho = \lambda s \leq 1$$

In addition, the end-to-end system is stable if, for an N stage path

$$\lambda \leq \frac{1}{Max[s_i]} \quad \text{where } s = \sum_{i=1}^{N} s_i$$

that is, the inter-arrival time does not exceed the maximum service time for any step within the single-thread path. Then the total delay over the N stage path is

$$D = \sum_{i=1}^{N} D_i$$

The mean transaction delay for each class of activity within a host computer is

$$t_{qj} = \frac{1}{1 - \sum_{i=1}^{j-1} \rho_i} \left\{ \frac{\sum_{i=1}^{j} \rho_i s_i}{1 - \sum_{i=1}^{j} \rho_i} + s_j \right\} \quad \text{where } \rho_i = \lambda_i s_i \tag{2}$$

In this model, m is introduced to specify the number of concurrent threads being serviced by a host computer, and therefore $m\lambda$ is the arrival/departure rate at a host computer node. The arrival/departure rate for workstations is just λ.

2.2.3. Node/NIU Interface Delay

At the node/NIU interface, the effective transfer rate for the half-duplex link is calculated as

$$ETR = (1 - IOH)\, ITR \tag{3}$$

where *IOH* is the interface overhead due to overhead bits and *ITR* is the transfer rate for the link in bits per second. Several assumptions are made for the node/NIU interface model:

1. Single waiting line and single server are provided.
2. Two classes of activity are transactions from the node and transactions from the NIU.
3. No priority is assigned.
4. Each class of activity is represents an independent Poisson distributed source with mean arrival rate λ_i.
5. Each class of activity is served on an exponentially distributed basis with mean service time S_i.
6. Queue sizes are of sufficient size such that no data is lost.

The following equations are provided for t_{qj}

$$t_{qj} = \rho\bar{s}/(1-\rho) + s_j \tag{4}$$

where

$$\rho = \lambda'\bar{s},$$
$$\lambda' = \lambda_1 + \lambda_2,$$
$$\bar{s} = (\lambda_1 s_1 + \lambda_2 s_2)/\lambda'$$

For the host/NIU interface, $\lambda_1 = \lambda_2 = m\lambda$. For the workstation/NIU interface, $\lambda_1 = \lambda_2 = \lambda$.

2.2.4. NIU Processor Delay

NIU delay is modeled in a similar manner to computer and workstation delay with the exception that the classes of activity are served in the following order of priority:

1. Network link-in. Link level processing for transactions inbound from the LAN.
2. Node link-in. Link level processing for transactions inbound from the node.
3. Network link-out. Link level processing for transactions outbound to the network.
4. Node link-out. Link level processing for transactions outbound to the node.
5. Protocols. Transport and network protocol processing.

2.2.5. NIU-NIU Link Delay

For the NIU-NIU link, coaxial cable capable of a 1.544 mbps data rate is chosen. The carrier sense multiple access with collision detection (CSMA/CD) access control method is incorporated into the LAN model. The following assumptions are made for the queuing analysis:

1. Poisson arrivals.
2. Negligible collision detection time compared to propagation time.
3. Retransmissions treated as independent Poisson arrivals.
4. Instantaneous channel sensing.
5. Uniform propagation time between any two nodes; equal to the maximum propagation time.
6. Packets retransmitted based on Binary Exponential Backoff rule.
7. Random interval parameters for Backoff algorithm uniformly distributed and identical at all NIUs.

The following equation is provided for normalized throughput

$$S = \frac{Ge^{-aG}}{(1+a)Ge^{aG} + (1+aG)(1-e^{-aG})^2 + 1} \tag{5}$$

where $S = \lambda\ T$ and G is the offered packet traffic including retransmissions. Finally, the average packet transmission delay is

$$D = (A_1 + A_2 + A_3)T \tag{6}$$

where A_1 is normalized time delay due to collisions, A_2 is time delay due to retransmissions and rescheduling, and A_3 is propagation and transmission time. Given N1, the average number of times a packet encounters a collision or busy NIU, N2, the average number of times a packet encounters a collision, R1, the normalized mean wait period after a collision, and R2, the normalized mean wait period after a busy NIU is found, then

$$A_1 = N_2(W+a)$$
$$A_2 = R_2(2^{N_2+1} - 1) + (N_1 - N_2)R_1$$
$$A_3 = a+1$$

where

$$W = \frac{1-e^{-aG}}{G} - ae^{-aG}$$
$$N_1 = G/S - 1$$
$$N_2 = (1+aG)e^{aG} - 1$$

2.3. Execution Path Analysis

Execution path analysis examines the structure of each software routine to be executed and provides an expression of time delay as a function of path length, path iteration, probability that the path will be traversed, and data input size (see diagram). It is not necessary that the program already exist to perform the analysis. Analysis may be based on design information.

The execution path analysis is based on the following elements:

1. Analysis of basic constructs in a structured program.

2. Time complexity analysis.

3. Sensitivity analysis for response time as a function of application program path length.

2.3.1. Analysis of a Structured Program

McCabe [4] applied graph theory to develop a measure of complexity based on the number of vertices, edges, and connected components in a model of the program. He studied program complexity with the goal of reducing complexity in order to improve testability and maintainability. Some of McCabe's methods can be applied to the determination of application path length.

McCabe asserts that a structured program can be described by some combination of five constructs:

1. Sequence B_1; B_2.

2. If B_1, Then B_2, Else B_3.

3. While B_1, Do B_2.

4. Case B_1 of (B_2; B_3; ...; B_n).

5. Do B_1 Until B_2.

These constructs can be diagrammed as shown in Figure 3. Each labeled node (B_n) represents a block of code of some undetermined length. In this model of a program, all of the code in a program is contained within blocks. Thus the program can be modeled by a some combination of the five constructs, with each construct containing some number of blocks.

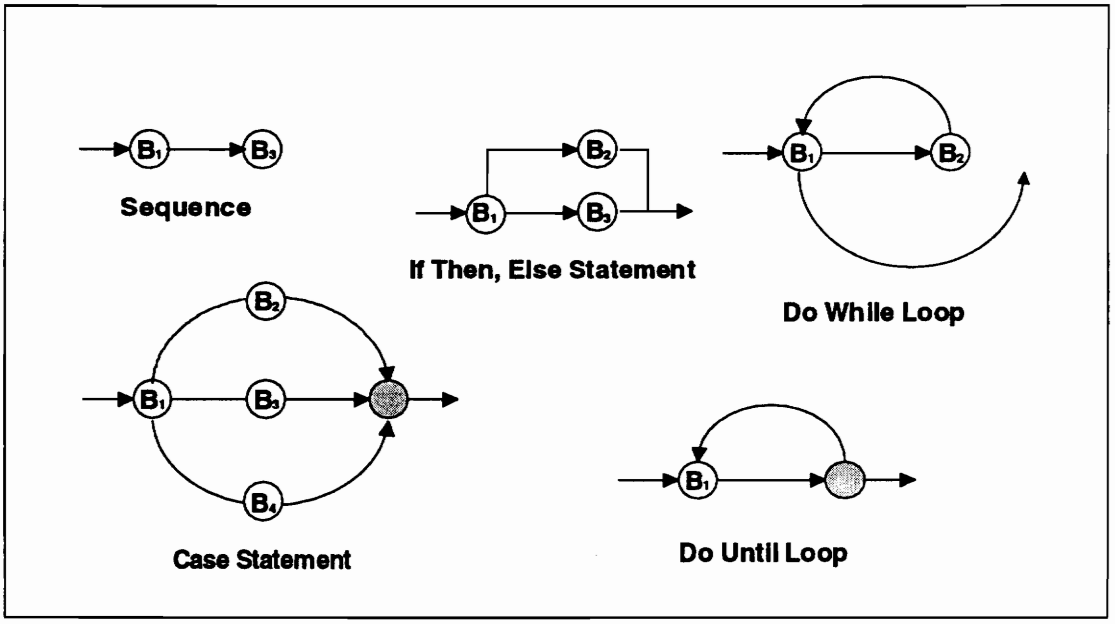

Figure 3. Structured Program Constructs

We can use the structured program constructs to develop a set of expressions for determining the program path length. The program path length can be estimated using the basic expressions provided in Table 1, below. The expressions incorporate path length as well as the probability of execution for each path, where p_i is the probability that the block will execute, l_i is the estimated block path length, and m is the estimated iteration count. Once the program path length is estimated, the program execution time, or delay, can be determined using the average execution time per instruction. Note that the probability that a given block will execute (p_i) is generally related to the value of the program input, while the estimated iteration count (m) is generally related to the size of the program input.

Table 1. Path Length and Probability of Execution Expressions

Construct	Path Length	Probability of Execution
Sequence B_1; B_2.	$p_1l_1 + p_2l_2$	$p_1 = 1; p_2 = 1$
If B_1, Then B_2, Else B_3.	$p_1l_1 + p_2l_2 + p_3l_3$	$p_1 = 1; p_2 + p_3 = 1$
While B_1, Do B_2.	$p_1l_1m + p_2l_2(m-1)$	$p_1 = 1; p_2 = 1$
Case B_1 of (B_2; B_3; ...; B_n).	$p_1l_1 + p_2l_2 + \cdots + p_nl_n$	$p_1 = 1; p_2 + \cdots + p_n = 1$
Do B_1 Until B_2.	$p_1l_1m + p_2l_2m$	$p_1 = 1; p_2 = 1$

2.3.2. Time Complexity Analysis

Knuth [5,6] analyzed the time complexity of algorithms. He showed that the execution time of a program is related to its complexity or "order" as well as to the size of the program input. His concepts can be applied to the determination of application program path length.

