12-11-1984

An Analysis of the Effect of Network Load and Topology on the Performance of Concurrency Control Algorithm in Distributed Database Systems

Anoop Singhal

Amit P. Sheth
Wright State University - Main Campus, amit.sheth@wright.edu

Ming T. Liu

AN ANALYSIS OF THE EFFECT OF NETWORK LOAD AND TOPOLOGY ON THE PERFORMANCE OF A CONCURRENCY CONTROL ALGORITHM IN DISTRIBUTED DATABASE SYSTEMS*

ANOOP SINGHAL, AMIT P. SHETH AND MING T. LIU
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE
THE OHIO STATE UNIVERSITY
COLUMBUS, OH 43210

ABSTRACT

Earlier studies in the performance analysis of concurrency control algorithms in distributed database systems (DDBS) have assumed that the message transmission time between any two nodes of a network is constant. Hence they disregard the effect of communication subnet related parameters such as network traffic, network topology, the capacity of transmission channels and the size of messages transmitted. In this paper an analytical model is used to estimate the delays in transmission channels for a DDBS in a long haul network. The analysis shows that the constant transmission time assumption cannot be justified in many cases and that response time is sensitive to the parameters mentioned above. Conditions under which the DDBS is node bound or network bound are also derived.

Key words and phrases : distributed databases, concurrency control, long haul networks, response time, utilization, two phase commit, queueing theory

1 INTRODUCTION

A distributed database system (DDBS) is a database system whose physical copies of data (often redundant) are distributed on a computer network. A DDBS has the potential of offering improved throughput, better sharing and modular expansion, as well as improved reliability and response time as compared to its centralized counterpart. However, concurrent execution of transactions in a DDBS can result in an inconsistent database state. To prevent this from happening, the execution of transactions should be synchronized so that all the database integrity constraints are satisfied. Concurrency control algorithms (CCAs) achieve this effect by guaranteeing that the visible states of the database are exactly those which are reached if all the transactions are executed sequentially. This property, called serializability, has been fundamental to most of the work in this area[1, 2, 3, 4].

While many CCAs have been proposed, there has been relatively little work that describes the performance of these algorithms, compares them in similar environments or studies their performance under different database and network environments. It is the last point that we are primarily interested in. Some interesting and important studies include[5,6] and[7]. Performance of three CCAs for DDBS has been carried out by Garcia-Molina[5] using queueing analysis as well as simulation with such assumptions as fully replicated databases, update only transactions, no crashes, constant transmission time between any two nodes, light communication load, and a low degree of conflict among transactions. Dantas[6] carried queueing analysis of Thomas' majority consensus algorithm[8] and its variance under a high degree of conflict among transaction. Cheng's study[7] also considered the effect of system failures.

However, all of the aforementioned analytical studies have ignored the effect of communication subnet related parameters such as network traffic, network topology, and capacity of transmission channels by assuming a constant transmission time between any two nodes. One exception to this is a recent paper by Galationas[9] which relaxes this assumption but only in the context of local area networks with broadcast capability. In this paper we are primarily concerned with a DDBS over a long haul network. We feel that in such an environment it is important to study the queueing effect in the transmission channels because of the following reasons:

1. Transmission channels (TCs) are (in the context of traditional DDBS over long haul network) much slower than CPUs and storage devices. For example, ARPANET can move data at about 50 kbps while a standard disk can move data at the rate of 1 Mbps.

* Research reported herein was supported by US Army CECOM, Ft. Monmouth, N.J., under Contract No. DAAB07-83-K-K542. The views, opinions, and/or findings contained in this paper are those of the authors and should not be construed as an official Department of the Army position, policy or decision.

2. Failures of TCs may change the network topology temporarily and load some channels heavily[10].

3. Changes in network traffic with time may vary queueing delays for concurrency control (CC) messages.

4. Some TCs may be more heavily loaded than others due to characteristics of the CCA and network topology[10].

A model that includes the queueing delays in the communication subnet can help identify DDBS performance bottleneck and consequently determine the steps to be taken to improve transaction throughput and response time. For example, if the system is network bound, then CCA performance can be improved by increasing capacities of overloaded lines, by changing routing strategy, or by changing to a CCA that decreases the number of CC related messages. Such a model can also identify the relationships among DDBS throughput, changes in network traffic, different CCAs and various network topologies.

The primary objective of this paper is to analyze the effect of network load and topology on the performance of any CCA. As most commercial DDBS use the centralized locking algorithm[11], we shall analyze the effect of network load and topology on the performance of a centralized locking algorithm that uses two phase commit.

In the next section, we present the two phase commit protocol. Section 3 discusses the performance evaluation model of a DDBS node. In section 4 a performance evaluation methodology to analyze the effect of network load and topology is presented. Section 5 discusses the performance evaluation results.