As previously noted, the estimated iteration count (m) is generally related to the size of the program input. Thus the estimated iteration count contributes to the program's execution time. More interesting is the question of the order of the program. Consider a structure consisting of a "Do Until" construct within a "Do Until" construct, commonly referred to as a nested loop. The program path length for the aggregate structure can be expressed as follows:

$$(p_1 l_1 m_1 + p_2 l_2 m_1)(p_3 l_3 m_2 + p_4 l_4 m_2), \text{ where } p_1 = 1; p_2 = 1; p_3 = 1; p_4 = 1$$

Substituting in the probability values provides:

$$(l_1 m_1 + l_2 m_1)(l_3 m_2 + l_4 m_2)$$

Variables m_1 and m_2 are related to the size of the input, while l_1, l_2, l_3, and l_4 are constant expressions of block length. Based on time complexity theory, the double nested loop example would have a *worst case time complexity* of *order* m^2.

2.3.3. Sensitivity Analysis

As part of his work, Mitchell performed sensitivity analysis for the end-to-end response time as a function of application program path length. He found that the relationship was non-linear, that is, response time rose at an increasing rate as application path length increased beyond a certain point.

3. DESIGN

3.1. Introduction

The design for end-to-end delay incorporates execution path analysis into the *N* stage queuing model. Two case studies have been performed: the first case study examines Mitchell's example host/workstation delay model and extends it to include an elaborate application submodel; the second case study examines a critical execution thread in NASA's Second TDRSS Ground Terminal. A spreadsheet program model has been developed and used to calculate the end-to-end delay for each case study incorporating the application program delay model.

3.2. Spreadsheet Models

The spreadsheet model actually consists of two models: an end-to-end path delay model and a host application path length model. The host application path length spreadsheet model is used to determine the total application path length by examining the application program structures and iterations. The result is then applied to the end-to-end path delay spreadsheet model which in turn calculates the following:

1. Host computer delay
2. Host/NIU interface delay
3. Host NIU delay
4. NIU/NIU delay
5. Workstation NIU delay
6. Workstation/NIU interface delay
7. Workstation delay

Any or all of the above delay components may be present. The spreadsheet models can be easily modified to match the specific system configuration. An additional feature of the spreadsheet

models is that they can be used for performance planning. For example, if a specific end-to-end delay value is required, spreadsheet parameters such as component delay, arrival rate, computer processor rating, and link data rate can be adjusted until the desired end-to-end delay value is achieved.

3.3. The Extended Host/Workstation Delay Model

The extended host/workstation delay model utilizes the single-thread example described in Section 2 as presented in Figure 2. Mitchell presented his calculated results for this example in [1]. Thus, the model serves as a useful case study because it can be used to verify the correct implementation and calibration of the spreadsheet tool. Details of the implementation and results are provided in Section 4.

3.4. The STGT Critical Thread Model

The STGT computer system controls and monitors NASA's Tracking and Data Relay Satellite System (TDRSS) [7]. The STGT critical thread model [8] examines the results of a simulation modeling effort performed for the STGT computer systems and local area network. During the STGT simulation effort, specific time-critical end-to-end single-thread tasks were analyzed to determine whether performance requirements could be met. In this paper, a spreadsheet model is presented which takes another look at a specific critical thread from the simulation effort. Details of the implementation and results are provided in Section 5.

4. THE HOST/WORKSTATION CASE STUDY

4.1. Overview

The extended host/workstation delay model utilizes the single-thread example described in Section 2 as presented in Figure 2. In this section, the spreadsheet model implementation and results as well as sensitivity analysis are presented. The end-to-end delay has been analyzed as a function of host application program execution path length. In addition the results have been compared to those previously obtained.

4.2. Spreadsheet Implementation and Results Using Mitchell's Example

Table 2 (generated by the spreadsheet and then copied to this document) provides the end-to-end delay for the host/terminal path in seconds calculated based on a 75,000 machine instruction host path length. The delay has been provided as a function of the arrival rate listed in the first column. All values in this table are calculated based on the spreadsheet model listed in Appendix A, Spreadsheet Model for the Host/Workstation Configuration. Figure 4 provides the comparative delay for 75,000, 150,000, and 225,000 machine instruction host path lengths. The 225,000 machine instruction host path length results in a processor utilization overload condition at approximately .15 arrivals per second.

The spreadsheet model in Appendix A is the actual listing from the computer spreadsheet model. It provides values for all performance parameters in the host/workstation path based on 75,000, 150,000, and 225,000 machine instruction host path lengths. Parameters include arrivals, multiprogramming level, path length for the application and the protocols, message length at each node and through each link.

4.3. Spreadsheet Implementation and Results with Optimized Host Application Program

Figure 5 presents the spreadsheet model used to determine the host application path length for an example application program. The model calculates total path length expressed in machine instructions for the application based on the block path length, block execution probability, and

Table 2. End-to-End System Delay for the Host/Workstation Path Based on 75,000-Instruction Host Application Path Length

Arrival Rate (msgs./ sec.)	Through-put S Normal-ized to Arrivals	Host Delay	Host/ NIU Interface Delay	Host NIU Delay	Ether-net Delay	Work-station NIU Delay	Work-station/ NIU Interface Delay	Work-station Delay	End-to-End System Delay
0.017	0.9823	0.0941	0.0172	0.0202	0.0120	0.0200	1.4551	0.2461	1.8647
0.034	1.9303	0.0984	0.0174	0.0204	0.0121	0.0200	1.4920	0.2474	1.9077
0.051	2.8458	0.1032	0.0175	0.0207	0.0121	0.0200	1.5308	0.2487	1.9531
0.068	3.7304	0.1085	0.0177	0.0209	0.0122	0.0200	1.5717	0.2501	2.0011
0.085	4.5858	0.1146	0.0179	0.0211	0.0122	0.0200	1.6149	0.2514	2.0521
0.102	5.4132	0.1214	0.0180	0.0214	0.0123	0.0200	1.6604	0.2528	2.1064
0.119	6.2141	0.1293	0.0182	0.0216	0.0123	0.0200	1.7087	0.2542	2.1643
0.136	6.9898	0.1383	0.0184	0.0219	0.0124	0.0201	1.7598	0.2556	2.2264
0.153	7.7413	0.1490	0.0186	0.0221	0.0124	0.0201	1.8141	0.2569	2.2932
0.170	8.4698	0.1616	0.0188	0.0224	0.0125	0.0201	1.8718	0.2584	2.3655
0.187	9.1763	0.1768	0.0190	0.0227	0.0125	0.0201	1.9333	0.2598	2.4442

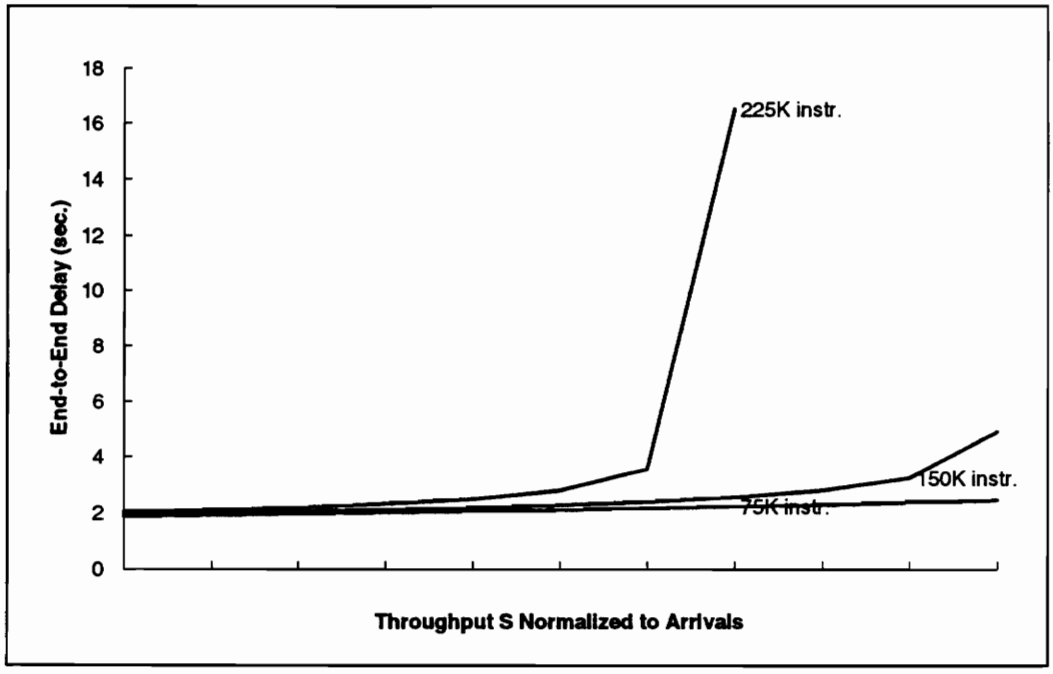

Figure 4. Host Application Path Length Sensitivity

Level One Structures	Level Two Structures	Level Three Structures	Level Four Structures	Level Five Structures
Sequence construct:				
Block B1	**Do-until construct, m=4**			
Path Length Exec. Prob.	**Block B1.1**	**Do-until construct, m=10**		
196,000 1	Path Length Exec. Prob.	**Block B1.1.1**	**Sequence construct**	
	49,000 1	Path Length Exec. Prob.	**Block B1.1.1.1**	
		4,900 1	Path Length Exec. Prob.	
			900 1	
			Block B1.1.1.2	**Do-until construct, m=5**
Block B2			Path Length Exec. Prob.	**Block B1.1.1.2.1**
Path Length Exec. Prob.			4,000 1	Path Length Exec. Prob.
10,000 1				800 1
Block B3	**If-then,else construct**			
Path Length Exec. Prob.	**Block B3.1**			
1,605 1	Path Length Exec. Prob.			
	100 1			
	Block B3.2	**Do-while construct, m=10**		
	Path Length Exec. Prob.	**Block B3.2.1**		
	3,500 0.3	Path Length Exec. Prob.		
		350 1		
	Block B3.3			
	Path Length Exec. Prob.			
	650 0.7			
Block B4	**Sequence construct:**			
Path Length Exec. Prob.	**Block B4.1**			
3,000 1	Path Length Exec. Prob.			
	2,200 1			
	Block B4.2			
	Path Length Exec. Prob.			
	800 1			
Case construct:				
Block B6				
Path Length Exec. Prob.				
300 0.2				
Block B7				
Path Length Exec. Prob.				
400 0.2				
Block B8	**Do-until construct, m=4**			
Path Length Exec. Prob.	**Block B8.1**	**Do-until construct, m=3**		
9,360 0.6	Path Length Exec. Prob.	**Block B8.1.1**		
	3,900 1	Path Length Exec. Prob.		
		1,300 1		
Sequence construct:				
Block B9				
Path Length Exec. Prob.				
3,000 1				

Application program size	23,600
Application path length	223,665

Figure 5. Spreadsheet Model for Host Application Path Length

execution relationship to other blocks. The model also calculates the application program size expressed in machine instructions. Notice that the example program utilizes the five structured program constructs described in Section 2. The example program has a total size of 23,600 machine instructions; however, the total path length is 223,665 machine instructions. This example illustrates the fact that program size is not indicative of program path length.

Figure 6 presents an optimized version of the same application program. In this version, the Block B1 path has been optimized. Specifically, the path length of Block B1.1.1.2.1 has been reduced from 800 machine instructions to 560 machine instructions. Also, the path length of Block B1.1.1.1 has been reduced from 900 machine instructions to 630 machine instructions. The resulting total program size is reduced from 23,600 machine instructions to 23,090 machine instructions, a 2.2% reduction. More significantly, the execution path length is reduced from 223,665 machine instructions to 164,865 machine instructions, a 26.3% reduction.

Table 3 provides the end-to-end delay for the host/terminal path based on the non-optimized host path length. The delay has been provided as a function of the arrival rate listed in the first column. All values in this table are calculated based on the spreadsheet model listed in Appendix B, Spreadsheet Model for the Host/Workstation Configuration with Optimized Host Application Program. Notice that for the three highest arrival rates, the host processor is overloaded.

Table 3. End-to-End System Delay for the Host/Workstation Path Based on Non-Optimized Host Application Program

Arrival Rate (msgs./ sec.)	Through-put S Normal-ized to Arrivals	Host Delay	Host/NIU Interface Delay	Host NIU Delay	Ethernet Delay	Work-station NIU Delay	Work-station/ NIU Interface Delay	Work-station Delay	End-to-End System Delay
0.017	0.9823	0.2541	0.0172	0.0202	0.0120	0.0200	1.4551	0.2461	2.0247
0.034	1.9303	0.2922	0.0174	0.0204	0.0121	0.0200	1.4920	0.2474	2.1015
0.051	2.8458	0.3450	0.0175	0.0207	0.0121	0.0200	1.5308	0.2487	2.1948
0.068	3.7304	0.4230	0.0177	0.0209	0.0122	0.0200	1.5717	0.2501	2.3156
0.085	4.5858	0.5502	0.0179	0.0211	0.0122	0.0200	1.6149	0.2514	2.4878
0.102	5.4132	0.7951	0.0180	0.0214	0.0123	0.0200	1.6604	0.2528	2.7800
0.119	6.2141	1.4617	0.0182	0.0216	0.0123	0.0200	1.7087	0.2542	3.4968
0.136	6.9898	10.4814	0.0184	0.0219	0.0124	0.0201	1.7598	0.2556	12.5695
0.153	host application utilization exceeds 100%								
0.170	host application utilization exceeds 100%								
0.187	host application utilization exceeds 100%								

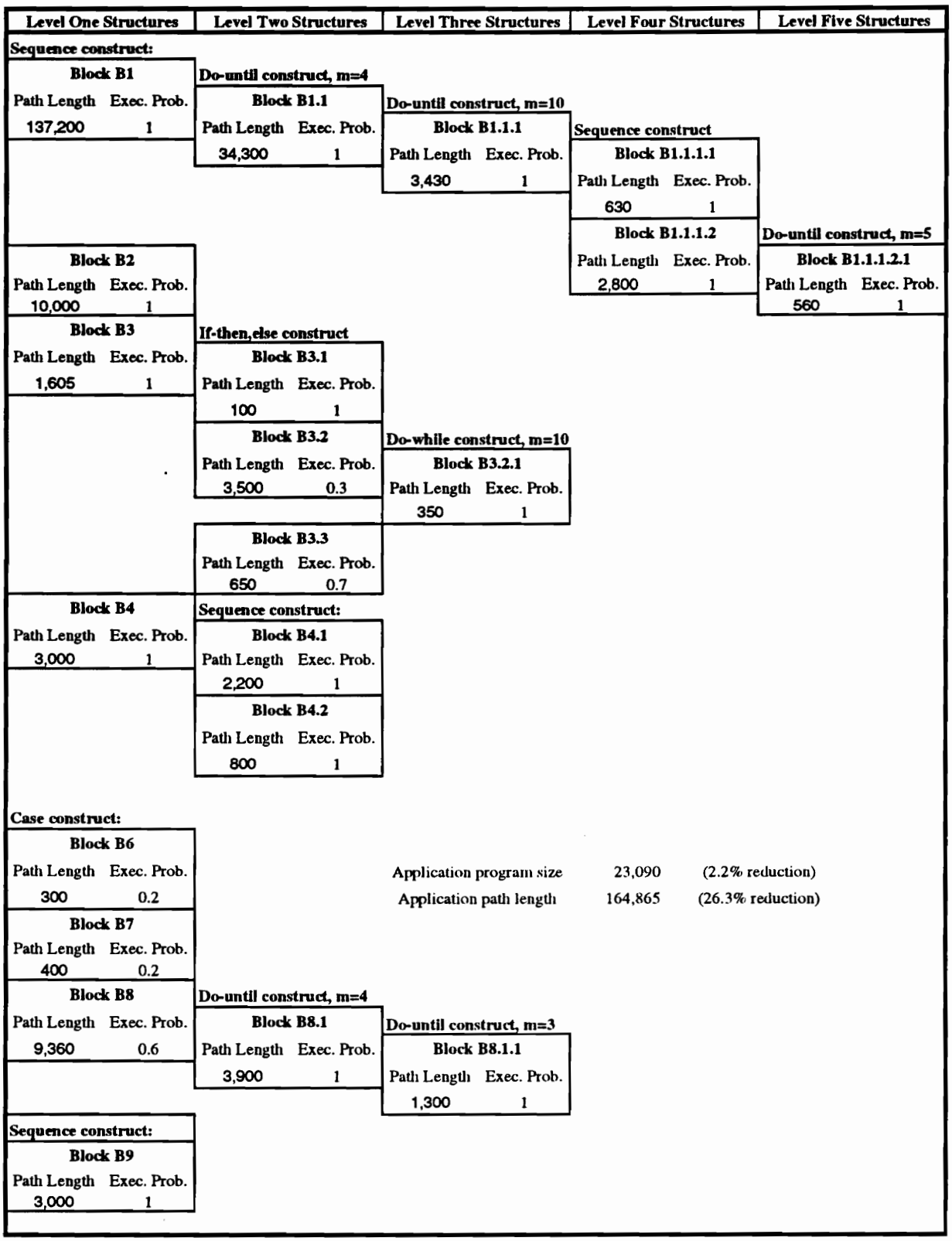

Figure 6. Spreadsheet Model for Optimized Host Application Path Length

Table 4. End-to-End System Delay for the Host/Workstation Path Based on Optimized Host Application Program

Arrival Rate (msgs./ sec.)	Through-put S Normal-ized to Arrivals	Host Delay	Host/NIU Interface Delay	Host NIU Delay	Ethernet Delay	Work-station NIU Delay	Work-station/ NIU Interface Delay	Work-station Delay	End-to-End System Delay
0.017	0.9823	0.1877	0.0172	0.0202	0.0120	0.0200	1.4551	0.2461	1.9583
0.034	1.9303	0.2072	0.0174	0.0204	0.0121	0.0200	1.4920	0.2474	2.0164
0.051	2.8458	0.2316	0.0175	0.0207	0.0121	0.0200	1.5308	0.2487	2.0815
0.068	3.7304	0.2633	0.0177	0.0209	0.0122	0.0200	1.5717	0.2501	2.1559
0.085	4.5858	0.3060	0.0179	0.0211	0.0122	0.0200	1.6149	0.2514	2.2436
0.102	5.4132	0.3668	0.0180	0.0214	0.0123	0.0200	1.6604	0.2528	2.3518
0.119	6.2141	0.4604	0.0182	0.0216	0.0123	0.0200	1.7087	0.2542	2.4955
0.136	6.9898	0.6231	0.0184	0.0219	0.0124	0.0201	1.7598	0.2556	2.7112
0.153	7.7413	0.9771	0.0186	0.0221	0.0124	0.0201	1.8141	0.2569	3.1214
0.170	8.4698	2.3450	0.0188	0.0224	0.0125	0.0201	1.8718	0.2584	4.5488
0.187	host application utilization exceeds 100%								