2 TWO PHASE COMMIT PROTOCOL

In this section the two phase commit (2-PC) protocol[2, 12] is discussed in brief.

Suppose a transaction is submitted to site i of a DDBS. Then site i acts as the coordinator for the execution of transaction; all other sites are considered as subordinates. The various steps corresponding to the processing of a transaction under 2-PC are as follows:

Step 1: The coordinator sends a 'request locks' message to the central controller.

Step 2: The central controller checks for requested locks. When all the requested locks are available, it sends a 'locks granted' message to the coordinator.

Step 3: The coordinator computes the updates and sends a 'precommit' message to all the subordinates.

Step 4: Each subordinate site that is willing to let the transaction commit writes a 'prepare log' record on the stable storage and then sends a 'Yes' vote to the coordinator. A 'No Vote' is sent if the subordinate wants the transaction to be aborted.

Step 5: After the coordinator receives votes from all its subordinates, it initiates the second phase of the protocol. If all the subordinates voted 'Yes' then the coordinator updates its own database and sends a 'Commit' message to all the subordinates. Otherwise it writes an 'abort record' on the stable storage and sends 'Abort' messages to all the subordinates.

Step 6: Upon receiving a 'Commit' message, each subordinate site updates its database. Then it sends an 'Ack' message to the coordinator. On receiving an 'Abort' message the subordinate aborts the transaction (by writing an abort record on the stable storage) and then sends an 'Ack' message to the coordinator.

Step 7: After the coordinator receives all 'Ack' messages, it sends a 'release locks' message to the central controller.

3 THE PERFORMANCE EVALUATION MODEL

In this section a performance evaluation model called the network server model is presented. The assumptions made, performance measures, and the two options of 2-PC implementation are also discussed.

3.1 The Network Server Model

Figure 1 shows the model used to represent the DDBS at each node. Requests for service arrive at a node from three sources: the users at the node, the network and the node itself. The I/O, CPU and network servers at each node service at most one request at a time. FIFO queues are provided for requests waiting at each server. In order to illustrate the operation of the model, we describe the processing of a lock request. Consider a lock request that arrives at the central node from node A. In order to read the lock table, the transaction first obtains service from the I/O server. The CPU

server then checks the availability of the locks. When the locks are available, proper entries are made into the lock table and the transaction receives service from the I/O server to write the updated lock table on the disk. Finally, the network server services the transaction by transmitting the lock grant message to node A over a TC.

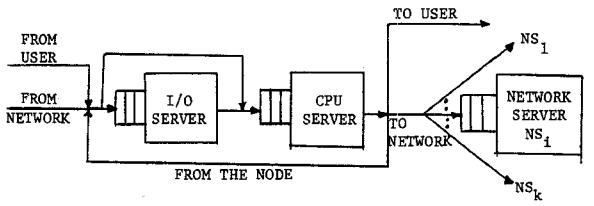

Figure 1: The Performance Evaluation Model

Models used in the earlier performance studies did not include the network server because of the assumption of constant transmission time between any two nodes. This assumption is justifiable only if all the TCs are lightly loaded. But because of low service rates of TCs in the long haul networks and other factors noted earlier, the transmission time may be comparable to or larger than other factors comprising the transaction response time. The network server helps us take care of this. Note that the model still leaves out details such as multiprocessing at nodes. A more complex model may make an analytical solution too difficult or intractable. The remainder of this section gives the parameters used in the study and the assumptions made.

3.2 PARAMETERS AND ASSUMPTIONS

1. Mean Interarrival Time: It is assumed that the interarrival time of transactions is exponentially distributed with mean $1/\lambda$.

2. Mean Base Set: It is assumed that the number of items in the base set has an exponential distribution with mean $1/\mu$. The size of the update set is assumed to be the same as that of the base set.

3. I/O Time Slice, I: This is the time it takes to read or write a lock or an item.

4. The size of an item, S.

5. The block size, B.

6. The number of nodes, N.

7. The Channel Capacity, C: In order to simplify the analysis, we assume that every TC connecting a pair of nodes is of capacity C.

8. Database Traffic: There are three classes of messages:

   a. Short Messages: Examples of short messages are lock grant, lock release, vote, and ack messages, with mean length $1/\mu_1$.

   b. Medium Length Messages: Example of a medium length message is a lock request message, with mean length $1/\mu_2$. It is longer than say an ack message because it includes the addresses of the base set items in addition to the transaction id and message type information.

   c. Long Messages: An example of a long message is a precommit message with mean length $1/\mu_3$. A precommit is a long message because it contains the updated records.

   The length of a class 2 or a class 3 message is exponentially distributed since it is proportional to the size of the base set of a transaction. Although in practice the length of class 1 messages is constant, less than ten percent of total traffic comprises of such messages in the worst case. In order to simplify the analysis we assume that the length of class 1 messages is also exponentially distributed. Two options for the algorithm implementation are possible which decide the size of the class 3 messages and whether a commit message is of class 1 or class 3. These variations, discussed in the next subsection, have a major effect on the performance of the algorithm.

9. The number of items in the database, M.

10. The database is assumed to be fully replicated and failure free to simplify the analysis.

11. Queueing delays at the CPUs are neglected as they work much faster than the I/O devices and the TCs.

3.3 PERFORMANCE MEASURES

In this paper, the performance of DDBS is made in terms of the following measures:

1. Execution response time: It is the time elapsed since transaction submission to the earliest instant when the system can guarantee the user that the transaction execution will successfully complete. In case of the 2-PC, this time equals to the interval starting from the transaction submission to the end of first phase of the two phase commit.

2. Node utilization: It is the fraction of time the I/O server at the node is busy.

3. Channel utilization: It is the fraction of time the TC server is busy.

3.4 TWO OPTIONS

While studying details of the 2-PC, two options for its implementation seem viable and interesting. Let us assume the worst case placement of items in the blocks (e.g. see[13]). For example, if items numbered 11,33 and 47 are required by a transaction, they may be distributed among three blocks as shown in figure 2.

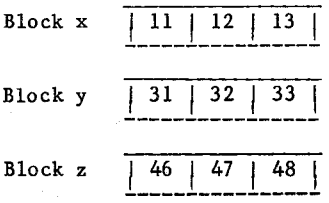

Figure 2: Worst Case Placement of Items in Blocks

While reading an item, a complete block that contains the required item will have to be read into memory from the disk. Two options when creating the precommit message for this transaction seem to be interesting. In the **first option**, the precommit message consists of all the three blocks with updated values of items 11,33 and 47. In the **second option**, the precommit message includes only the items that are updated and discards the rest. The trade off becomes clear when the second phase of the 2-PC is considered. In the second option, the commit message could be very short (as other class 1 messages), but a receiver node will have to first read in the three blocks that contain items 11,33 and 47, read the precommit record ('prepare log' record created in the step 4), update the blocks and then write them. In the first option, the blocks with the updated items are again sent along with the commit message. Hence, the receiver will only need to write these updated blocks (assuming that the block structure at both the sender and the receiver nodes is identical). Note that in this case, the commit message will be as long as a precommit message. Two points are worth noting: Firstly, even though the precommit messages in both options will be class 3 messages, the precommit messages (and also the commit messages) in the first option will be larger by a factor equal to the number of items in a block than those in the second option. Secondly, even if we neglect the time to read a precommit record (usually it will not be longer than one block), there will be one additional read in the second option. Thus if the disk utilization is high, the first option seems preferable, and if the system is network bound, the second option seems preferable. In the following sections, the first option is assumed for deriving the channel utilization and the disk utilization. Some performance results are also given for the second option.

4 PERFORMANCE EVALUATION METHODOLOGY

As explained in[14] the interconnection of all nodes forms a network of queues. The analysis of such a network is very complex for the general case. In this section a performance evaluation methodology that simplifies the analysis is presented.

Jackson[15] studied an arbitrary network of queues with each node having an exponential service time and receiving requests from outside the system in the form of a Poisson process. Jackson showed that each node in such a system behaves as if it were a M/M/1 system with a Poisson arrival rate equal to the sum of all the original arrival rates at that node. Jackson's model was generalized in[16] to allow for different classes of requests and queueing disciplines other than FCFS. In our model of DDBS, all external arrival rates are Poisson. Because the base set of a transaction is assumed to be exponential and I/O service time is proportional to the number of items in the base set, I/O service time is exponential. Since the length of messages is exponentially distributed, the network service time is also exponential. Therefore, Jackson's assumptions are valid, and each node can be analyzed independently.

4.1 END TO END TRANSMISSION DELAY

The end-to-end transmission delay is defined as the time elapsed from the transmission of the message at its source to its destination. It is a key parameter of our analysis as our main objective is to study the effect of network load and network topology. Wong[17] has derived expressions for the distribution of end-to-end delay in message switched networks that satisfy independence assumptions[14]. His results can be summarized as follows.

Let

r denote a source destination pair (s,d);

a(r) denote the path (essentially an ordered set of channels) over which a message originating at s and destined for d is routed;

t_i denote mean service time of the ith channel that belongs to a(r);

ρ_i denote the total utilization of the ith channel; and

T_r denote the average end-to-end transmission delay.