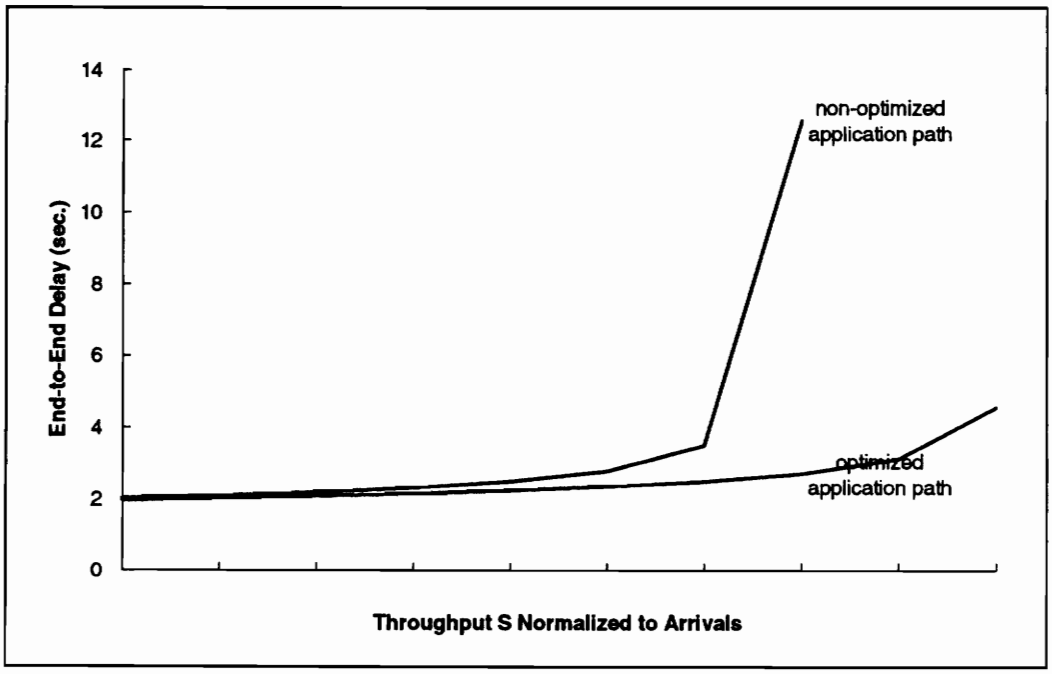

Figure 7. Host Application Path Length Sensitivity

The end-to-end delay model comparing non-optimized and optimized programs is provided in Appendix B, Spreadsheet Model for the Host/Workstation Configuration with Optimized Host Application Program. Notice that for the highest arrival rate, the host processor is overloaded. This is a significant improvement over the non-optimized application. The improvement, however, is more clearly seen in Figure 7 which compares the path length sensitivity for the non-optimized and optimized application paths.

5. THE STGT CASE STUDY

5.1. Overview

The STGT computer system controls and monitors the Tracking and Data Relay Satellite System (TDRSS) (see Appendix C). The STGT critical thread model examines the results of a simulation modeling effort performed for the STGT computer systems and local area network. During the STGT simulation effort, specific time-critical end-to-end single-thread tasks were analyzed to determine whether performance requirements could be met. In this section, a spreadsheet model implementation and results are presented which takes another look at a specific critical thread from the simulation effort. The results are analyzed including sensitivity analysis as a function of host application program execution path length.

5.2. Spreadsheet Implementation and Results with Reduced Measurand Processing

One of the time critical threads identified in the STGT Timing and Utilization Analysis document [8] is that of the User Service Subsystem (USS) computers receiving equipment status messages from the USS ground communications equipment and examining those messages to detect equipment problems and failures. Each equipment status message is composed of a large quantity of equipment status measurands, where each measurand is a binary equipment status code or an analog value such as voltage.

According to the STGT document, the current design for this activity was simulated by GE and found to greatly exceed the specified timing requirement of 1.0 second. The USS computers were in fact overloaded. The design was then revised to reduce the number of status measurands processed from 2,105 (all measurands processed) to 1,059. With reduced measurand processing, the GE simulation model still predicted that the specified timing requirement would be marginally exceeded (1.4 sec. vs. 1.0 sec.) in the worst case scenario.

Appendix D contains the spreadsheet model implementation and results for this critical thread. Results are given for the lowest arrival rate, 2 messages per second. In the spreadsheet and in Table 5, below, the results show that the 1 second requirement is exceeded at between 22 and 23

message arrivals per second with reduced measurand processing. The spreadsheet results also show that the USS computer is overloaded when processing all measurands.

It should be noted that the USS computer in question, the S-band multiple access USS computer, will, according to the design, receive 2 messages per second from the USS ground communications equipment. Two messages per second is the lowest arrival rate used in the spreadsheet model (see Table 5). Thus, the higher arrival rates account for other traffic flowing through the USS computer node. Also note in Table 5 that the bus delay is constant, regardless of the message arrival rate. According to the STGT Timing and Utilization Analysis document, this is due to the fact that (1) 2 messages per second are received via a MIL-STD-1553B bus from the USS ground communications equipment, and (2) the bus can easily handle this traffic without contention.

Table 5. End-to-End System Delay for the STGT Critical Thread with Reduced Measurand Processing

Arrival Rate (msgs./sec.)	Throughput S Normalized to Arrivals	Bus Delay	Host Delay	End-to-End System Delay
2.000	2.000	0.0678	0.0585	0.1263
15.000	15.000	0.0678	0.1294	0.1972
16.000	16.000	0.0678	0.1447	0.2125
17.000	17.000	0.0678	0.1647	0.2324
18.000	18.000	0.0678	0.1917	0.2595
19.000	19.000	0.0678	0.2304	0.2981
20.000	20.000	0.0678	0.2904	0.3581
21.000	21.000	0.0678	0.3961	0.4639
22.000	22.000	0.0678	0.6326	0.7003
23.000	23.000	0.0678	1.6340	1.7018

5.3. Spreadsheet Implementation and Results with Reduced Measurand Processing and Optimized Host Application Program

The spreadsheet model was subsequently revised to incorporate assumed optimizations in the application programs (see Appendix D). Table 6 shows the host application path parameters from the spreadsheet model using a two message per second arrival rate. In the Monitor_USS_-Ground_Equipment routine, 20 machine instructions are required to process each measurand. Thus, 42,100 machine instructions are required to process all measurands and 21,180 machine

instructions are required to process the reduced number measurands. In the Perform_Fault_-Detection_Limit_Checking routine, 150 machine instructions are required to process each measurand. Thus, 315,750 machine instructions are required to process all measurands and 158,850 machine instructions are required to process the reduced number measurands.

Finally, assumptions are applied concerning application optimization. The Monitor_USS-_Ground_Equipment routine is assumed to be optimized to 20,00 machine instructions. The Perform_Fault_Detection_Limit_Checking routine is assumed to be optimized to 135,000 machine instructions. The results are shown in Table 7.

Figure 8 presents the sensitivity analysis for the reduced measurand case as well as the reduced measurand/optimized application case. For the former case, performance deteriorates rapidly when arrivals exceed 21 per second. For the latter case, performance does not degrade noticeably, but remains relatively stable in the 2 to 23 messages per second range.

Table 6. USS Computer Host Application Path Parameters

Monitor_USS_Ground_Equipment:						
	path length to process all measurands				42,100	
	arrivals/sec	2	serv. time	0.0110789	delay	0.0115211
	path length to process reduced measurands				21,180	
	arrivals/sec	2	serv. time	0.0055737	delay	0.0057513
	path length to process reduced measurands with optimized application				20,000	
	arrivals/sec	2	serv. time	0.0052632	delay	0.0054297
Perform_Fault_Detection_Limit_Checking						
	path length to process all measurands				315,750	
	arrivals/sec	2	serv. time	0.0830921	delay	0.1012446
	path length to process reduced measurands				158,850	
	arrivals/sec	2	serv. time	0.0418026	delay	0.0463027
	path length to process reduced measurands with optimized application				135,000	
	arrivals/sec	2	serv. time	0.0355263	delay	0.0388183

Table 7. End-to-End System Delay for the STGT Critical Thread with Reduced Measurand Processing and an Optimized Application Program

Arrival Rate (msgs./sec.)	Throughput S Normalized to Arrivals	Bus Delay	Host Delay	End-to-End System Delay
2.000	2.000	0.0678	0.0507	0.1185
15.000	15.000	0.0678	0.0911	0.1589
16.000	16.000	0.0678	0.0979	0.1656
17.000	17.000	0.0678	0.1058	0.1736
18.000	18.000	0.0678	0.1154	0.1831
19.000	19.000	0.0678	0.1271	0.1949
20.000	20.000	0.0678	0.1418	0.2096
21.000	21.000	0.0678	0.1608	0.2286
22.000	22.000	0.0678	0.1864	0.2541
23.000	23.000	0.0678	0.2225	0.2903