Then the expression for T_r for a message switched network with fixed routing is given by:

$$T_r = \sum_{i \in a(r)} \frac{(t_i)}{(1-\rho_i)}$$

As the underlying (long haul) communication network for a DDBS satisfies independence assumptions, the above expression can be used to compute the end to end delay. It is of interest to note that each term in the summation is a function of the total utilization of the channel (ρ_i) and not of the utilization due to that class of messages. Therefore, the background traffic can be taken into account by subtracting it from the original channel capacity to derive the effective channel capacity which will be used in subsequent sections to study the effect of background traffic on performance of the 2-PC protocol.

5 PERFORMANCE EVALUATION RESULTS

In this section, the results for the 2-PC protocol for two cases of network topology, completely connected and star, are discussed.

Based on the assumptions given in the previous sections, queueing models are developed for the 2-PC protocol. The queueing models are derived as follows: The I/O service time and channel service time depend on the type of request. By inspecting the algorithm, one can determine the service time and relative frequency of each type of request. From this information equations for the mean service time, utilization and waiting time can be derived for the I/O servers and the network servers. A summary of these equations (for option 1 discussed in the third section) is given in the appendix. The response time can be computed by identifying the steps that a request has to go through and by summing up the time spent in the queue and processing at each step. Table 1 gives the parameter values assumed for plotting the graphs.

I	0.025 sec
$1/\mu$	5 items
$1/\mu_1$	125 bytes
$1/\mu_2$	125 bytes
$1/\mu_3$	5 k bytes
M	1000 items
N	12
S	200 bytes
B	5 items

Table 1: Parameter Values Used for Response Time Calculation

5.1 NODE BOUND vs. NETWORK BOUND

Using the analysis given in the appendix, it can be decided whether the DDBS under consideration is network bound or node bound by comparing the utilization of the most busy node with the most busy TC. These are the central node and a TC transmitting messages from a noncentral node to the central node, respectively, in both cases of network topology.

Thus a DDBS over a completely connected network will be node bound if

$$\rho(c) > \rho(chl)$$

Substituting the equations given in the appendix,

$$(5N + 1)\lambda I/\mu > \lambda(2S1 + S2 + 2S3)$$

$$\Rightarrow I > \frac{\mu}{(5N+1)} (2S1 + S2 + 2S3),$$

where $Si = 1/\mu_i C$.

A DDBS over a star network will be node bound if

$$\rho(c) > \rho(chl)$$

Substituting the equations given in the appendix,

$$(5N+1)\lambda I/\mu > ((2N-2)S1 + S2 + (2N-2)S3)\lambda$$

$$\Rightarrow I > \frac{\mu}{(5N+1)}((2N-2)S1+S2+(2N-2)S3),$$

where $Si = 1/\mu_i C$.

5.2 RESPONSE TIME ANALYSIS

Response times (R) and utilizations (U) for the two network topologies are plotted against the transaction arrival rate for the channel capacity of 50 Kbits/sec. in figures 3a and 3b, respectively. Figure 3b shows that the channel utilization in a star topology is higher than the utilization of the central node. However, in a completely connected network node utilization is higher than channel utilization. Thus in the case of star topology, the network is the bottleneck and the response time essentially follows the channel utilization curve. On the other hand, in the case of the completely connected topology, the system is node bound, and the response time is dependent on the central node utilization. Response time and utilization curves are plotted for the channel capacity of 10 Kbits/sec. in figures 4a and 4b, and for 100 Kbits/sec. in figures 5a and 5b. The curves for 100 Kbits/sec. follow the same trend as in the case of channel capacity of 50 Kbits/sec. In case of 10 Kbits/sec. the system is network bound for both cases of network topology and the response time curve follows the channel utilization curve.

In figure 6 response time curves are plotted for a star network for channel capacity of 50 Kbits/sec. under the assumption of constant transmission time (i.e. when queueing effects are ignored) and when queueing effects are not ignored. As the system is network bound in this case, there is a lot of difference between the two curves. In figure 7 response time curves are plotted for a completely connected network for channel capacity of 50Kbits/sec. for the two cases. As the system is node bound in this case, there is little difference between the two curves. Therefore, form the above discussion, we can see that the constant transmission time assumption is invalid when the system is network bound while it may be justified when the system is node bound.

Figure 8 plots the response time for the star network with different numbers of nodes. Increases in the number of nodes increase the channel and the node utilizations which result in higher response times.

All the curves in figures 3 through 8 are for the option 1 of the 2-PC implementation discussed in the third section. Figures 9 and 10 give response times for both options for a completely connected network and a star network respectively. For the completely connected network, since the system is node bound, option 1 performs better than option 2 for higher arrival rates. Response time for lower arrival rates is more for option 1 due to higher service rates at the channels. In the case of star network, since the system is network bound, option 2 performs better than option 1. Therefore, the relative performance of option 1 and option 2 depends on whether the system is node bound or network bound.

6 CONCLUSION

In this paper an analysis of the effect of network load and topology on the performance of a CCA is discussed. Earlier studies in the performance analysis of CCAs have assumed a constant message transmission time between any two nodes of a network. Our analysis shows that this assumption is not justified in context of long haul networks and that it is important to study the queueing effect in transmission channels. The response time of a 2-PC protocol is shown to be sensitive to communication subnet parameters such as network load, network topology, the capacity of transmission channels and the size of concurrency related messages. A completely connected network topology results in a system that is node bound whereas a star topology gives a system that is network bound. The analysis given in this paper can help identify DDBS performance bottleneck and consequently determine the steps to be taken to improve transaction throughput and response time. For example, if the system is network bound (for option 1 of 2-PC) then its performance can be improved by increasing the capacity of overloaded lines or by changing to a CCA (choosing the option 2) that decreases the size of concurrency related messages.

Several assumptions were made to simplify the analysis. Our future work will be relaxing some of these assumptions. There is also a need to study the effect of network load and topology on the performance of other kind of CCAs such as time-stamping and distributed locking.

7 APPENDIX

In this section equations for mean service time, utilization and response time are given.

7.1 EQUATIONS FOR MEAN SERVICE TIME AND UTILIZATION

Table 2 summarizes equations for mean service time and utilization for I/O servers and channel servers for two cases of network topology: completely connected and star.

Using the equations given in table 2 the mean wait time (W) for each of the servers can be determined by using the standard M/M/1 equation.

7.2 RESPONSE TIME FOR A COMPLETELY CONNECTED NETWORK

Non Conflict Case

The response time can be computed by identifying the steps that a request has to go

through and by summing up the time spent in the queue and in processing at each step. The average response time (R_{nc}) for a transaction originating at a noncentral node is as follows:

$$R_{nc} = 2W(ch1) + 2W(ch2) + 2/\mu_1 C + 1/\mu_2 C + 1/\mu_3 C + 2W(c) + 2W(nc) + 5I/\mu$$

The average response time (R_c) for a transaction originating at a central node is as follows:

$$R_c = W(ch1) + W(ch2) + 1/\mu_1 C + 1/\mu_3 C + W(nc) + 3W(c) + 5I/\mu$$

The expression for average response time is obtained by taking a weighted average of R_{nc} and R_c. It is given as follows:

$$R = \left(\frac{N-1}{N}\right)R_{nc} + \left(\frac{1}{N}\right)R_c$$

Response time equations for the case of a star network can be derived in a similar way.

Type of server	Mean Service Time X()	Utilization ρ
Noncentral node (nc)	I/μ	$(2N+1)\lambda I/\mu$
Central node (c)	$\frac{(5N+1)}{(4N+1)} I/\mu$	$(5N+1)\lambda I/\mu$
Completely conn.		
1)Noncentral to central (ch1)	$\frac{1}{5}(2S1+S2+2S3)^*$	$5\lambda X(ch1)$
2)Central to noncentral (ch2)	$\frac{1}{5}(3S1 + 2S3)$	$5\lambda X(ch2)$
3)Noncentral to noncentral (ch3)	$\frac{1}{4}(2S1 + 2S3)$	$4\lambda X(ch3)$
Star		
1)Noncentral to central (ch1)	$\frac{1}{4N-3}((N-1)(2S3+2S1)+S2)$	$(4N-3)\lambda X(ch1)$
2)Central to noncentral (ch2)	$\frac{1}{4N-3}(2(N-1)S3+(2N-1)S1)$	$(4N-3)\lambda X(ch2)$

(where $Si = 1/\mu_i C$)

Table 2: Equations for mean service time and Utilization

7.3 CONFLICT CASE

In case of conflicts the response time will be larger than the one given by the above equation due to additional wait time to acquire locks. The expression for this additional delay was derived in[5] and is given as follows:

$$R = R + N\lambda Pr(C)(L/2 + W(c) + I/\mu)$$

where

$Pr(C) = 1/\mu^2 M$ is the probability of conflict

L = Average time a transaction holds locks

The above equation is used iteratively until the increase in R is negligible. For the given parameter values increases in the response time due to conflicts turn out to be negligible.

REFERENCES

1. Eswaran K.P. et al,, "The Notions of Consistency and Predicate Locks in a Database System," CACM, Vol. 19, No. 11, November 19/6, pp. 624-633.

2. Gray J.,, Notes on Database Operating Systems, Springer-Verlag, N.Y.,N.Y., 1978, pp. 394-481.

3. Papadimitriou C.M.,, "Serializability of Concurrent Database Updates," JACM, Vol. 26, No. 4, October 1979, pp. 631-653.