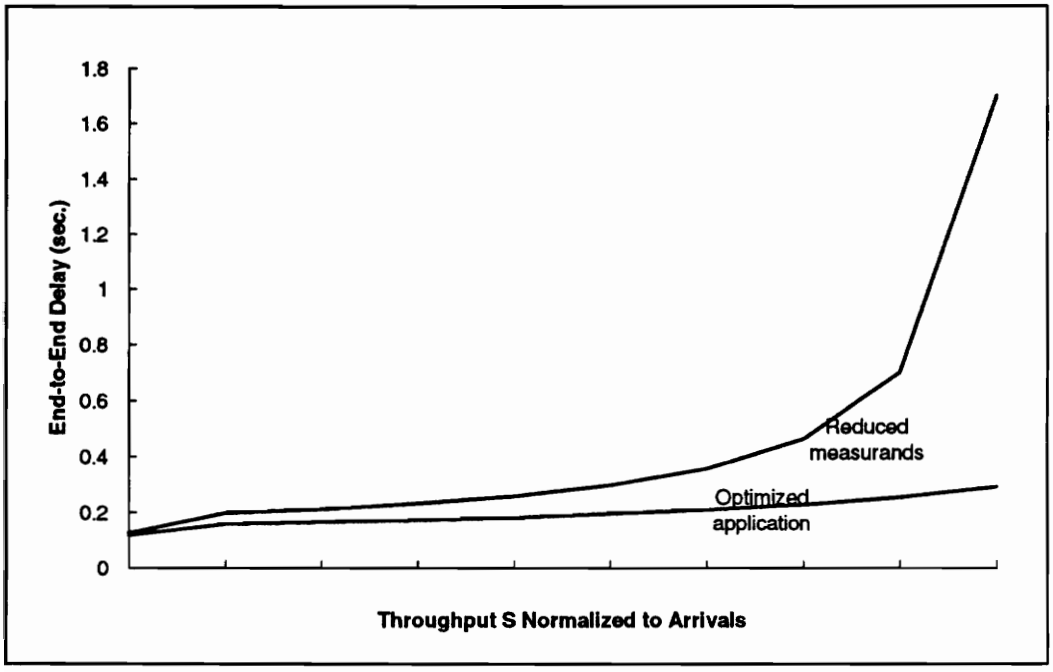

Figure 8. Host Application Path Length Sensitivity

6. CONCLUSION

The end-to-end performance for a single-thread transaction in a LAN-based computer system can be modeled as a series of independent queues and servers. The queueing model that is described in this paper represents a single-thread transaction traversing a computer LAN. Delay is modeled independently at each node and in each link using open loop queues based on Jackson's Theorem. In addition, delay due to other activity in the system is addressed by the analytical model.

This paper extends the previously developed model to include comprehensive modeling of the host application execution path. Such a model is useful where the host application software is a significant factor in end-to-end performance. This is often the case in general purpose computer systems such as that described in the host/workstation case study. It is also a concern in network performance and monitoring systems such as that described in the STGT case study.

As demonstrated in both case studies, the end-to-end delay may heavily depend on the host application path length. As shown in the host/workstation case study, a small reduction in program size (2.2%) may provide a significant reduction in path length (26.3%) and may reduce end-to-end delay from infinite (overloaded) to moderate.

The computer spreadsheet tool has proved very powerful and flexible in implementing the end-to-end delay model as well as the host application path model. With this tool, models were constructed to utilize numerous performance parameters in order to compute service time, delay, and utilization, locally and end-to-end. The spreadsheet tool was also used to chart and compare the sensitivity analysis for each case. Finally, the tool provided great flexibility in employing a trial and error method to establish the performance parameters, i.e., adjusting the arrival rate and processor MIPS rating to achieve a specific end-to-end delay value.

REFERENCES

[1] "A Methodology for Predicting End-to-End Responsiveness in a Local Area Network," *Proceedings, Computer Network Symposium,* 1981. L. Mitchell.

[2] "End-to-End Performance Modeling of Local Area Networks," *IEEE Journal on Selected Areas in Communications,* September 1986. L. Mitchell and D. Lide.

[3] *Local Networks.* W. Stallings.

[4] "A Complexity Measure," *IEEE Transactions on Software Engineering,* December 1976. T. McCabe.

[5] *Data Structures and Algorithms.* A. Aho, J. Hopcroft and J. Ullman.

[6] *The Turing Omnibus.* A. Dewdney.

[7] "NASA's Advanced Tracking and Data Relay Satellite System for the Years 2000 and Beyond," *Proceedings of the IEEE,* July 1990. D. Brandel, W. Watson and A. Weinberg.

[8] *STGT Timing and Utilization Analysis, Jan. 1991.* General Electric Corp. for NASA.

[9] *Quantitative Analysis of Computer Systems.* C. Leung.

[10] "Experiences in Implementing Solution Techniques for Networks of Queues," *Computer and Telecommunications Performance Engineering,* Edinburgh, 1991. D. Thomas.

[11] "End-to-End Delay Analysis on Local Area Networks: An Office Building Scenario," *Proceedings, National Telecommunications Conference,* 1981. B. Maglaris, T. Lissack, and M. Austin.

[12] "Job-shop like queueing systems," *Management Science,* Vol. 10, 1963. J. Jackson.

[13] *Analysis and Synthesis of Computer Systems.* E. Gelenbe and I. Mitrani.

APPENDIX A. SPREADSHEET MODEL FOR THE HOST/WORKSTATION PATH BASED ON MITCHELL'S EXAMPLE

System Delay

Total System Delay (Seconds)

75K instr. host application path	1.8646721
150K instr. host application path	1.942195
225K instr. host application path	2.0262769

Processor Capacities (MIPS)

host	1.1
NIU	0.615
workstat.	0.115

Traffic Parameters

arrival rate (msgs./sec)	0.017
aggregate load (bps)	100,000

Host Processor Delay

Host processor delay for each class of activity:

Host multi-programming level 32

$$t_{qj} = \frac{1}{1-\sum_{i=1}^{j-1}\rho_i}\left\{\frac{\sum_{i=1}^{j}\rho_i s_i}{1-\sum_{i=1}^{j}\rho_i} + s_j\right\}$$

Link In Parameters (priority class 1)

path length (instruct'ns) 75

arrivals/sec 0.544 serv. time 6.818E-05 delay 6.818E-05

Link Out Parameters (priority class 2)

path length (instruct'ns) 75

arrivals/sec 0.544 serv. time 6.818E-05 delay 6.819E-05

Protocols In Parameters (priority class 3)

path length (instruct'ns) 12,000

arrivals/sec 0.544 serv. time 0.0109091 delay 0.010975

Protocols Out Parameters (priority class 4)

path length (instruct'ns) 12,000

arrivals/sec 0.544 serv. time 0.0109091 delay 0.0111069

Application (Software Path) Parameters (priority class 5)

path length (instruct'ns) 75,000

arrivals/sec 0.544 serv. time 0.0681818 delay 0.0718353

path length (instruct'ns) 150,000

arrivals/sec 0.544 serv. time 0.1363636 delay 0.1493582

path length (instruct'ns) 225,000

arrivals/sec 0.544 serv. time 0.2045455 delay 0.2334401

Total Host Processor Delay (Seconds):

Assuming 75,000-Instruction Application Path Length	0.0940535
Assuming 150,000-Instruction Application Path Length	0.1715765
Assuming 225,000-Instruction Application Path Length	0.2556584

Host/NIU Interface Delay

interface delay: $t_{qj} = \rho\bar{s}/(1-\rho) + s_j$ where: $\rho = \lambda'\bar{s}$, $\lambda' = \lambda_1 + \lambda_2$, $\bar{s} = (\lambda_1 s_1 + \lambda_2 s_2)/\lambda'$

Parameters

utilization 0.0092695
mean service time 0.0085197
NIU-to-host (inbound) message arrivals/sec 0.544
host-to-NIU (outbound) message arrivals/sec 0.544
NIU-to-host (inbound) serv. time 0.00125
(assumes 100 char. msg. length, 800,000 bps transfer rate, 20% overhead factor)
host-to-NIU (outbound) serv. time 0.0157895
(assumes 1,500 char. msg. length, 800,000 bps transfer rate, 5% overhead factor)

NIU-to-Host (Inbound) Delay	0.0013297
Host-to-NIU (Outbound) Delay	0.0158692
Total Host/NIU Interface Delay (Seconds)	0.0171989

Host NIU Processor Delay

class of activity: $t_{qj} = \frac{1}{1-\sum_{i=1}^{j-1}\rho_i}\left\{\frac{\sum_{i=1}^{j}\rho_i s_i}{1-\sum_{i=1}^{j}\rho_i} + s_j\right\}$

Host multi-programming level 32

Network Link In Parameters (priority class 1)
path length (instruct'ns) 75
arrivals/sec 0.544 serv. time 0.000122 delay 0.000122

Node Link In Parameters (priority class 2)
path length (instruct'ns) 75
arrivals/sec 0.544 serv. time 0.000122 delay 0.000122

Network Link Out Parameters (priority class 3)
path length (instruct'ns) 75
arrivals/sec 0.544 serv. time 0.000122 delay 0.000122

Node Link Out Parameters (priority class 4)
path length (instruct'ns) 75
arrivals/sec 0.544 serv. time 0.000122 delay 0.000122

TCP/IP Parameters (priority class 5)
path length (instruct'ns) 12,000
arrivals/sec 0.544 serv. time 0.0195122 delay 0.0197269

Total Host NIU Processor Delay (Seconds)	0.0202148

NIU-NIU Link Delay (CSMA/CD)

Normalized throughput: $$S = \frac{Ge^{-aG}}{(1+a)Ge^{aG} + (1+aG)(1-e^{-aG})^2 + 1}$$

Average packet transmission delay: $$D = (A_1 + A_2 + A_3)T$$

$$A_1 = N_2(W+a)$$
$$A_2 = R_2(2^{N_2+1} - 1) + (N_1 - N_2)R_1$$
$$A_3 = a+1$$

$$W = \frac{1-e^{-aG}}{G} - ae^{-aG}$$
$$N_1 = G/S - 1$$
$$N_2 = (1+aG)e^{aG} - 1$$

Propagation time (microsec.)	30	Retransmission interval	5
Transfer rate (bps)	1,544,000		

Inbound NIU-NIU Link Delay

parameter a	0.0579	message length (chars.)	100
normalized offered packet traffic G (pkts/sec)		0.017	
normalized throughput S		1.67E-02	
average packet transmission delay D		0.0057807	

Outbound NIU-NIU Link Delay

parameter a	0.004825	message length (chars.)	1,200
normalized offered packet traffic G		0.017	
normalized throughput S		1.67E-02	
average packet transmission delay D		0.0062476	

Total NIU-NIU Link Delay (Seconds) 0.0120283

Workstation NIU Processor Delay

Workstation NIU processor delay for each class of activity:

$$t_{qj} = \frac{1}{1 - \sum_{i=1}^{j-1} \rho_i} \left\{ \frac{\sum_{i=1}^{j} \rho_i s_i}{1 - \sum_{i=1}^{j} \rho_i} + s_j \right\}$$

Network Link In Parameters (priority class 1)

path length (instruct'ns)	75				
arrivals/sec	0.017	serv. time	0.000122	delay	0.000122

Node Link In Parameters (priority class 2)

path length (instruct'ns)	75				
arrivals/sec	0.017	serv. time	0.000122	delay	0.000122

Network Link Out Parameters (priority class 3)

path length (instruct'ns) 75

arrivals/sec 0.017 serv. time 0.000122 delay 0.000122

Node Link Out Parameters (priority class 4)

path length (instruct'ns) 75

arrivals/sec 0.017 serv. time 0.000122 delay 0.000122

TCP/IP Parameters (priority class 5)

path length (instruct'ns) 12,000

arrivals/sec 0.017 serv. time 0.0195122 delay 0.0195188

Total Workstation NIU Processor Delay (Seconds) 0.0200066

Workstation/NIU Interface Delay

interface delay: $t_{qj} = \rho\bar{s}/(1-\rho) + s_j$ where: $\rho = \lambda'\bar{s}$, $\lambda' = \lambda_1 + \lambda_2$, $\bar{s} = (\lambda_1 s_1 + \lambda_2 s_2)/\lambda'$

Parameters

utilization 0.0241393

mean service time 0.7099781

workstation-to-NIU (inbound) message arrivals/sec 0.017

NIU-to-workstation (outbound) message arrivals/sec 0.017

workstation-to-NIU (inbound) serv. time 0.1041667

(assumes 100 char. msg. length, 9,600 bps transfer rate, 5% overhead factor)

NIU-to-workstation (outbound) serv. time 1.3157895

(assumes 1,500 char. msg. length, 9,600 bps transfer rate, 20% overhead factor)

NIU-to-Workstation (Inbound) Delay 1.3333518

Workstation-to-NIU (Outbound) Delay 0.1217289

Total Workstation/NIU Interface Delay (Seconds) 1.4550807

Workstation Delay

Workstation delay for each class of activity:

$$t_{qj} = \frac{1}{1-\sum_{i=1}^{j-1}\rho_i}\left\{\frac{\sum_{i=1}^{j}\rho_i s_i}{1-\sum_{i=1}^{j}\rho_i} + s_j\right\}$$

Link In Parameters (priority class 1)

path length (instruct'ns) 75

arrivals/sec 0.017 serv. time 0.0006522 delay 0.0006522

Protocols In Parameters (priority class 2)

path length (instruct'ns) 12,000

arrivals/sec 0.017 serv. time 0.1043478 delay 0.1045344

Application (Software Path) Parameters (priority class 3)

path length (instruct'ns)	4,000				
arrivals/sec	0.017	serv. time	0.0347826	delay	0.0350513
Link Out Parameters (priority class 4)					
path length (instruct'ns)	75				
arrivals/sec	0.017	serv. time	0.0006522	delay	0.0008604
Protocols Out Parameters (priority class 5)					
path length (instruct'ns)	12,000				
arrivals/sec	0.017	serv. time	0.1043478	delay	0.1049909
Total Workstation Delay (Seconds)					0.2460893

APPENDIX B. SPREADSHEET MODEL FOR THE HOST/WORKSTATION PATH WITH OPTIMIZED HOST APPLICATION PROGRAM

System Delay

Total System Delay (Seconds)

Non-optimized	2.0247181
Optimized	1.9583134

Processor Capacities (MIPS)

host	1.1
BIU	0.615
workstat.	0.115

Traffic Parameters

arrival rate (msgs./sec)	0.017
aggregate load (bps)	100,000

Host Processor Delay

Host processor delay for each class of activity:

$$t_{qj} = \frac{1}{1-\sum_{i=1}^{j-1}\rho_i}\left\{\frac{\sum_{i=1}^{j}\rho_i s_i}{1-\sum_{i=1}^{j}\rho_i} + s_j\right\}$$

Host multi-programming level 32

Link In Parameters (priority class 1)

path length (instruct'ns) 75

arrivals/sec 0.544 serv. time 6.818E-05 delay 6.818E-05

Link Out Parameters (priority class 2)

path length (instruct'ns) 75

arrivals/sec 0.544 serv. time 6.818E-05 delay 6.819E-05

Protocols In Parameters (priority class 3)

path length (instruct'ns) 12,000

arrivals/sec 0.544 serv. time 0.0109091 delay 0.010975

Protocols Out Parameters (priority class 4)

path length (instruct'ns) 12,000

arrivals/sec 0.544 serv. time 0.0109091 delay 0.0111069

Application (Software Path) Parameters (priority class 5)

path length (instruct'ns) 223,665

arrivals/sec 0.544 serv. time 0.2033318 delay 0.2318813

path length (instruct'ns) 164,865

arrivals/sec 0.544 serv. time 0.1498773 delay 0.1654765

Total Host Processor Delay (Seconds):

Assuming Non-optimized Application Path Length	0.2540996
Assuming Optimized Application Path Length	0.1876948

Host/NIU Interface Delay

interface delay: $t_{qj} = \rho\bar{s}/(1-\rho) + s_j$ where: $\rho = \lambda'\bar{s}$, $\lambda' = \lambda_1 + \lambda_2$, $\bar{s} = (\lambda_1 s_1 + \lambda_2 s_2)/\lambda'$

Parameters

utilization 0.0092695
mean service time 0.0085197
NIU-to-host (inbound) message arrivals/sec 0.544
host-to-NIU (outbound) message arrivals/sec 0.544
NIU-to-host (inbound) serv. time 0.00125
(assumes 100 char. msg. length, 800,000 bps transfer rate, 20% overhead factor)
host-to-NIU (outbound) serv. time 0.0157895
(assumes 1,500 char. msg. length, 800,000 bps transfer rate, 5% overhead factor)

NIU-to-Host (Inbound) Delay	0.0013297
Host-to-NIU (Outbound) Delay	0.0158692
Total Host/NIU Interface Delay (Seconds)	0.0171989

Host NIU Processor Delay

class of activity:

$$t_{qj} = \frac{1}{1-\sum_{i=1}^{j-1}\rho_i}\left\{\frac{\sum_{i=1}^{j}\rho_i s_i}{1-\sum_{i=1}^{j}\rho_i} + s_j\right\}$$

Host multi-programming level 32

Network Link In Parameters (priority class 1)
path length (instruct'ns) 75
arrivals/sec 0.544 serv. time 0.000122 delay 0.000122

Node Link In Parameters (priority class 2)
path length (instruct'ns) 75
arrivals/sec 0.544 serv. time 0.000122 delay 0.000122

Network Link Out Parameters (priority class 3)
path length (instruct'ns) 75
arrivals/sec 0.544 serv. time 0.000122 delay 0.000122

Node Link Out Parameters (priority class 4)
path length (instruct'ns) 75
arrivals/sec 0.544 serv. time 0.000122 delay 0.000122

TCP/IP Parameters (priority class 5)
path length (instruct'ns) 12,000
arrivals/sec 0.544 serv. time 0.0195122 delay 0.0197269

Total Host NIU Processor Delay (Seconds)	0.0202148

NIU-NIU Link Delay (CSMA/CD)

Normalized throughput: $$S = \frac{Ge^{-aG}}{(1+a)Ge^{aG} + (1+aG)(1-e^{-aG})^2 + 1}$$

Average packet transmission delay: $$D = (A_1 + A_2 + A_3)T$$

$$A_1 = N_2(W+a)$$
$$A_2 = R_2(2^{N_2+1} - 1) + (N_1 - N_2)R_1$$
$$A_3 = a+1$$

$$W = \frac{1-e^{-aG}}{G} - ae^{-aG}$$
$$N_1 = G/S - 1$$
$$N_2 = (1+aG)e^{aG} - 1$$

Propagation time (microsec.)	30	Retransmission interval	5
Transfer rate (bps)	1,544,000		

Inbound NIU-NIU Link Delay

parameter a	0.0579	message length (chars.)	100
normalized offered packet traffic G (pkts/sec)		0.017	
normalized throughput S		1.67E-02	
average packet transmission delay D		0.0057807	

Outbound NIU-NIU Link Delay

parameter a	0.004825	message length (chars.)	1,200
normalized offered packet traffic G		0.017	
normalized throughput S		1.67E-02	
average packet transmission delay D		0.0062476	

Total NIU-NIU Link Delay (Seconds) 0.0120283

Workstation NIU Processor Delay

Workstation NIU processor delay for each class of activity: $$t_{qj} = \frac{1}{1 - \sum_{i=1}^{j-1} \rho_i} \left\{ \frac{\sum_{i=1}^{j} \rho_i s_i}{1 - \sum_{i=1}^{j} \rho_i} + s_j \right\}$$

Network Link In Parameters (priority class 1)

path length (instruct'ns)	75				
arrivals/sec	0.017	serv. time	0.000122	delay	0.000122

Node Link In Parameters (priority class 2)

path length (instruct'ns)	75				
arrivals/sec	0.017	serv. time	0.000122	delay	0.000122

Network Link Out Parameters (priority class 3)

path length (instruct'ns)	75				
arrivals/sec	0.017	serv. time	0.000122	delay	0.000122