4. Bernstein P.A. and Goodman N.,, "Concurrency Control in Distributed Database Systems, ACMCS, Vol. 13, No. 2, June 1981, pp. 185-221.

5. Garcia-Molina H.,, Preformance of Update Algorithms for Replicated Data in a Distributed Database, PhD dissertation, Dept. of Computer Science,Stanford Univ., June 1979.

6. Dantas J.E.R.,, Performance Analysis of Distributed Database System, PhD dissertation, Dept. of Computer Science,Univ. of California,Los Angeles., 1980.

7. Cheng W.K.,, Performance Analysis of Update Synchronization Algorithms for Distributed Databases, PhD dissertation, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign,Urbana,Illinois., 1981.

8. Thomas R. H., A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases ," ACM TODS, Vol. 4, No. 2, June 1979, pp. 180-209.

9. Galatianos G.A. and Tsai W-C.,, ``"Performance Evaluation of Database Update /synchronization on Ethernet Environments",'' Proceeding of the fourth Int. Conf. on Distributed Computing Systems, May 1984, pp. 503-512.

10. Sheth A.P., Singhal A., and Liu M.T.,, ``"An Adaptive Concurrency Control Strategy for Distributed Database Systems",'' Proc. First Int. Conf. on Data Engineering, April 1984, pp. 474-82.

11. Mohan C., ``Recent and Future Trends in Distributed Data Base Management,'' Proc. of NYU Symposium on New Directions in Data Base Systems, May 1984.

12. Mohan C. and Lindsay B., ``Efficient Commit Protocols for the tree of processes model of distributed transactions,'' Proc. of Second SIGACT-SIGOPS Symposium on Principles Of Distributed Computing, ACM, 1983, pp. 76-88.

13. Ries D.R.,, The Effect of Concurrency Control on Database Management System Performance, PhD dissertation, Computer Science Dept.,Univ. of California,Berkeley., April 1979.

14. Kleinrock L., Queueing Systems, Computer Applications, Wiley-Interscience, , Vol. 2, 1976.

15. Jackson J. R., ``Network of Waiting Lines,'' Networks of Waiting Lines, Vol. 5, 1957, pp. 518-521.

16. Baskett F., Chandy F. M., Muntz R. R. and Palacious F. G., ``Open, closed and mixed network of queues with different classes of customers,'' JACM, Vol. 22, 1975, pp. 248-260.

17. Wong J. W., ``Distribution of End-to-End Delay in Message Switched Networks,'' Computer Networks, Vol. 2, 1978, pp. 44-49.