Node Link Out Parameters (priority class 4)

path length (instruct'ns) 75

arrivals/sec 0.017 serv. time 0.000122 delay 0.000122

TCP/IP Parameters (priority class 5)

path length (instruct'ns) 12,000

arrivals/sec 0.017 serv. time 0.0195122 delay 0.0195188

Total Workstation NIU Processor Delay (Seconds) 0.0200066

Workstation/NIU Interface Delay

interface delay: $t_{qj} = \rho\bar{s} / (1-\rho) + s_j$ where: $\rho = \lambda'\bar{s}$, $\lambda' = \lambda_1 + \lambda_2$, $\bar{s} = (\lambda_1 s_1 + \lambda_2 s_2) / \lambda'$

Parameters

utilization 0.0241393

mean service time 0.7099781

workstation-to-NIU (inbound) message arrivals/sec 0.017

NIU-to-workstation (outbound) message arrivals/sec 0.017

workstation-to-NIU (inbound) serv. time 0.1041667

(assumes 100 char. msg. length, 9,600 bps transfer rate, 5% overhead factor)

NIU-to-workstation (outbound) serv. time 1.3157895

(assumes 1,500 char. msg. length, 9,600 bps transfer rate, 20% overhead factor)

NIU-to-Workstation (Inbound) Delay 1.3333518

Workstation-to-NIU (Outbound) Delay 0.1217289

Total Workstation/NIU Interface Delay (Seconds) 1.4550807

Workstation Delay

Workstation delay for each class of activity:

$$t_{qj} = \frac{1}{1 - \sum_{i=1}^{j-1} \rho_i} \left\{ \frac{\sum_{i=1}^{j} \rho_i s_i}{1 - \sum_{i=1}^{j} \rho_i} + s_j \right\}$$

Link In Parameters (priority class 1)

path length (instruct'ns) 75

arrivals/sec 0.017 serv. time 0.0006522 delay 0.0006522

Protocols In Parameters (priority class 2)

path length (instruct'ns) 12,000

arrivals/sec 0.017 serv. time 0.1043478 delay 0.1045344

Application (Software Path) Parameters (priority class 3)

path length (instruct'ns) 4,000

arrivals/sec 0.017 serv. time 0.0347826 delay 0.0350513

Link Out Parameters (priority class 4)					
path length (instruct'ns)	75				
arrivals/sec	0.017	serv. time	0.0006522	delay	0.0008604
Protocols Out Parameters (priority class 5)					
path length (instruct'ns)	12,000				
arrivals/sec	0.017	serv. time	0.1043478	delay	0.1049909
Total Workstation Delay (Seconds)					0.2460893

APPENDIX C. THE STGT SYSTEM

Overview

NASA's Tracking and Data Relay Satellite System (TDRSS) is part of the NASA Space Network. TDRSS is currently composed of a constellation of three communication satellites and ground-based subsystems used to provide near-worldwide coverage to the Hubble space telescope, the polar orbiting platforms and other user spacecraft [8].

The ground-based subsystems are located at White Sands, NM at the White Sands Ground Terminal (WSGT). Another ground terminal at White Sands, the Second TDRSS Ground Terminal (STGT), is currently under development as part of the 1996 baseline system. It will be connected to the WSGT by an interfacility link. The WSGT and STGT will operate simultaneously as well as provide mutual backup support.

Tracking and Data Relay Satellite System (TDRSS)

TDRSS consists of two active Tracking and Data Relay Satellites (TDRS), one over the Pacific Ocean and one over the Atlantic Ocean, a centrally-located spare satellite and a ground site, the WSGT, at White Sands.

The WSGT provides a two-way communication link between user spacecraft and NASA facilities located in the United States. The WSGT also provides tracking, telemetry and control (TT&C) of the three TDRSs. Ground communication links via the NASA communication link (NASCOM) are provided to payload operations control centers (POCCs), to the Network Control Center (NCC) at Goddard Space Flight Center (GSFC) and to the Flight Dynamics Facility (FDF).

A 1996 TDRSS baseline architecture is currently under development to meet expanding user needs. The 1996 baseline includes:

1. Addition of a second ground terminal at White Sands, the STGT.

2. Upgrade of the WSGT.

3. Two additional TDRSs for a total of four active and one spare TDRSS spacecraft.

The 1996 baseline expansion will provide the following level of telecommunications service:

1. S-band multiple access (SMA). Twenty SMA return link channels are provided from user spacecraft to the four combined TDRS. Each signal is converted to a distinct Ku-band frequency at the TDRS. Each Ku-band single access (KuSA) signal is then relayed to the WSGT and the STGT. The maximum data rate per channel is 50 Kbps. One SMA forward link per TDRS is also provided with a maximum data rate of 10 Kbps.

2. S-band single access (SSA). Twelve SSA forward and 12 SSA return link channels are provided between user spacecraft and the two ground terminals via the four combined TDRS. The maximum data rate per forward channel is 300 Kbps. The maximum data rate per return channel is 6 Mbps.

3. Ku-band single access (KuSA). Twelve KuSA forward and 12 KuSA return link channels are provided between user spacecraft and the two ground terminals via the four combined TDRS. The maximum data rate per forward channel is 25 Mbps. The maximum data rate per return channel is 300 Mbps.

The combined STGT and upgraded WSGT will include:

1. Three space-ground link terminals (SGLTs) at each ground terminal, each capable of supporting a TDRS. The third SGLT will not support SMA service.

2. A standalone S-band TT&C terminal at each ground terminal for backup TT&C support.

3. Interconnection by an interfacility link between the STGT and the upgraded WSGT for transfer of user data and transfer of control, thus providing for mutual backup support.

Second TDRSS Ground Terminal (STGT)

The STGT in Figure 9 will be constructed as part of the 1996 TDRSS baseline expansion. It will provide a backup for the upgraded WSGT and meet the expanding telecommunications needs of NASA's space programs. As with the WSGT, the STGT will control operational TDRSS spacecraft and provide communication and tracking links between ground control facilities at various locations and user spacecraft.

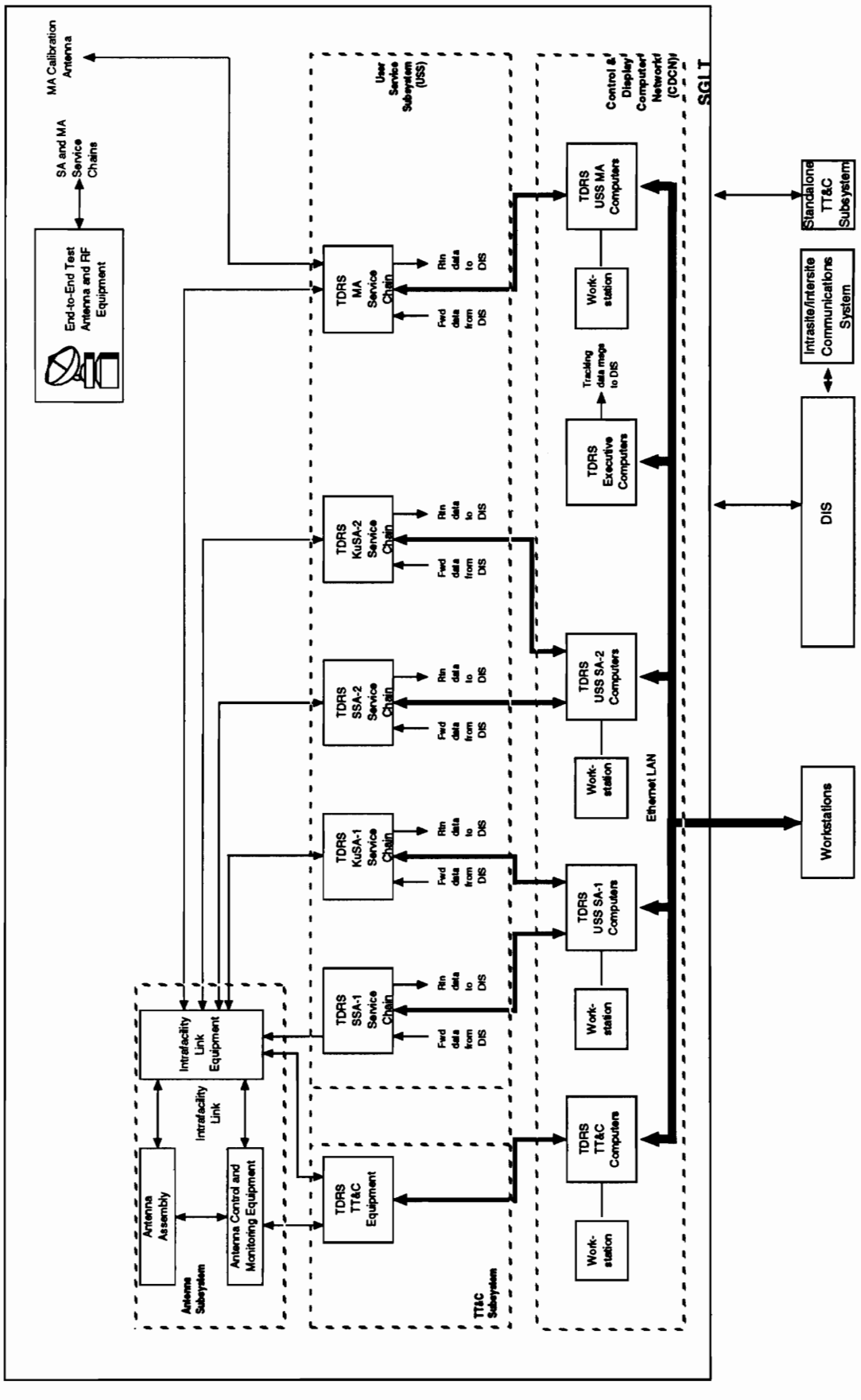

Figure 9. SGLT Architecture

The STGT will contain three of the SGLTs detailed in Figure 9. Each SGLT contains a complete complement of communications and computer equipment to support two-way Ku- and S-band satellite communications. As indicated in Figure 9, the Data Interface Subsystem equipment and the standalone TT&C equipment is shared by the three SGLTs.