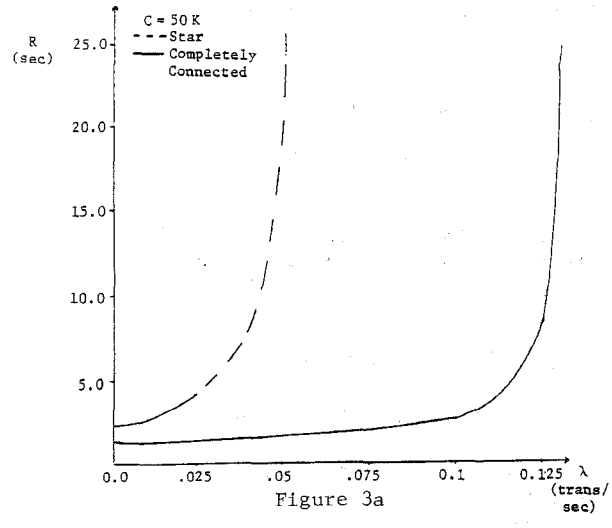

Figure 3a

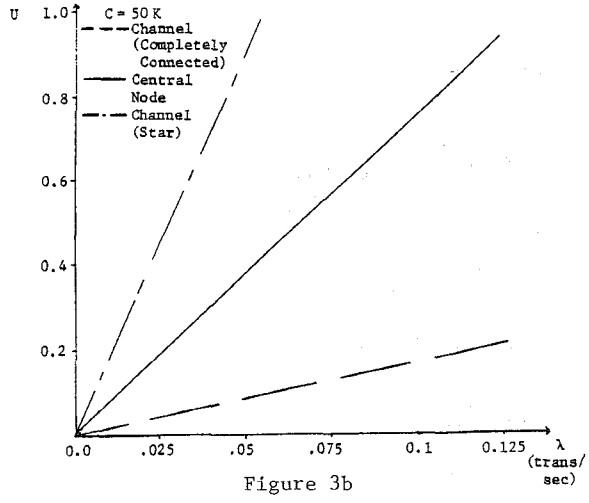

Figure 3b

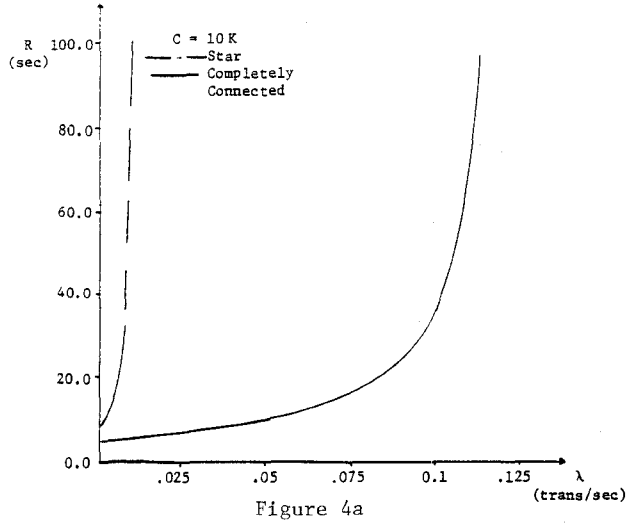

Figure 4a

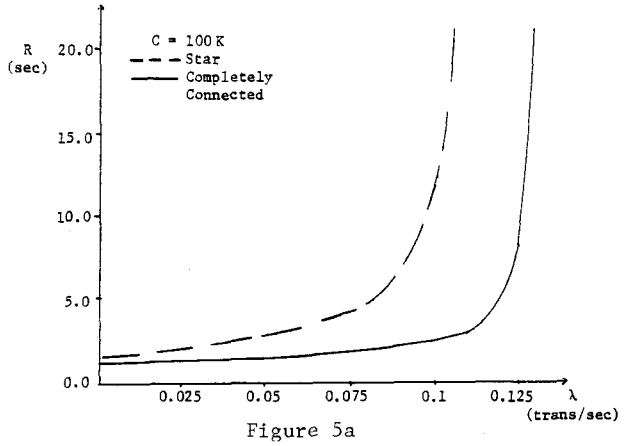

Figure 5a

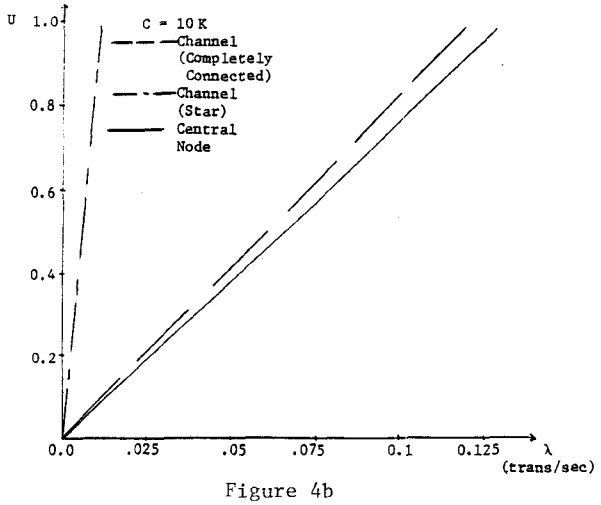

Figure 4b

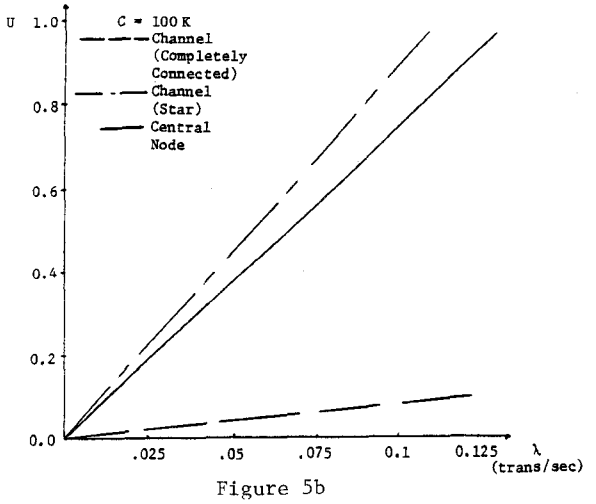

Figure 5b

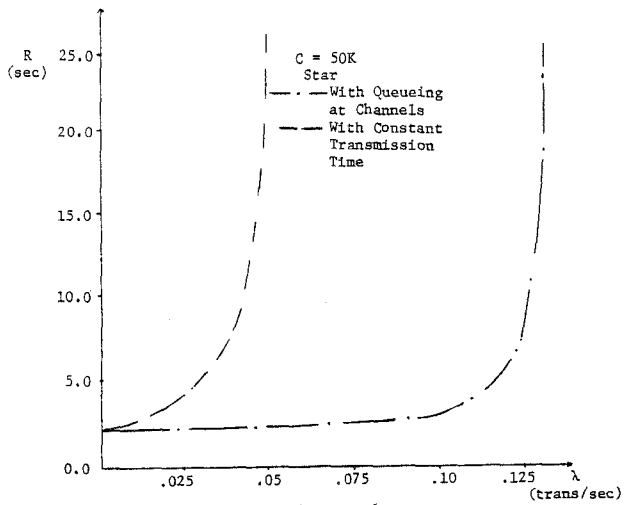

Figure 6

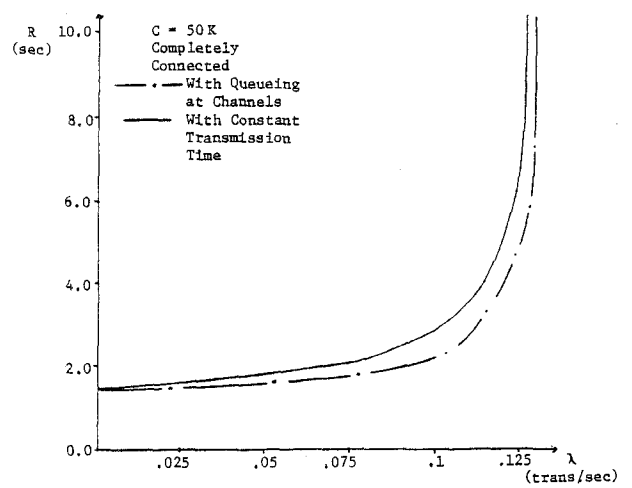

Figure 7

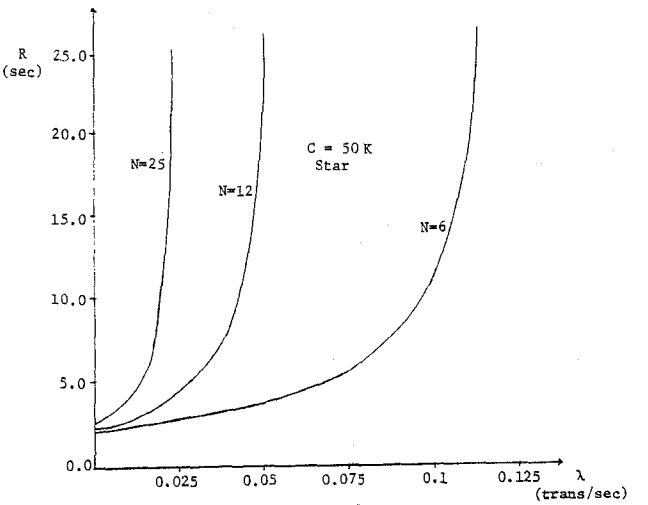

Figure 8

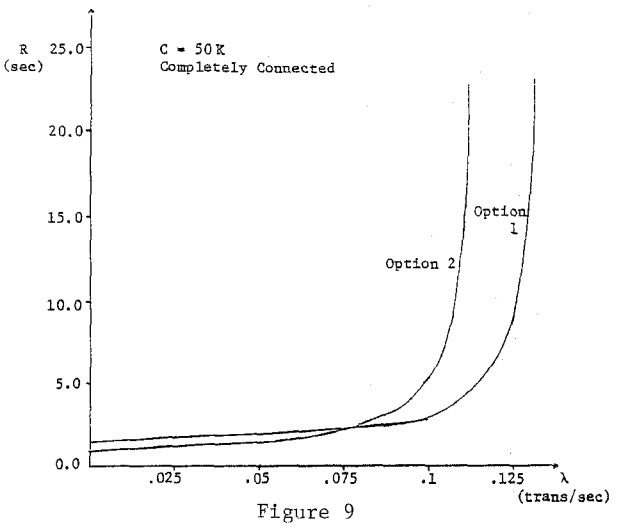

Figure 9

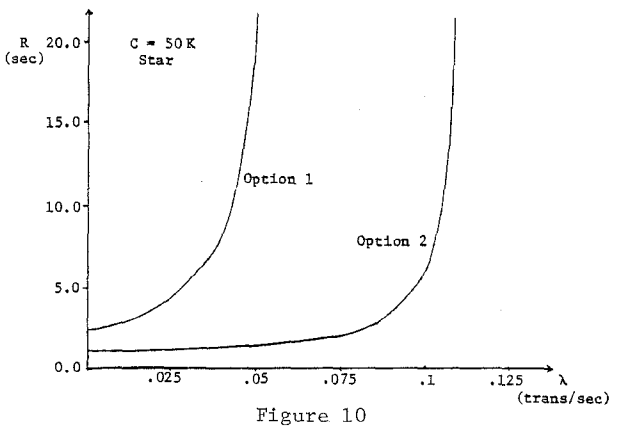

Figure 10