The SGLT equipment possesses the following major design characteristics:

1. All communications and computer components are redundant and designed for automatic rapid failover capability.

2. All computers are connected by an ethernet LAN; in addition, the three SGLTs are connected by a backbone LAN.

3. The S-band services, Ku-band services and TT&C support are each provided using independent strings of communications/computer equipment.

4. The communications equipment in each string is controlled as the result of commands sent from a computer via a MIL-STD-1553B bus to a subsystem controller and then to the communications equipment.

5. The equipment can be controlled and monitored by any of several workstations that are attached to the SGLT LAN or by other workstations that are attached directly to the computers.

SGLT Computer System

The SGLT computer system shown in Figure 9 consists of five dual VAX 8600 computers and several VAX 3100 workstations connected by an ethernet LAN. In addition, the Data Interface Subsystem computers and the S-band TT&C computers are shared by the three SGLTs.

Six primary software components are present at each SGLT:

1. TT&C software component. Controls TT&C communications equipment. Accepts telemetry data from and sends commands to TDRS spacecraft and user spacecraft. Installed on TT&C computers.

2. User services software component. Controls S-band and Ku-band communications equipment. Installed on SA-1, SA2 and MA computers.

3. Workstation software component. Provides human-computer interface screens for all other software components. Provides data entry, update and query, as well as up to four active windows on the screen and varying levels of access to different users. Installed on all workstations.

4. Data interface subsystem software component. Controls data interface subsystem communications switches. Installed on the data interface subsystem computers.

5. Executive software component. TBD. Installed on the executive computers.

6. COM software component. Provides operating system, intercomputer communications and failover capability. Installed on all computers.

STGT Computer System Performance Model

The STGT developer performed a computer network timing and utilization analysis [8] that included computer prototyping and simulation. The goals were to identify the performance problems and risk areas and to identify the performance drivers for the problem areas.

The analysis dealt with the questions of how efficiently system resources would be utilized as well as predicting the response time and turnaround time.

Performance Methodology

The STGT timing and utilization analysis methodology was as follows:

1. Identify the performance requirements as extracted or derived from the STGT system requirements. The requirements were allocated to system components, i.e., disk device, CPU, software routine, etc.

2. Focus the analysis on the worst case scenario. The performance requirements were analyzed in the context of the worst case scenario, that is, the situation involving peak resource utilization. Fourteen specific application activities with critical timing requirements (e.g., maximum time allowed to process a forward link EIRP reconfiguration

request was 10 seconds) were analyzed with regard to timing. In addition, four elements with peak utilization limit requirements (e.g., peak allowable CPU utilization was 50%), CPU, memory, disk and LAN, were analyzed with regard to resource utilization.

3. Develop and execute a simulation model. Simulations were executed using the Performance Analysis Workbench System (PAWS) to predict the system's ability to comply with timing and utilization requirements.

4. Execute prototype benchmarks using actual equipment. The actual resource utilization information from these runs was used to refine the assigned resource utilizations in the simulation model.

5. Verify ability of implemented system to meet performance requirements. The implemented system was verified using actual system test results.

Results

Following the timing and utilization analysis, the simulations indicated that:

1. Most of the computers came close to or exceeded the 50% utilization requirement. Two were saturated. As a result, NASA ordered larger computers to solve the problem.

2. All other resource utilization requirements were met.

3. Critical timing requirements were met for all except one of the fourteen specific application activities. The timing requirement for S-band multiple access status data processing is 1 second. This requirement was sometimes exceeded during simulation. Steps were being taken to solve this problem.

In general the STGT computer network simulation and prototyping appears to have helped in meeting system performance objectives. The final outcome is not clear as the STGT is not yet operational.

APPENDIX D. SPREADSHEET MODEL FOR THE STGT CRITICAL THREAD

System Delay

Total System Delay (Seconds)	all measurands (overloaded)		0.2539223
	reduced measurands processed		0.1262666
	reduced measurands/optimized application path		0.1184606
Processor Capacities (MIPS)			
DEC VAX 6310	3.8		
NIU	0.615		
Workstation	0.115	arrival rate (msgs./sec)	2

Subsystem Controller Bus Delay (MIL-STD-1553 Bus)

Transfer rate (bps) 1.00E+06

Inbound Bus Delay

all measurands	2105	resulting message length (chars.)	16,840
reduced measurands	1059	resulting message length (chars.)	8,472
average transmission delay D (all measurands)			0.13472
average transmission delay D (reduced measurands)			0.067776

Total Bus Delay (Seconds)	all measurands	0.13472
	reduced measurands	0.067776

Host Processor Delay

Host processor delay for each class of activity:

Host multi-programming level 1

$$t_{qj} = \frac{1}{1-\sum_{i=1}^{j-1}\rho_i}\left\{\frac{\sum_{i=1}^{j}\rho_i s_i}{1-\sum_{i=1}^{j}\rho_i} + s_j\right\}$$

Link In Parameters (priority class 1)

path length (instruct'ns)	75				
arrivals/sec	2	serv. time	1.974E-05	delay	1.974E-05

Link Out Parameters (priority class 2)

path length (instruct'ns)	75				
arrivals/sec	2	serv. time	1.974E-05	delay	1.974E-05

Protocols In Parameters (priority class 3)

path length (instruct'ns)	12,000				
arrivals/sec	2	serv. time	0.0031579	delay	0.0031782

Protocols Out Parameters (priority class 4)

path length (instruct'ns)	12,000				
arrivals/sec	2	serv. time	0.0031579	delay	0.0032189

Application (Software Path) Parameters (priority class 5)

Monitor_USS_Ground_Equipment:					
path length to process all measurands				42,100	
arrivals/sec	2	serv. time	0.0110789	delay	0.0115211
path length to process reduced measurands				21,180	
arrivals/sec	2	serv. time	0.0055737	delay	0.0057513
path length to process reduced measurands with optimized application				20,000	
arrivals/sec	2	serv. time	0.0052632	delay	0.0054297
Perform_Fault_Detection_Limit_Checking					
path length to process all measurands				315,750	
arrivals/sec	2	serv. time	0.0830921	delay	0.1012446
path length to process reduced measurands				158,850	
arrivals/sec	2	serv. time	0.0418026	delay	0.0463027
path length to process reduced measurands with optimized application				135,000	
arrivals/sec	2	serv. time	0.0355263	delay	0.0388183

Total Host Delay (Seconds)	all measurands (overloaded)	0.1192023
	reduced measurands processed	0.0584906
	reduced measurands/optimized application path	0.0506846

VITA

JOHN J. DEEDS

ACADEMIC BACKGROUND

MS candidate, Electrical Engineering (Communications), Virginia Tech
MA, Computer Science, University of Detroit, College of Engineering & Science
BS, Mathematics, University of Michigan

SUMMARY OF PROFESSIONAL EXPERIENCE

Background includes 15 years of systems and software engineering including systems software, telecommunication networks, data base and MIS software. Currently providing systems engineering support to NASA for the Space Network (SN) Network Control Center and the Tracking and Data Relay Satellite System (TDRSS). Research interests include computer system performance analysis and software engineering methods and tools.

Stanford Telecommunications, Inc., Jan. 1991 - Present

Perform TDRS II ground segment systems engineering for NASA project office. Provide system specification, architecture, analyses, models, SOW and CDRL. Specific tasks:

- Computer network specifications and architecture definition, high-level design and loading analysis
- Simulation modeling of the Network Control Center
- Evaluation and recommendation of software engineering methods and tools
- Software/firmware specifications and standards definition
- Life cycle cost analysis for TDRSS ground communications and computer systems

SofTech, Inc., Jan. 1983 - Jan. 1991

Software quality assurance manager for DoD special-purpose operating systems, simulators, Ada compilers, verification suites, software reusability tools and test language software. Specific tasks:

- Evaluated DoD-STD-2167A and MIL-STD-1521B software products
- Planned/prepared for PDRs, CDRs, FQTs, FCAs and PCAs
- Researched IEEE software standards, SEI quidelines, CASE tools, object-oriented methods and software metrics
- Implemented Oracle-based review log and action item tools

Engineer for real-time avionics operating system (80,000 Assembly source lines) for Navy AN/UYK-14 processor. Authored parts of design specification, program design language (PDL), test plan and procedures.

JOHN J. DEEDS

Project leader for enhancement of data management language translator/graphics access method for General Motors' CAD/CAM package (6 million PL/1 source lines) for IBM 30XX MVS/XA computers.

Developed 15 worldwide turnkey CAD/CAM systems for General Motors based on IBM 4381 MVS/XA computers. Specific tasks:

- Defined initial baseline in conjunction with users, developers and vendors
- Provided computer operations tools, documentation and training
- Performed integration, test, release scheduling, trouble shooting of communications interfaces, graphics displays, plotters and software

Coastal Corp., Nov. 1980 - Dec. 1982

Specified MIS requirements for $2 billion new technology coal gasification facility. For payroll/personnel system, defined user requirements and design, lead the implementation and test. Redesigned, implemented and tested software for employee savings plan.

NBD Bancorp, Aug. 1978 - Nov. 1980

Implemented investment portfolio analysis software using FORTRAN with statistical/graphics capability. Maintained/enhanced online bank accounting system (1 million source lines) used to manage $15 billion in assets.

Navy Data Automation Command, US Government, Oct. 1976 - Aug. 1978

Installed compiler releases and patches. Developed and documented system utilities.